

Uma Mudenagudi

Aditya Nigam

Ravi Kiran Sarvadevabhatla

Ayesha Choudhary *Editors*

Proceedings of the Satellite Workshops of ICVGIP 2021

Lecture Notes in Electrical Engineering

Volume 924

Series Editors

Leopoldo Angrisani, Department of Electrical and Information Technologies Engineering, University of Napoli Federico II, Naples, Italy

Marco Arteaga, Departament de Control y Robótica, Universidad Nacional Autónoma de México, Coyoacán, Mexico

Bijaya Ketan Panigrahi, Electrical Engineering, Indian Institute of Technology Delhi, New Delhi, Delhi, India

Samarjit Chakraborty, Fakultät für Elektrotechnik und Informationstechnik, TU München, Munich, Germany

Jiming Chen, Zhejiang University, Hangzhou, Zhejiang, China

Shanben Chen, Materials Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

Tan Kay Chen, Department of Electrical and Computer Engineering, National University of Singapore, Singapore, Singapore

Rüdiger Dillmann, Humanoids and Intelligent Systems Laboratory, Karlsruhe Institute for Technology, Karlsruhe, Germany

Haibin Duan, Beijing University of Aeronautics and Astronautics, Beijing, China

Gianluigi Ferrari, Università di Parma, Parma, Italy

Manuel Ferre, Centre for Automation and Robotics CAR (UPM-CSIC), Universidad Politécnica de Madrid, Madrid, Spain

Sandra Hirche, Department of Electrical Engineering and Information Science, Technische Universität München, Munich, Germany

Faryar Jabbari, Department of Mechanical and Aerospace Engineering, University of California, Irvine, CA, USA

Limin Jia, State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China

Janusz Kacprzyk, Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

Alaa Khamis, German University in Egypt El Tagamoa El Khames, New Cairo City, Egypt

Torsten Kroeger, Stanford University, Stanford, CA, USA

Yong Li, Hunan University, Changsha, Hunan, China

Qilian Liang, Department of Electrical Engineering, University of Texas at Arlington, Arlington, TX, USA

Ferran Martín, Departament d'Enginyeria Electrònica, Universitat Autònoma de Barcelona, Bellaterra, Barcelona, Spain

Tan Cher Ming, College of Engineering, Nanyang Technological University, Singapore, Singapore

Wolfgang Minker, Institute of Information Technology, University of Ulm, Ulm, Germany

Pradeep Misra, Department of Electrical Engineering, Wright State University, Dayton, OH, USA

Sebastian Möller, Quality and Usability Laboratory, TU Berlin, Berlin, Germany

Subhas Mukhopadhyay, School of Engineering & Advanced Technology, Massey University,

Palmerston North, Manawatu-Wanganui, New Zealand

Cun-Zheng Ning, Electrical Engineering, Arizona State University, Tempe, AZ, USA

Toyoaki Nishida, Graduate School of Informatics, Kyoto University, Kyoto, Japan

Luca Oneto, Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genova, Genova, Genova, Italy

Federica Pascucci, Dipartimento di Ingegneria, Università degli Studi "Roma Tre", Rome, Italy

Yong Qin, State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China

Gan Woon Seng, School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore, Singapore

Joachim Speidel, Institute of Telecommunications, Universität Stuttgart, Stuttgart, Germany

Germano Veiga, Campus da FEUP, INESC Porto, Porto, Portugal

Haitao Wu, Academy of Opto-electronics, Chinese Academy of Sciences, Beijing, China

Walter Zamboni, DIEM - Università degli studi di Salerno, Fisciano, Salerno, Italy

Junjie James Zhang, Charlotte, NC, USA

The book series *Lecture Notes in Electrical Engineering* (LNEE) publishes the latest developments in Electrical Engineering—quickly, informally and in high quality. While original research reported in proceedings and monographs has traditionally formed the core of LNEE, we also encourage authors to submit books devoted to supporting student education and professional training in the various fields and applications areas of electrical engineering. The series cover classical and emerging topics concerning:

- Communication Engineering, Information Theory and Networks
- Electronics Engineering and Microelectronics
- Signal, Image and Speech Processing
- Wireless and Mobile Communication
- Circuits and Systems
- Energy Systems, Power Electronics and Electrical Machines
- Electro-optical Engineering
- Instrumentation Engineering
- Avionics Engineering
- Control Systems
- Internet-of-Things and Cybersecurity
- Biomedical Devices, MEMS and NEMS

For general information about this book series, comments or suggestions, please contact leontina.dicecco@springer.com.

To submit a proposal or request further information, please contact the Publishing Editor in your country:

China

Jasmine Dou, Editor (jasmine.dou@springer.com)

India, Japan, Rest of Asia

Swati Meherishi, Editorial Director (Swati.Meherishi@springer.com)

Southeast Asia, Australia, New Zealand

Ramesh Nath Premnath, Editor (ramesh.premnath@springernature.com)

USA, Canada

Michael Luby, Senior Editor (michael.luby@springer.com)

All other Countries

Leontina Di Cecco, Senior Editor (leontina.dicecco@springer.com)

**** This series is indexed by EI Compendex and Scopus databases. ****

Uma Mudenagudi · Aditya Nigam ·
Ravi Kiran Sarvadevabhatla · Ayesha Choudhary
Editors

Proceedings of the Satellite Workshops of ICVGIP 2021

Editors

Uma Mudenagudi
KLE Technological University
Hubli, India

Aditya Nigam
Indian Institute of Technology Mandi
Mandi, India

Ravi Kiran Sarvadevabhatla
International Institute of Information
Technology Hyderabad
Hyderabad, India

Ayesha Choudhary
Jawaharlal Nehru University
New Delhi, India

ISSN 1876-1100

ISSN 1876-1119 (electronic)

Lecture Notes in Electrical Engineering

ISBN 978-981-19-4135-1

ISBN 978-981-19-4136-8 (eBook)

<https://doi.org/10.1007/978-981-19-4136-8>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Disclaimer: The presentation of material and details in maps used in this chapter does not imply the expression of any opinion whatsoever on the part of the Publisher or Author concerning the legal status of any country, area or territory or of its authorities, or concerning the delimitation of its borders. The depiction and use of boundaries, geographic names and related data shown on maps and included in lists, tables, documents, and databases in this chapter are not warranted to be error free nor do they necessarily imply official endorsement or acceptance by the Publisher or Author.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd. The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Contents

Workshop on Digital Heritage (WDH)

Systematic Approach to Tuning a Deep CNN Classifying Bharatanatyam Mudras	3
R. Jisha Raj, Smitha Dharan, and T. T. Sunil	
Comparative Analysis of Neural Architecture Search Methods for Classification of Cultural Heritage Sites	25
Sunil V. Gurlahosur, S. M. Meena, Uday Kulkarni, Winston Dcosta, Vineet Lokur, Rohan V. Sirigeri, Sajal Porwal, S. P. Sammed, and Uma Mudenagudi	
Heritage Representation of Kashi Vishweshwar Temple at Kalabgoor, Telangana with Augmented Reality Application Using Photogrammetry	43
Tejas Pawar, Aman Sharma, and Shiva Ji	
Augmented Data as an Auxiliary Plug-In Toward Categorization of Crowdsourced Heritage Data	53
Shashidhar Veerappa Kudari, Akshaykumar Gunari, Adarsh Jamadandi, Ramesh Ashok Tabib, and Uma Mudenagudi	
Evolution of Bagbazar Street Through Visibility Graph Analysis (1746–2020)	63
Shilpi Chakraborty and Shiva Ji	
Mapping Archaeological Remains of 14th Century Fort of Jahanpanah Using Geospatial Analysis	79
Gaurav Kumar Pal and M. B. Rajani	
Spatial Analysis and 3d Mapping Historic Landscapes—Implications of Adopting an Integrated Approach in Simulation and Visualization of Landscapes	87
Mythrayi Harshavardhan	

Medical Image Processing (MedImage)	
HSADML: Hyper-Sphere Angular Deep Metric Based Learning for Brain Tumor Classification	105
Aman Verma and Vibhav Prakash Singh	
Document Analysis and Recognition (DAR)	
Model Compression Based Lightweight Online Signature Verification Framework	123
Chandra Sekhar Vorugunti, S. Balasubramanian, Pulabaigari Viswanath, and Avinash Gautam	
End-to-End Transformer-Based Architecture for Text Recognition from Document Images	135
Dipankar Ganguly, Akkshita Trivedi, Bhupendra Kumar, Tushar Patnaik, and Santanu Chaudhury	
A Hybrid Approach for Table Detection in Document Images	147
Sunil Kumar Vengalil, Kevin Xavier, Konda Amith Sai, Sree Harsha, Ganesh Barma, and Neelam Sinha	
Workshop on Computer Vision Applications (WCVA)	
The Ikshana Hypothesis of Human Scene Understanding	161
Venkata Satya Sai Ajay Daliparthi	
Worst-Case Adversarial Perturbation and Effect of Feature Normalization on Max-Margin Multi-label Classifiers	183
Ritesh Kumar Gupta and Yashaswi Verma	
Catch Me if You Can: A Novel Task for Detection of Covert Geo-Locations (CGL)	199
Binoy Saha and Sukhendu Das	
MATIC: Memory-Guided Adaptive Transformer for Image Captioning	219
Gaurav O. Gajhiye and Abhijeet V. Nandedkar	
Semantic Map Injected GAN Training for Image-to-Image Translation	235
Balaram Singh Kshatriya, Shiv Ram Dubey, Himangshu Sarma, Kunal Chaudhary, Meva Ram Gurjar, Rahul Rai, and Sunny Manchanda	
TextGen3D: A Real-Time 3D-Mesh Generation with Intersecting Contours for Text	251
Ankit Dhiman, Praveen Agrawal, Sourav Kumar Bose, and Basavaraja Shanthappa Vandrotti	

About the Editors

Uma Mudenagudi is Dean, R&D, and Professor of ECE department at KLE Technological University Hubballi. She completed her Ph.D. from the Department of Computer Science, IIT Delhi, and M.Tech. degree from IIT Bombay. Her research areas are computer vision, computer graphics, image, and video analysis. She has got over 70 projects to her credit.

Aditya Nigam received his Master and Doctoral degrees from the Indian Institute of Technology Kanpur in 2009 and 2014, respectively. Presently, he is an Assistant Professor at IIT Mandi in the School of Computing and Electrical Engineering (SCEE). His research areas are biometrics, image processing, computer vision, and machine learning. He has several papers published to his credit.

Ravi Kiran Sarvadevabhatla is an Assistant Professor at the International Institute of Information Technology, Hyderabad. His work is primarily in the area of computer vision and applied machine learning. He has broad-ranging research interests and likes to work on inter-disciplinary problems involving multi-modal multimedia data (e.g., images, videos, text, audio/speech, eye-tracking data) and disciplines (e.g., graphics, robotics, human-computer interaction). He has got over 20 papers published to his credit.

Ayesha Choudhary is an Assistant Professor at the School of Computer & Systems Sciences at Jawaharlal Nehru University, India. She completed her Ph.D. from the Department of Computer Science and Engineering at the Indian Institute of Technology, Delhi (IIT Delhi). Her areas of research are computer vision and machine learning and their applications in Intelligent Transportation Systems, Assisted Living and Smart Agriculture. She has publications in various international journals and conferences to her credit.

Workshop on Digital Heritage (WDH)

Systematic Approach to Tuning a Deep CNN Classifying Bharatanatyam Mudras



R. Jisha Raj , Smitha Dharan , and T. T. Sunil 

1 Introduction

Dance and music are an inevitable part of human culture across the world, and it reflects the lives, beliefs and cultural traditions of people practising it. Archaeologists have excavated sculptures and paintings of dancers across India [3, 15, 21]. Sangeet Nataka Academy has identified eight different classical dances in India [16].

Bharatanatyam is a classical dance that originated in South India and is performed by both men and women [30]. The performers of Bharatanatyam dance use hand gestures (mudras), facial expressions and body movements that are very graceful [24]. They use these media to communicate to the audience the intended meaning of the verse or shloka being sung with the accompaniment of traditional musical instruments.

According to Natyashastra, a classical text on Indian dance, there are 28 Asamyukta Hastas (single hand gestures) and 23 Samyukta Hastas (Double hand gestures) [7, 9, 23] in Bharatanatyam. Computer vision techniques can be applied to Bharatanatyam mudras to obtain a better understanding and annotating a dance performance. Open datasets on Bharatanatyam hand gestures are not presently available. An exhaustive Bharatanatyam Mudra dataset consisting of 15,396 static single hand gesture images and 13,035 static double hand gesture images was proposed by the authors and is available in public domain now. The full dataset can be downloaded from <https://github.com/jisharajr/Bharatanatyam-Mudra-Dataset.git>. Using this dataset classification using conventional machine learning techniques, feature descriptors and visualisation was done and are being reported elsewhere.

R. Jisha Raj (✉) · S. Dharan · T. T. Sunil
College of Engineering, Chengannur, Attingal, Kerala, India
e-mail: vu2swx@gmail.com

S. Dharan
e-mail: smitha@ceconline.edu

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022
U. Mudenagudi et al. (eds.), *Proceedings of the Satellite Workshops of ICVGIP 2021*,
Lecture Notes in Electrical Engineering 924,
https://doi.org/10.1007/978-981-19-4136-8_1

In this paper, a deep convolutional neural network (CNN) is used for classification of the Bharatanatyam hand gesture images. The hyperparameter tuning of this CNN is researched and discussed. Hyperparameters of a CNN are dataset dependent. Various hyperparameters such as number of epochs, batch size, dropout probability and learning rate are tuned to obtain a high accuracy CNN classifier. Performance variation with different optimisers and the effect of validation split are also explored.

2 Literature Review

In the context of Bharatanatyam mudras, studies based on very limited datasets were made by few authors. Mozarkar et al. [20] used 68 samples of 13 static double hand gestures of Bharatanatyam to develop a recognition system, and Sriparna Saha et al. [24] used 28 single hand gestures of Bharatanatyam to develop such a system. A rotation and scale invariant gesture recognition system that could be applied for Bharatanatyam dance mudra recognition was experimented upon by Divya et al. [12]. A CNN-based classifier for pose and hand gesture classification developed by Aparna Mohanty et al. [19] used 1400 samples of 10 single hand gestures and 840 samples of 14 double hand gestures of Bharatanatyam.

Hyperparameter tuning is very essential for training a CNN classifier. Many authors have researched on this area. Radiuk et al. studied the effect of batch size on CNN using MNIST and CIFAR-10 datasets [22]. Batch sizes of the power of 2 and also 50, 100, 150 and 200 were tested by him. Some authors have suggested batch size above 64 was optimal [28, 31]. Bengio et al. suggested that using batch size 32 was a good choice [2]. This was one of his many recommendations for gradient-based training. Masters et al., however, suggested using batch sizes between 2 and 32 for robust deep neural networks [18].

The study of batch size and learning rate variation for Stochastic Gradient Descent (SGD) and Adam optimisers was made by Kandel et al. for histopathological dataset [14]. Tuning of learning rate, batch size, momentum and weight decay in a disciplined way was studied by Smith [26]. Adaptive learning rates in deep learning were studied by Chandra et al. [6]. An efficient dropout mechanism was explored by Cai et al. [5]. Effect of dropout and batch normalisation was studied by the authors in [8]. Controlled dropout, a different approach to dropout was explored by Ko et al. [17]. Shivam et al. have proposed a method to determine the optimum number of epochs needed to train the network [25].

3 Dataset Description

In Bharatanatyam, there are 28 single hand gestures (Asamyukta Hastas) and 23 double hand gestures (Samyukta Hastas) [7, 9, 23]. 15,396 single hand gesture images and 13,035 double hand gesture images were collected from 15 volunteers.



Fig. 1 Six different views of **a** *Mayura mudra*, one of the single hand mudras **b** *Kurma mudra*, one of the double hand mudras

All volunteers were trained in Bharatanatyam for more than 5 years. One of the single hand mudras, *Samdamsam Mudra* being dynamic was excluded. *Katakamukha mudra* has three variations, and each of them was taken as a separate class. Thus 29 classes of static single hand mudras were included in the dataset. *Utsanga Mudra* and *Dola Mudra* were two double hand gesture images excluded from the dataset. Images of these two mudras included a large part of the dancer's body. Thus double hand mudras in the dataset comprised of 21 different classes. Six different views of *Mayura mudra*, one of the single hand mudras and *Kurma mudra*, one of the double hand mudras are shown in Fig. 1a, b respectively.

The images were captured in a studio environment using Apple iPhone 6S 12 megapixel camera. The original images were in RGB format of size 3000×4000 . The camera was mounted on a tripod stand and kept at one metre distance from the screen. A green screen served as the permanent background. The data collection environment is shown in Fig. 2.

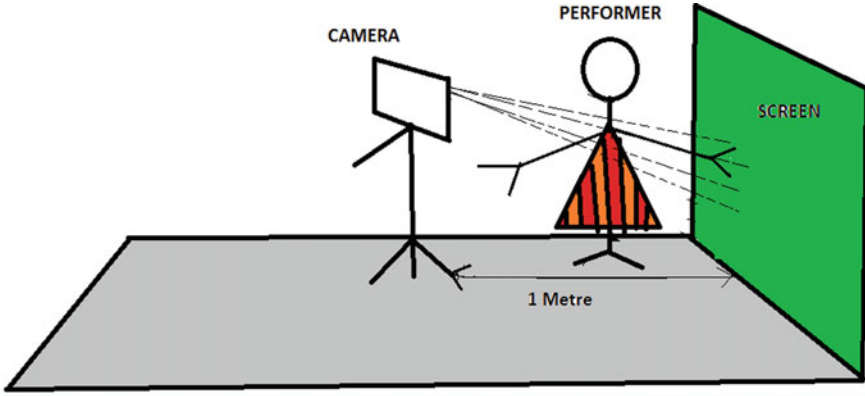


Fig. 2 Data collection environment

During preprocessing of the images, all images were resized to 100×100 and grayscale converted. Each channel is assigned a weight according to the colour and the formula [4, 13] used to obtain the grayscale image is:

$$R = G = B = 0.298R + 0.58G + 0.114B \quad (1)$$

Pixel values of each image were normalised by dividing them by 255.

4 Methodology

A deep CNN that provides accurate classification of hand gesture images of the Bharatanatyam Mudra dataset is proposed. Various hyperparameter tuning procedures that result in an accurate CNN classifier for the dataset are investigated.

4.1 Convolutional Neural Network

CNN is a structure that consists of convolutional layers, pooling layers and fully connected layers. The convolutional layers and pooling layers make up the feature extraction layers. The abstract high level features are extracted by these layers. The fully connected layers along with the softmax layer do the classification.

Convolutional Layers The convolutional layer is used to extract features from the image. Many different filters are used in each layer for the purpose. Filters of specific sizes are used to slide over the image and perform a convolution operation.

Convolution results in feature maps. These feature maps are fed to other layers for extracting more abstract features.

As number of filters in each layer increases layer becomes wider. This leads to identification of more features in the image with the large number of parameters. Adding more layers can also make the network deeper and improve the generalisation capacity of the model. Deep neural networks become more powerful with greater depth and width [26].

Pooling Convolution operations are followed by pooling operations. Pooling is done by downsampling the feature maps. It lowers the computational burden by reducing the number of connections between convolutional layers [11]. Max pooling and average pooling are usually used.

Fully connected layers The outputs from the convolutional layers are flattened and given to fully connected layers that are then connected to the softmax layer, which is the output layer. The softmax layer has nodes equal to the class number in the dataset.

4.2 Hyperparameter Tuning

Hyperparameters are those that are determined outside the learning algorithm [10]. They are not used for the calculation of loss function. The hyperparameters are tuned such that the network does not overfit or underfit. This paper explores effect of validation splits and performance of different optimisers along with the tuning of hyperparameters like batch size, learning rate, dropout probability and number of epochs.

4.2.1 Optimisers

We have explored the performance of various optimisers like Stochastic Gradient Descent (SGD) with and without momentum, Adam, RMSProp and AdaGrad.

Stochastic Gradient Descent (SGD) In Stochastic Gradient Descent Algorithm, weight update occurs at every data point. In mini-batch SGD, the gradient is calculated for every B number of training instances that form a batch. In momentum-based Stochastic Gradient Descent Algorithm, an update rule is used for a mini-batch of m examples from the training set and the formula is [10]

$$\begin{aligned}
v &\leftarrow \alpha v - \epsilon \nabla_{\theta} \left\{ \frac{1}{m} \sum_{i=0}^m (L(f(x^{(i)}; \theta), y^{(i)})) \right\} \\
\theta &= \theta + v
\end{aligned} \tag{2}$$

where α is the momentum parameter, ϵ is the learning rate, θ is initial parameter and v is initial update [10].

The learning rate and also the momentum parameter were varied. Learning rate of 0.01 with momentum of 0.9 gave the best results.

AdaGrad It is an adaptive learning rate optimisation algorithm. In AdaGrad algorithm, the parameters with the largest partial derivative of the loss have a correspondingly rapid decrease in their learning rate, while parameters with small partial derivatives have a relatively small decrease in their learning rate [10]. The net effect is greater progress in the more gently sloped directions of parameter space. The update equation for AdaGrad is given as [10]

$$\begin{aligned}
g &\leftarrow \frac{1}{m} \nabla_{\theta} \sum_i (L(f(x^{(i)}; \theta), y^{(i)})) \\
r &\leftarrow r + g \odot g \\
\Delta\theta &\leftarrow - \frac{\epsilon}{\delta + \sqrt{r}} \odot g \\
\theta &= \theta + \Delta\theta
\end{aligned} \tag{3}$$

Here ϵ is learning rate, θ is the initial parameter, r is the accumulate square gradient, δ is of order 10^{-7} and m is the number of examples in mini-batch.

For all learning rates except 0.001 the model was giving poor validation accuracy. Even for this learning rate, validation loss curves were not as required.

RMSProp The RMSProp algorithm modifies AdaGrad to perform better in the non-convex setting by changing the gradient accumulation into an exponentially weighted moving average [10]. AdaGrad is designed to converge rapidly when applied to a convex function [10, 29].

$$\begin{aligned}
g &\leftarrow \frac{1}{m} \nabla_{\theta} \sum_i (L(f(x^{(i)}; \theta), y^{(i)})) \\
r &\leftarrow \rho r + (1 - \rho) g \odot g \\
\Delta\theta &\leftarrow - \frac{\epsilon}{\delta + \sqrt{r}} \odot g \\
\theta &= \theta + \Delta\theta
\end{aligned} \tag{4}$$

Here, ϵ is the learning rate, θ is the initial parameter, δ is a small constant for numerical stabilisation (suggested value is 10^{-6}), r is the squared gradient accumulation variable.

Learning rate was the only parameter that was varied. Learning rate of 0.001 gave validation loss curves that were not flat.

Adam It is also an adaptive learning rate optimisation algorithm. The term Adam refers to ‘adaptive moments’ [10]. Here, the biased first and second moments have correction factors. Adam optimiser is fairly robust against hyperparameter variations [10].

$$\begin{aligned}
 g &\leftarrow 1/m \nabla_{\theta} \sum_i (L(f(x^{(i)}; \theta), y^{(i)})) \\
 t &\leftarrow t + 1 \\
 s &\leftarrow \rho_1 s + (1 - \rho_1) g \\
 r &\leftarrow \rho_2 r + (1 - \rho_2) g \odot g \\
 \hat{s} &\leftarrow \frac{s}{\sqrt{1 + \rho_1^t}} \\
 \hat{r} &\leftarrow \frac{r}{\sqrt{1 + \rho_2^t}} \\
 \Delta\theta &\leftarrow -\epsilon \frac{\hat{s}}{\sqrt{\hat{r} + \delta}} \\
 \theta &= \theta + \Delta\theta
 \end{aligned} \tag{5}$$

Here, ϵ is the learning rate, θ is the initial parameter, δ is a small constant for numerical stabilisation (suggested value is 10^{-8}), ρ_1 and ρ_2 are exponential decay rates for moment estimates (suggested values are 0.9 and 0.999 respectively) [10].

Adam with a learning rate 0.001 gave asymptotic validation loss curves with good accuracy. Higher values of learning rates like 0.01 and 0.1 gave very low validation accuracies.

4.2.2 Batch Size

Batch size is the hyperparameter of the gradient descent algorithm. It is the number of training instances the algorithm goes through before the model parameters are updated [2, 31].

1. For Batch Gradient Descent, batch size = training dataset
2. For Stochastic gradient descent, batch size = 1
3. For Mini-batch stochastic gradient descent, batch size = subset of training instances.

Batch sizes are usually varied in powers of 2 [28]. Batch sizes of 16, 32, 64, 128, 256 and 512 are usually used.

4.2.3 Dropout

It is a regularisation technique used to remove overfitting. Regularisation techniques are used to make the models behave with minimum validation error. The idea is to randomly drop units or nodes (along with their connections) from the neural network during training. This prevents units from co-adapting too much [27].

4.2.4 Learning Rate

The learning rate is a hyperparameter that controls the rate at which model weights are updated (in response to the estimated error) to minimise the loss function. Small learning rates converge slowly and get stuck in a local minimum [1] of the parametric subspace. This can cause the training to be longer and may get stuck. If the learning rate is large it may overshoot the global minimum [1]. Thus as the learning rate increases oscillations increase.

Learning rate is tuned with different values on the log scale i.e. 0.00001, 0.0001, 0.001, 0.01, 0.1, 1 etc. For a fixed dataset and architecture it is better to start tuning with a large learning rate [26]. Since the CNN architecture is 6-layer deep, Adam optimiser with an initial learning rate of 0.001 was used.

4.2.5 Epochs

It is one pass of the learning algorithm over the entire dataset. The number of epochs should be chosen such that it mitigates chances of overfitting. The network should be trained only upto the point when the validation accuracy starts falling or generalisation error starts increasing. Thus epoch determination is based on a hit-and-trial basis [25].

4.2.6 Validation Split

Percentage of validation split decides how many of the dataset images are reserved for training and how many for validation purposes. For initial trials, 30% validation split was used. But the model showed performed better with 40% validation split. Thus out of 15,396 single hand gesture images, 9238 were kept as training images and 6158 as validation images. Out of 13,035 double hand gesture images, 7821 were training and 5214 were validation images.

5 Results and Discussion

5.1 Architecture

Number of layers in a model is primarily decided by the size of the dataset. Since the dataset was of moderate size, an eight layer network was the initial model. It consisted of five convolution layers, two fully connected layers and an output layer. Through trials, it was observed that one of the convolution layer and one fully connected layer were redundant. They were ablated to obtain a more compact network. Trials made for deciding the number of convolution and fully connected layers are shown

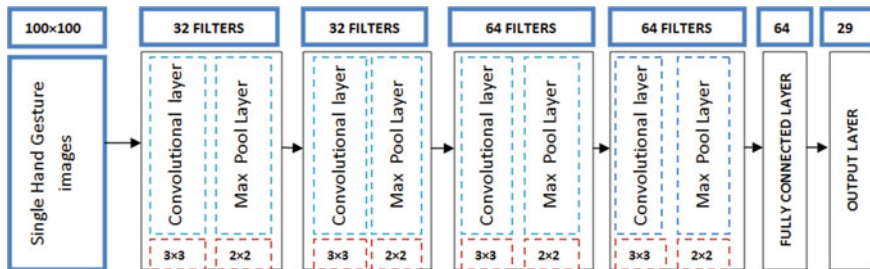


Fig. 3 The architecture of CNN used for classifying single hand gesture images

Table 1 Trials done to ablate the redundant layers using single hand gesture images

Number of convolution and fully connected layers	Accuracy (%)
Five Convolution and two fully connected layers	94
Five Convolution and one fully connected layers	96
Four Convolution and one fully connected layers	96.6

in Table 1. Trials were done using batch size of 32, Adam optimiser and 40 epochs. A validation split of 30% was used for these trials. The filter sizes were kept as minimum or 3×3 . Max pooling was used with a 2×2 stride of the window. Dropout of 50% was included after the fully connected layer. All experiments were done on a machine with a Core i9 processor, 32GB RAM and RTX 2080 GPU. The CNN model used for classifying 15,396 single hand gesture images of 29 classes is shown in Fig. 3. The same network with output layer (softmax layer) of 21 nodes was used for classifying 13,035 double hand gesture images of 21 classes. All images were resized to 100×100 and grayscale converted. Pixel normalisation was also done.

5.2 Batch Sizes

Since the dataset is already fixed and architecture have been decided next step is tuning various hyperparameters. Different batch sizes like 32, 64, 128, 256 and 512 were attempted keeping the dropout at 50% and number of epochs at 40 for single and 30 for double hand gesture images. Table 2 shows the details of the trials done on different batch sizes for single and double hand gesture images. Since validation loss curves are good indicators of a network's convergence, these curves are used for studying the performance of the network [26]. The validation loss curves for different batch sizes on using single and double hand gesture images are shown in Fig. 4a, b respectively. It was observed that batch size of 32 provided the most accuracy of 96.6% at the 36th epoch and a more asymptotic curve was obtained. For double hand

Table 2 Trials done on batch sizes using (a) single hand gesture images (b) double hand gesture images

Batch size	Time for training in 40 epochs	Accuracy %
32	4 min	96.6
64	3 min	96
128	2 min 30 sec	95.5
256	1 min 30 sec	94.81
512	1 min	94.3

(a)

Batch size	Time for training in 30 epochs	Accuracy %
32	3 min	99.25
64	2 min 44 sec	98.79
128	2 min	98.50
256	1 min 20 sec	98.37
512	55 sec	98.2

(b)

gesture images, the highest accuracy of 99.25% was obtained for batch size 32 at 28th epoch and a flattened curve was obtained (Table 2).

5.3 Dropout Probability

After deciding on the batch size, the dropout was varied. Table 3 shows how the dropout variation influenced the accuracy when single and double hand gesture images are used. For a batch size of 32, a validation split of 30% was maintained and training was completed in 40 epochs for single and 30 epochs for double hand gesture images. It was observed that when the heavy dropout of 50% was used, the validation accuracy was slightly greater than training accuracy. With a dropout of 10%, an asymptotic validation curve was obtained with validation accuracy lesser than training accuracy for both single and double hand gesture images. But at 10% dropout, the accuracy on using single hand gesture images dropped to 95.5%. Similar trials were also done for double hand gesture images, and it was found that accuracy

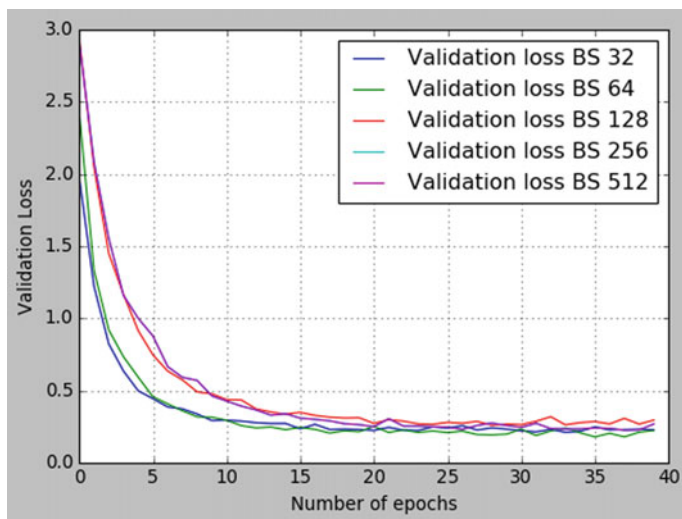
Table 3 Shows how dropout influences accuracy of (a) single hand gesture images (b) double hand gesture images

Dropout	Accuracy %
0.1	95.5
0.2	95.8
0.3	96
0.4	96.3
0.5	96.6
0.6	96

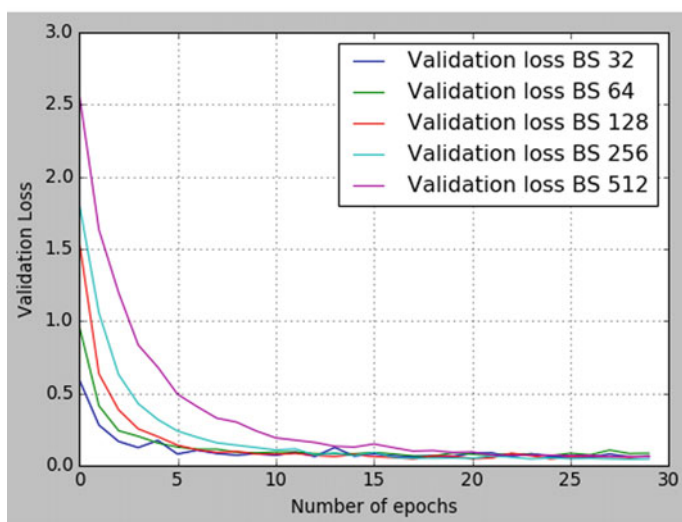
(a)

Dropout	Accuracy %
0.1	97.8
0.2	98.2
0.3	98.6
0.4	98.9
0.5	99.1
0.6	99

(b)

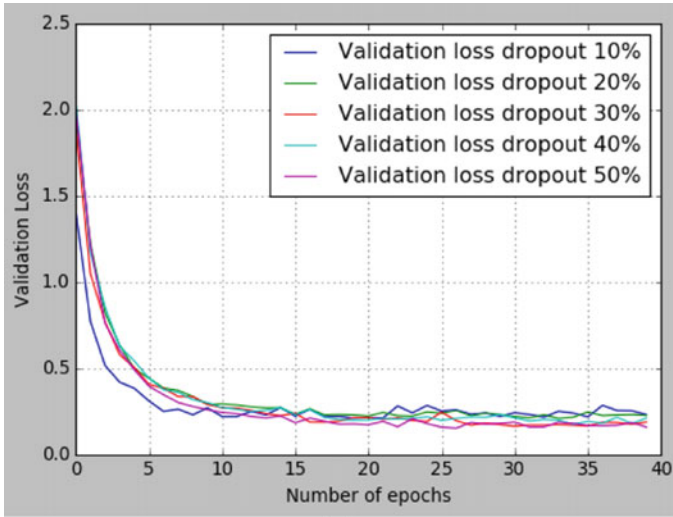


(a)

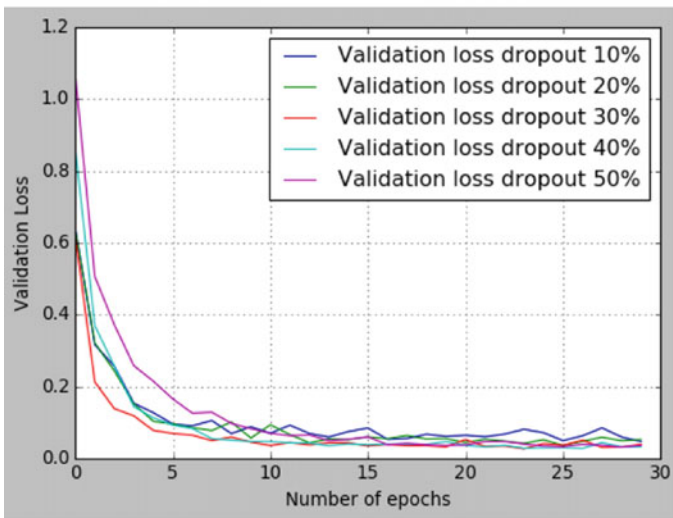


(b)

Fig. 4 Validation loss for **a** single hand gesture images **b** double hand gesture images for different batch sizes



(a)



(b)

Fig. 5 Validation loss for **a** single hand gesture images **b** double hand gesture images for different dropout probabilities

dropped to 97.8% at 10% dropout but a more flattened curve was obtained. The validation loss curves for different dropout probabilities on using single and double hand gesture images are shown in Fig. 5a, b respectively.

5.4 Validation Split Percentage

Validation split also plays an important role in making the validation curve more asymptotic. If the model is presented with more difficult validation data, accuracy may decrease slightly but the validation loss curve is more asymptotic. The validation loss curves for different validation split probabilities on using single and double hand gesture images are shown in Fig. 6a, b respectively. The validation loss curve has flattened more for validation split of 40%. Loss curves getting flattened and achieving this horizontal part is the main goal of hyperparameter tuning [26].

5.5 Optimisers and Learning Rate

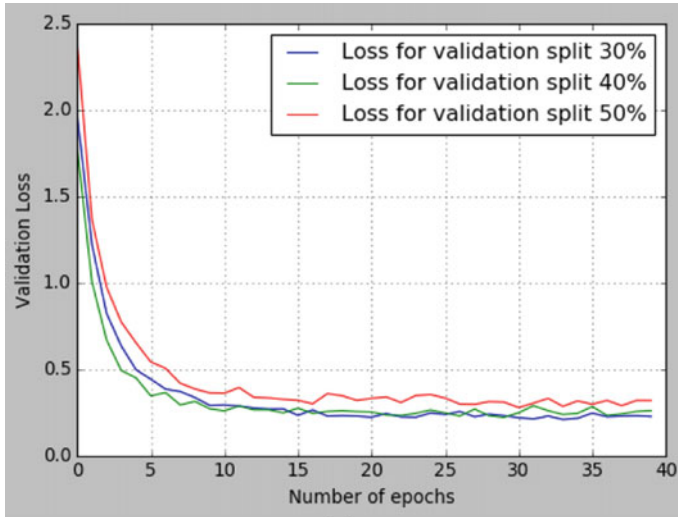
After having fixed the architecture, batch size, dropout probabilities, epochs and validation split different optimisers were attempted. For all the above attempts Adam optimiser with a learning rate 0.001 was used. Trials were made with other optimisers like SGD, RMSProp and AdaGrad to find whether a better alternative existed. The validation loss curves for different optimisers on using single and double hand gesture images are shown in Fig. 7a, b respectively. It is observed that Adam with a learning rate 0.001 and SGD (learning rate 0.01, momentum = 0.9) behaves similarly for both single and double hand gesture images.

In search of possibilities for further improvement, validation loss curves of SGD at different learning rates keeping the momentum constant and vice versa were studied. Validation loss curves of SGD with momentum 0.9 and different learning rates is shown in Fig. 8a and loss curves of SGD for different momentum at a learning rate of 0.01 is shown in Fig. 8b. Best loss curves were obtained for SGD with a learning rate 0.01 and momentum 0.9.

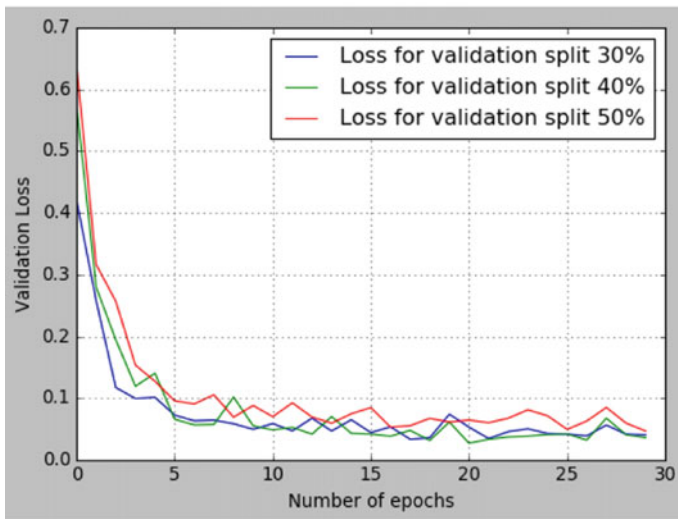
Adam with learning rate of 0.01 showed very poor performance. Performance of Adam at two different learning rates for double hand gesture images is shown in Fig. 9. Varying learning rates of RMSProp and AdaGrad from their initial learning rate of 0.001 also showed no further improvement.

5.6 Summary

After all the fine tuning of hyperparameters for the deep CNN architecture, validation and training loss and accuracy curves as shown in Figs. 10 and 11 are obtained. Flat asymptotic validation curves are obtained for the six-layered deep CNN for both single and double hand gesture images at validation accuracies of 95.5% and 97.8% respectively.

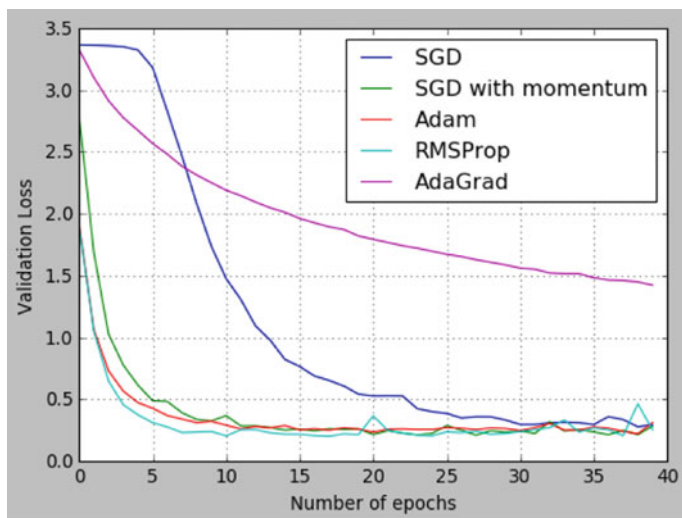


(a)

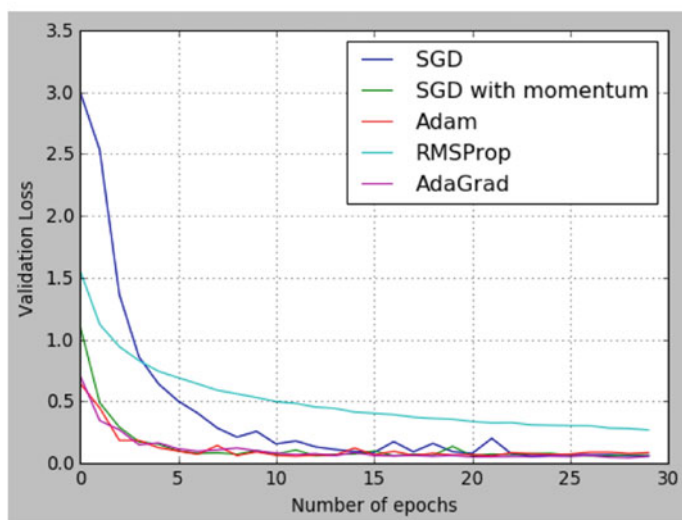


(b)

Fig. 6 Validation loss for **a** single hand gesture images **b** double hand gesture images for different validation split percentages

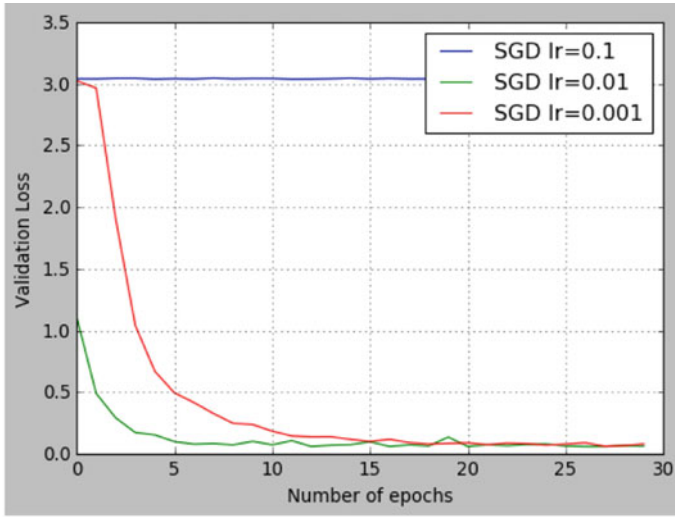


(a)

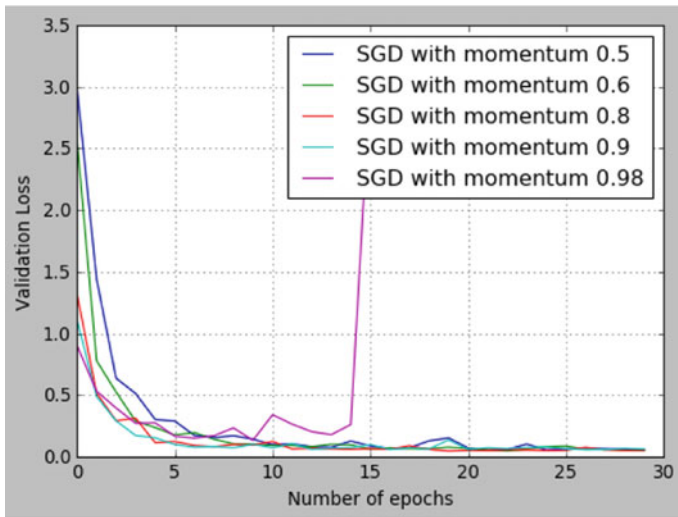


(b)

Fig. 7 Validation loss for **a** single hand gesture images **b** double hand gesture images for different optimisers



(a)



(b)

Fig. 8 Validation loss of SGD optimiser for double hand gesture images **a** for different learning rates (momentum 0.9) **b** for different momentum (learning rate is 0.001)

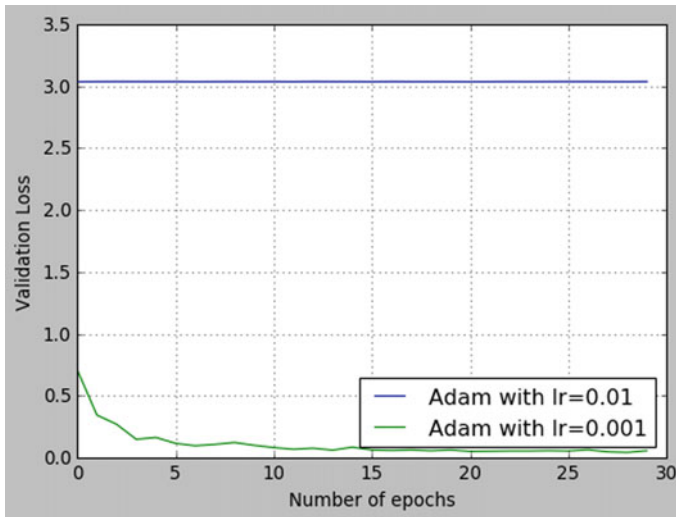
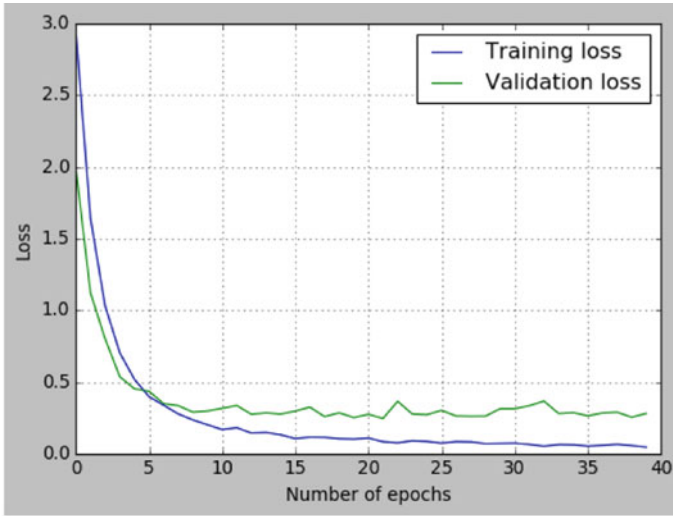


Fig. 9 Validation loss of Adam optimiser at different learning rates

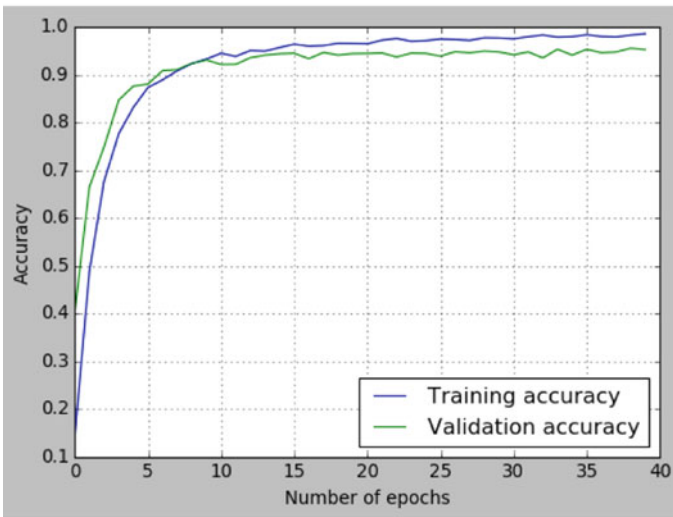
6 Conclusion

In this paper, attempt was made to systematically tune a CNN classifier that could accurately classify the images in Bharatanatyam mudra dataset. The created dataset is open and consists 15,396 distinct single hand gesture images of 29 classes and 13,035 distinct double hand gesture images of 21 classes. By using ablation studies, the architecture was decided as a six-layered deep CNN. Then systematic tuning of various parameters like batch size, dropout probability and learning rate was done. Role of validation split and the performance of different optimisers were also explored.

The optimum performance was with a batch size of 32. The dropout probability of 10% and validation split of 40% gave optimum performance. Adam optimiser of learning rate 0.001 and SGD with learning rate 0.01 and momentum 0.9 gave similar performance. But Adam was chosen due to its adaptive learning rates and robustness to hyperparameter variations. Using all these tuned parameters, flat asymptotic validation loss curves were obtained for CNN classifier using single hand gesture images and double hand gesture images at accuracies of 95.5% and 97.8%, respectively.

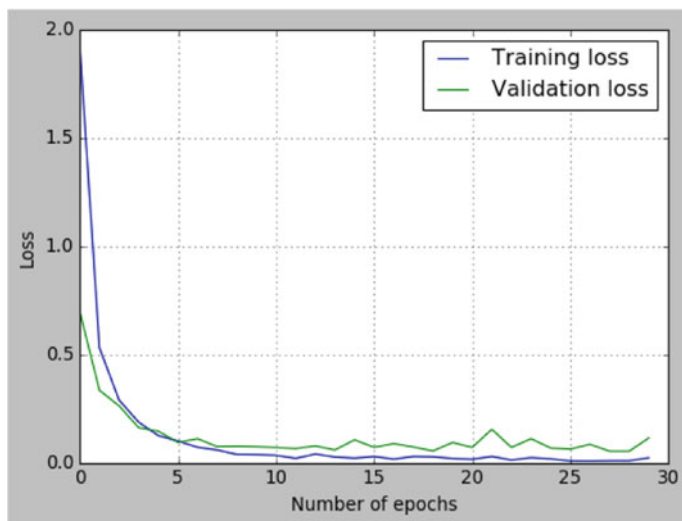


(a)

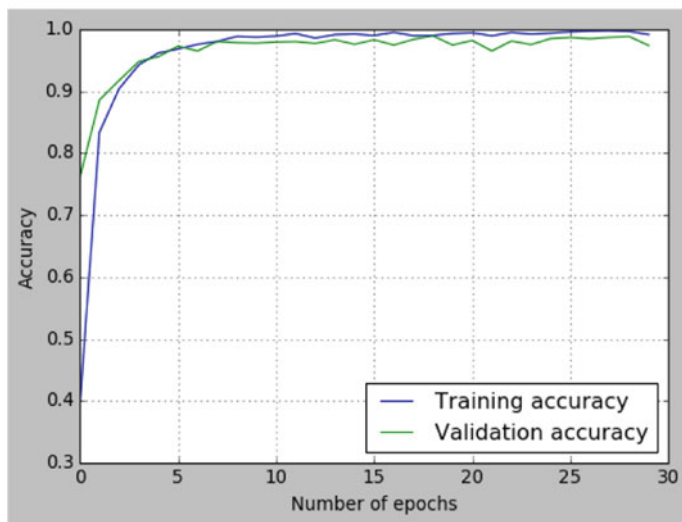


(b)

Fig. 10 **a** Training and validation loss **b** accuracy, for single hand gesture images using six layered deep CNN across 40 epochs



(a)



(b)

Fig. 11 a Training and validation loss b accuracy, for double hand gesture images using six layered deep CNN across 30 epochs










References

1. Amini A (2020) Introduction to deep learning. MIT 6:S191
2. Bengio Y (2012) Practical recommendations for gradient-based training of deep architectures. In: *Neural networks: tricks of the trade*. Springer, pp 437–478
3. Bhavanani A, Bhavanani D (2010) Bharatanatyam and yoga. *Yoga Mimamsa* 41:388–408
4. Bradski G (2000) The OpenCV library. Dr. Dobb's journal of software tools
5. Cai S, Shu Y, Chen G, Ooi BC, Wang W, Zhang M (2019) Effective and efficient dropout for deep convolutional neural networks. [arXiv:1904.03392](https://arxiv.org/abs/1904.03392)
6. Chandra B, Sharma RK (2016) Deep learning with adaptive learning rate using Laplacian score. *Expert Syst Appl* 63:1–7. <https://doi.org/10.1016/j.eswa.2016.05.022>
7. Coomaraswamy A, Duggirala GK (1917) *The mirror of gestures: being the abhinayadarpana of Nandikeswara* (English translation). Harvard University Press
8. Garbin C, Zhu X, Marques O (2020) Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimed Tools Appl* 79(19):1–39. <https://doi.org/10.1007/s11042-019-08453-9>
9. Ghosh M (1956) *Natyasastra* (English Translation):. Bibliotheca Indica, Manisha Granthalaya. <https://books.google.co.in/books?id=kXB0AAAAYAAJ>
10. Goodfellow I, Bengio Y, Courville A (2016) *Deep learning*. MIT Press. <http://www.deeplearningbook.org>
11. Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J et al (2018) Recent advances in convolutional neural networks. *Pattern Recognit* 77:354–377. <https://doi.org/10.1016/j.patcog.2017.10.013>
12. Hariharan D, Acharya T, Mitra S (2011) Recognizing hand gestures of a dancer. In: *International conference on pattern recognition and machine intelligence*. Springer, pp 186–192
13. He T, Li X (2019) Image quality recognition technology based on deep learning. *J Vis Commun Image Represent* 65:102654. <https://doi.org/10.1016/j.jvcir.2019.102654>
14. Kandel I, Castelli M (2020) The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express* 6(4):312–315. <https://doi.org/10.1016/j.icte.2020.04.010>
15. Ketkar S (2016) *The history of Indian art*. Jyotsna Prakashan, India
16. Know India: national portal of India: culture and heritage: performing arts. <https://knowindia.gov.in/culture-and-heritage/performing-arts.php>. Accessed 30 May 2021
17. Ko B, Kim HG, Oh KJ, Choi HJ (2017) Controlled dropout: a different approach to using dropout on deep neural network. In: *2017 IEEE international conference on big data and smart computing (BigComp)*. IEEE, pp 358–362. <https://doi.org/10.1109/BIGCOMP.2017.7881693>
18. Masters D, Luschi C (2018) Revisiting small batch training for deep neural networks. [arXiv:1804.07612](https://arxiv.org/abs/1804.07612)
19. Mohanty A, Vaishnavi P, Jana P, Majumdar A, Ahmed A, Goswami T, Sahay RR (2016) Nriyabodha: towards understanding Indian classical dance using a deep learning approach. *Signal Process Image Commun* 47:529–548. <https://doi.org/10.1016/j.image.2016.05.019>
20. Mozarkar S, Warnekar C (2013) Recognizing Bharatanatyam Mudra using principles of gesture recognition gesture recognition. *Int J Comput Sci Netw* 2(2):46–52
21. Patel D (2019) *The changing dynamics of a traditional art form case study of on Bharatanatyam Margam*. PhD thesis, Maharaja Sayajirao University of Baroda (India)
22. Radiuk PM (2017) Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Inf Technol Manag Sci* 20(1):20–24. <https://doi.org/10.1515/itms-2017-0003>
23. Ramachandrasekhar P (2013) *Abhinayadarpanam: Giri Trading Agency Private Limited*. Mumbai, Maharashtra
24. Saha S, Ghosh L, Konar A, Janarthanan R (2013) Fuzzy 1 membership function based hand gesture recognition for Bharatanatyam dance. In: *5th international conference on computational intelligence and communication networks (CICN)*. IEEE, pp 331–335

25. Sinha S, Singh T, Singh V, Verma A (2010) Epoch determination for neural network by self-organized map (som). *Comput Geosci* 14(1):199–206. <https://doi.org/10.1007/s10596-009-9143-0>
26. Smith LN (2018) A disciplined approach to neural network hyper-parameters: Part 1—learning rate, batch size, momentum, and weight decay. [arXiv:1803.09820](https://arxiv.org/abs/1803.09820)
27. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(1):1929–1958
28. Takáč M, Bijral A, Richtárik P, Srebro N (2013) Mini-batch primal and dual methods for svms. In: *International conference on machine learning*. PMLR, pp 1022–1030
29. Tieleman T, Hinton G (2012) Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw Mach Learn* 4(2):26–31
30. Wikipedia contributors: Bharatanatyam—Wikipedia, the free encyclopedia (2020). <https://en.wikipedia.org/w/index.php?title=Bharatanatyam&oldid=880467278>. Accessed 1 Mar 2020
31. Wilson DR, Martinez TR (2003) The general inefficiency of batch training for gradient descent learning. *Neural Netw* 16(10):1429–1451. [https://doi.org/10.1016/S0893-6080\(03\)00138-2](https://doi.org/10.1016/S0893-6080(03)00138-2)

Comparative Analysis of Neural Architecture Search Methods for Classification of Cultural Heritage Sites



Sunil V. Gurlahosur , S. M. Meena , Uday Kulkarni , Winston Dcosta ,
Vineet Lokur , Rohan V. Sirigeri , Sajal Porwal , S. P. Sammed ,
and Uma Mudenagudi 

1 Introduction

Machine learning (ML) has achieved considerable success in applications [1] like object detection, speech recognition, medical diagnosis, and weather forecasting. An ever-growing number of new disciplines like self-driving cars, recommendation systems, etc. also rely on ML algorithms for developing solutions. However, traditional approaches of ML are based on a trial-and-error process, and the efficiency of these ML-based solutions critically depends on data collection, preprocessing, feature engineering, and model optimization. Hence even the experts require substantial resources and time to create efficient ML models. In ML, specifically, the Deep Learning (DL) algorithms have become more efficient in image processing: object detection, image classification, hyperspectral imaging [5], video analytics because of their ability to operate directly on raw data and learn higher-level representation. However, DL models are highly complex in design and computationally expensive during training and testing. As the complexity of these algorithms is often beyond non-ML-experts, and the rapid growth of DL applications has created a demand for off-the-shelf methods that can be used without the essentials of expert knowledge. The Automated Machine Learning (AutoML) has emerged [14] to reduce the devel-

S. V. Gurlahosur (✉) · S. M. Meena · U. Kulkarni · W. Dcosta · V. Lokur · R. V. Sirigeri ·
S. Porwal · S. P. Sammed · U. Mudenagudi
KLE Technological University, Hubballi, India
e-mail: svgurlahosur@kletech.ac.in

S. M. Meena
e-mail: msm@kletech.ac.in

U. Kulkarni
e-mail: uday_kulkarni@kletech.ac.in

U. Mudenagudi
e-mail: uma@kletech.ac.in

opment costs, demand for ML engineers, enable domain experts to automatically build ML applications without much requirement for statistical and ML knowledge that targets progressive [20] automation of ML. Construction of AutoML pipeline for Deep Learning-based applications involves automated construction of the pipeline on the limited computational budget by automating the time-consuming, iterative tasks of learning model development [12]. There is growing interest in employing AutoML [30] techniques that automate architecture engineering methods to build models with reasonable accuracy for a specified task. Researchers [12] have proposed various Neural Architecture Search (NAS) approaches that automatically search for a suitable and optimized neural network architecture. The proposed NAS methods have generated CNN architecture for image classification task on datasets like CIFAR, MNIST, ImageNet [11] and have shown promising efficiency [31] compared to their handcrafted counterparts. Since NAS methods [29] employ various enhancements (hyperparameter optimization methods), NAS methods have become challenging for researchers to compare and reproduce the same results. Hence a promising NAS method on one data set may disappoint to produce the same efficient networks when an attempt is made to transfer it to other datasets. Additionally, engineers attempting to use NAS often find it challenging to understand the implications of advertised advances because of a deluge of research claims, inability to fairly compare methods, fragmented codebases, customized hyperparameters, and training techniques. NAS method efficiency should be uniform across datasets; thus, a custom image data set based on Indian Heritage sites using crowdsourcing framework is proposed to study whether the NAS methods are generalizable across tasks rather than dataset-specific (standard datasets). The contributions of the proposed work are:

1. Custom image data set comprising of Cultural Heritages sites in India.
2. Comparative performance analysis of the NAS methods on the custom dataset.
3. Propose a neural architecture model for the classification of Cultural Heritage sites with 88.625% accuracy.

The paper is organized into the following sections. Section 2, focuses on the components present in NAS. In Sect. 3, we discuss about the methodology followed for comparative analysis of NAS methods. In Sect. 4, results are discussed for image classification task on heritage sites using NAS methods.

2 Background Study

NAS is an algorithmic-based approach [38] for automating the process of building a neural network that aims to find the optimal network that outperforms hand-designed model in terms of accuracy, model size, latency, and power consumption [16]. NAS strives to learn a network topology that achieves the best performance on a particular task, and it goes with the principle “Better the design, Better the performance.” NAS consists of three components as shown in Fig. 1 (i) search space of neural

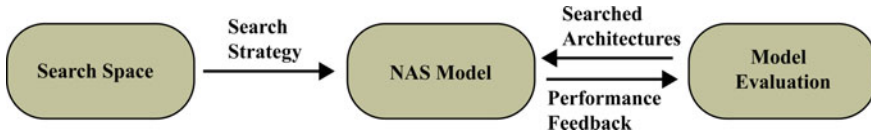


Fig. 1 Components of neural architecture search method

architecture, (ii) search strategy methods, and (iii) model evaluation methods. Search space defines a set of operations (e.g. convolution, fully-connected, pooling) and how these operations can be used to generate neural network architectures. Search strategy algorithm samples a population/child network architecture from the search space. Since search space can be extremely large, evaluation strategy estimates the sampled architecture performance to obtain the feedback for optimizing the search algorithm.

Zoph and Le [38] were one of the first to propose NAS, where a recurrent network is trained by reinforcement learning to search for the best-performing architecture automatically. Since this method successfully discovered neural networks achieving comparable results for image recognition tasks to human-designed models, there has been an explosion of research interest in AutoML [32], especially in the area of NAS. The recent advancements in NAS methods have made it possible to build problem-specific networks, which results in neural networks that are efficient in terms of latency, accuracy, and power consumption than their handcrafted counterparts.

2.1 Search Space

The NAS search space defines all possible architectures with the primary network operations like convolution, fully connected, max and avg pooling to build valid neural network architecture. The initial proposed NAS method [38] consists of sequential layer-wise operations where every operation has an association with different layer-specific parameters. This method consumes substantial computational resources to cover the search space [3] because of its vast representational power. Hence, cell-based representation [21] proposed for search space to address the issues of huge search space and the model size. The architecture obtained with a cell-based search method containing blocks (group of operations). These blocks, commonly referred to as cells, are classified into the normal cell if both input and output feature maps have the same size or reduction cell, if output feature maps are reduced by half. The number of cell repeats (motifs) can be altered to develop network architecture for the transfer of the solutions from one task to the other. The motifs generated by search space can be organized in a hierarchical [37] structure. A small set of operations (cell), including single operations of convolution and pooling, can be chosen in the earlier stage. Further, the small subgraphs (motifs) representing a set of operations are used in the later stages of the search space. The operations present in motifs of cell-

based and hierarchical structure search space cannot be modified during search space design. The memory-bank type of representation [6] for feed-forward networks is proposed to overcome the issue. In this method, the neural network is visualized as a system with multiple memory blocks, which can be read and modified at every layer. This method also accelerated the architecture selection process defining network connectivity patterns thus generating network weights for highly variable network architectures.

2.2 Search Strategy

Once search space is defined, search strategy samples promising architectures for estimating performance and thus avoiding the overhead of testing bad architectures. Several approaches are used for finding efficient architectures, and the most renowned among them are Reinforcement learning, Gradient-based optimization, and Evolutionary algorithms. The initially proposed NAS method [38] used Reinforcement learning, which focuses on maximizing the long-term rewards. It uses the Markov decision process to map a solution using a numerical approach and Recurrent Neural Network (RNN) to maximize the accuracy of training the generated/sampled architectures on a given data set. The RNN controller predicts the hyperparameter for one layer and repeats the same for several layers and terminates after the number of layers exceeds a certain (threshold) value. The controller in RNN further maximizes the expected reward for optimal architecture over the parameter θ_c represented as $J(\theta_c)$.

$$J(\theta_c) = E_{p(a_{1:T};\theta_c)}[R] \quad (1)$$

The tokens predicted by the controller can be viewed as a list of actions $a_{1:T}$ to design an architecture for a child network. The accuracy R achieved is used as a reward signal and to train the controller. The Policy Gradient Method [34] is used to iteratively update parameters θ_c since reward function R is non-differentiable.

$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^T E_{p(a_{1:T};\theta_c)}[\nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R] \quad (2)$$

Though the Reinforcement learning-based search algorithm does not require a large labeled dataset, it consumes a lot of time searching the architectures since the controller predicts hyper-parameters one at a time conditioned on previous predictions. To overcome the non-differentiable reward function and large search time, a gradient-based search algorithm [22] is proposed, which achieves better gradients compared to discrete search space. This method reduces the high search cost of Reinforcement learning by optimizing the architecture parameters. It uses a continuous relaxation method for the search space, which helps in the reduction of the search

cost of finding an architecture. The main drawback of this method is that it typically selects many skip connections that dominate over other types of operations, leading to degradation in performance when the number of search epochs increases [18].

The searched neural networks do not achieve multi-objective, like accuracy, model size, latency, FLOPSs, and power consumed by the hardware resources. Authors [33] proposed a search method for designing to solve multi-objective problems. The Neuro Evolution of Augmenting Topologies (NEAT) evolves connecting weights by gradient-based algorithms and neural network topologies with the genetic algorithms simultaneously. The evolutionary algorithm operates on the population that includes architectures produced using mutation and crossover techniques to search for the optimal architecture with multiple objectives. It has global search capabilities for searching of architectures. This method flaw is that appropriate genetic operations and population size majorly influence the architecture's accuracy. In addition, there are other search strategies available such as random and grid search, which are computationally expensive and inefficient. In random search, since the parameters are selected randomly and no intelligence is used to sample the architectures, there is a higher chance of getting lower accuracy networks. The Grid search fails when the search space dimensions are large and the number of hyperparameters increases exponentially.

2.3 Evaluation Strategy

Since search space can be extremely large, an evaluation strategy should be defined to estimate the sampled architecture performance to obtain the feedback for optimizing the search algorithm. When a reference data set is trained over a predefined number of epochs followed by testing of the model, typically, the accuracy of a model architecture plays an important role in estimating the performance of the child model. Apart from the model accuracy as a performance measure, few more parameters like model size, FLOPS, latency, etc., are considered as the present applications of AI/DL algorithms are ported onto embedded platforms. The evaluation process is computationally expensive since it should be carried out for all the sampled architecture. Hence the evaluation method should be chosen carefully and many promising evaluation methods are proposed for saving computation time. One of the earlier proposed NAS methods [38] followed the traditional approach of training the network from scratch to evaluate the searched models. In this method, all the child models sampled by search algorithms are trained until they reach convergence. Later the models are evaluated based upon the validation accuracy. Though it is a suitable evaluation method, it is computationally costly since the training of each network is carried out separately. Hence, Pham et al. [28] proposed a parameter sharing method to reduce this extensive training time by creating a dependency between parameters and reusing the weights instead of training each child model independently. In this method, all the child models are represented as subgraphs of a larger supergraph and they are sharing the parameters/weights of this supergraph.

Though the parameter sharing method helps to share the weights among all child models, training the models against a complete data set for a fixed number of epochs until convergence is inevitable and leads to an ample training time. Hence, a proxy method [8] is proposed for the evaluation of model where each child network uses a proxy task performance as the performance estimator. Researchers have proposed several proxy approaches, which are typically faster to calculate by training the model on the smaller data set or training the model for few epochs. It helps get a brief visualization of the model accuracy to decide training to a larger data set. Zoph et al. [39] proposed a method to first train and evaluate a network model which is down-scaled in the search stage and later change the filters and cell repeats in the model to further improve the model performance. Baker et al. [4] proposed a method to predict the learning curve. The validation accuracy is predicted by considering the hyperparameter's accuracy per epoch and architecture parameters in this approach. Instead of training the child to obtain the weights, David et al. [13] proposed a method in which model weights predicted are directly related to the network architecture parameters. Hypernets are employed for generating and validating model weights, thus overcoming the effort of training every child.

3 Methodology

Current NAS methods [27] use various enhancements (hyperparameter optimization methods) [31], custom training techniques to improve the accuracy. Hence a promising NAS method on one data set may not produce the same efficient network when an attempt is made to transfer [27] it to other datasets. NAS method efficiency should be uniform across datasets; thus, we propose a custom image data set based on Indian Heritage sites to study whether the NAS methods are generalizable across tasks rather than dataset-specific (standard datasets). We used a web-based crowdsourced framework [15, 17] for the data collection process using intelligent workflow. The framework allows users to contribute data for the existing list of heritage sites or add a new heritage site/label and contribute data for the same site. The roles of the users are assigned based on privileges, contributions, and functionalities defined in the workflow. Images uploaded by crowd/contributors are stored in the data repository as shown in Fig. 2 for further validation by the validators. Validation is performed to verify whether uploaded images in the data repository belong to the heritage site selected by the contributor/crowd. After the successful validation, images will be inspected by the analyst and will be considered to store in the data repository for pre-processing. With a collective effort of both crowdsource platform and visiting the places manually, we collected 4,68,651 images for 110 heritage sites [26]. In the NAS methods, search algorithm needs to explore the search space to sample possible architectures. The evaluation strategy needs to evaluate the sampled architecture to optimize the search algorithm and help in designing the optimal Neural architecture for the given task. Various studies have shown that small datasets like CIFAR, MNIST take less time to search the architectures [18] as the search space comprises fewer

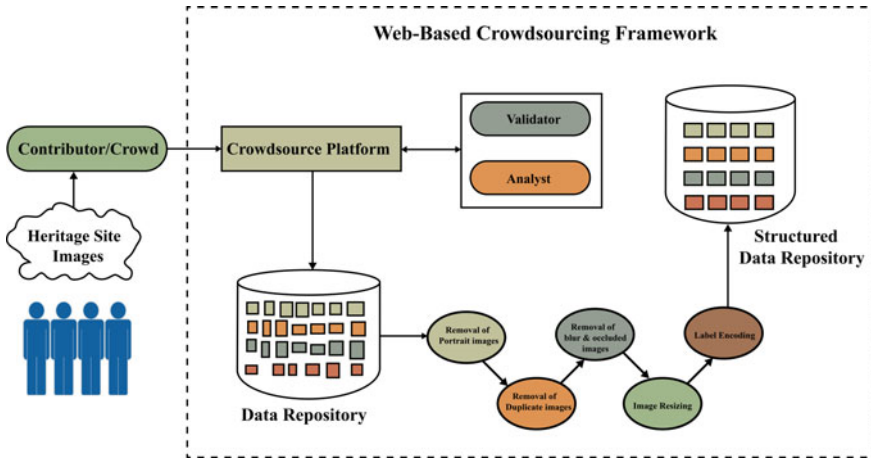


Fig. 2 Web-based crowdsourced framework for collection cultural heritage sites images

operations. In contrast, it consumes more time for larger datasets like ImageNet due to the larger search space.

Since the number of classes, quality, shape of images, and data set size influence the NAS algorithm search time and efficiency, we decided to build the custom data set of 40 different heritage sites with each image of size 1920*1080 pixels. The size of portrait and landscape images in the data set suggested, only landscape images were suitable for resize operation since portrait images were getting stretched and losing information both qualitatively and quantitatively. Hence, we handpicked heritage sites/labels which contain the majority of landscape images compared to portrait images from the data set repository of 110 classes. In data preprocessing shown in Fig. 2, we removed the remaining portrait images and redundant images with a similarity score of 85%. Quantitative analysis is performed on the data set to remove the blur and occluded images. Since images are of different sizes, images are resized to 1920*1080 pixels for uniform size and reduce the complexity of the model search. After the resize operation, the data set resulted in a size of 17.4 GB with 20,000 labeled images. The images are divided into 16,000 training images and 4000 test images (see Fig. 3).

The NAS algorithm involves various techniques in each of its components (search space, search strategy and evaluation strategy) we considered ENAS [28], DARTS [22] and NSGA-Net [24] for image classification task, since they cover most of promising techniques in each of its components. To search the optimal architecture as shown in Fig. 4, the first phase is search space to represent a pool of all possible operations required for building a neural network. Search strategy represents a method to build an architecture through the set of operations available in the search space. In the evaluation phase, generated architectures are trained to find architectures that achieve high predictive performance on unseen data (test data).



Fig. 3 Sample images of cultural heritage sites in structured data repository

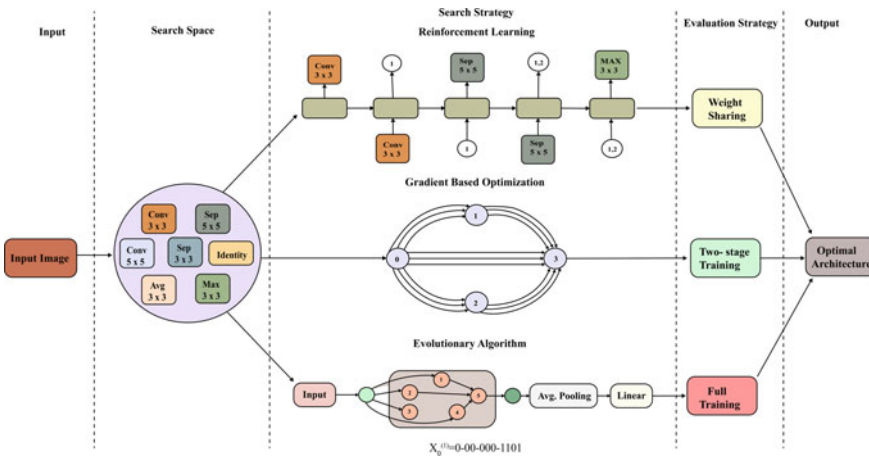


Fig. 4 Control flow in ENAS, DARTS, and NSGA-Net methods to search optimal neural network architecture

ENAS method follows both sequential and cell-based methods for its search space, Reinforcement Learning (RL) method for search algorithm, and shared weights method for model evaluation. The Reinforcement Learning based meta-controller agent discovers neural network architectures by searching for an optimal subgraph within a large computational graph. It predicts network transformation [7] actions by learning from the current network architecture. The meta-controller is implemented as a recurrent bi-directional network to handle a variable-length architecture configuration to grow the network for depth and width. Here network can grow incrementally in both directions, (i) layer will be replaced by a wider layer that will have more filters in convolution layer or more number of fully-connected layers by having the functionalities preserved, and (ii) new layer is inserted and it is initialized as adding an identity mapping between two layers to preserve the functionality. The weights

of previously validated networks can be reused for further exploration since the network is growing incrementally. With inherited weights, newly constructed networks only need light-weighted training. Training will be alternated between the shared model weights ω and the controller θ . The parameters of the controller LSTM θ are trained with the Monte-Carlo policy gradient method, where the reward $R(m, \omega)$ is computed on the validation set. The shared parameters of the child models ω are trained with standard supervised learning loss function using Eq. 3 for determining the model performance.

$$\nabla_{\omega} E_{m \sim \pi(m; \theta)} [L(m; \omega)] \approx \frac{1}{M} \sum_{i=1}^M \nabla_{\omega} L(m_i, \omega), \quad (3)$$

The controller is trained with a policy gradient method to select a subgraph to maximize the expected reward on a validation set. Meanwhile, the model corresponding to the selected subgraph is trained to minimize a canonical cross-entropy loss to get the parameters of the top-performing model. This method is encouraged by multi-task learning and the transfer learning parameters trained for different tasks and transferred to other tasks. It follows the three steps, namely design of networks for Recurrent cells, macro search space over entire convolutional networks and micro search space for convolutional cells. To create a recurrent cell, the controller RNN samples N blocks of decisions. The following Algorithm 1 illustrates the Generation of RNN cells in the ENAS via a recurrent cell with $N = 4$ computational nodes.

Algorithm 1 ENAS mechanism through a recurrent cell

- 1: The node receives the output from the previous step, input from the current step and generates an activation function
 - 2: $N[1] = \tanh(pre * W[I] + input * weight_{input})$
 - 3: A new activation function is generated,
 - 4: $N[2] = \text{relu}(N[1] * w[2][1])$
 - 5: A new activation function is generated,
 - 6: $N[3] = \text{relu}(N[2] * w[3][2])$
 - 7: A new activation function is generated,
 - 8: $N[4] = \tanh(N[1] * w[4][1])$
 - 9: Algorithm takes average of all the nodes which are not part of input to any other nodes
 - 10: Hence, the final output = $\frac{(N[3] + N[4])}{2}$
-

The macro-convnet design involves deciding computation operation (convolution, max pooling or average pooling) for design of layer in network. In total six operations [9] are available for the controller to decide: convolutional operation with filter size 3×3 and 5×5 , depth wise-separable convolutional operation with filter size 3×3 and 5×5 , and max pooling and average pooling of kernel size 3×3 . To generate the micro-convnet, multiple conv cells are replicated to make the final network. Here 5 operations are available like identity, max pooling, average pooling, convolutional, and separable during the replication process.

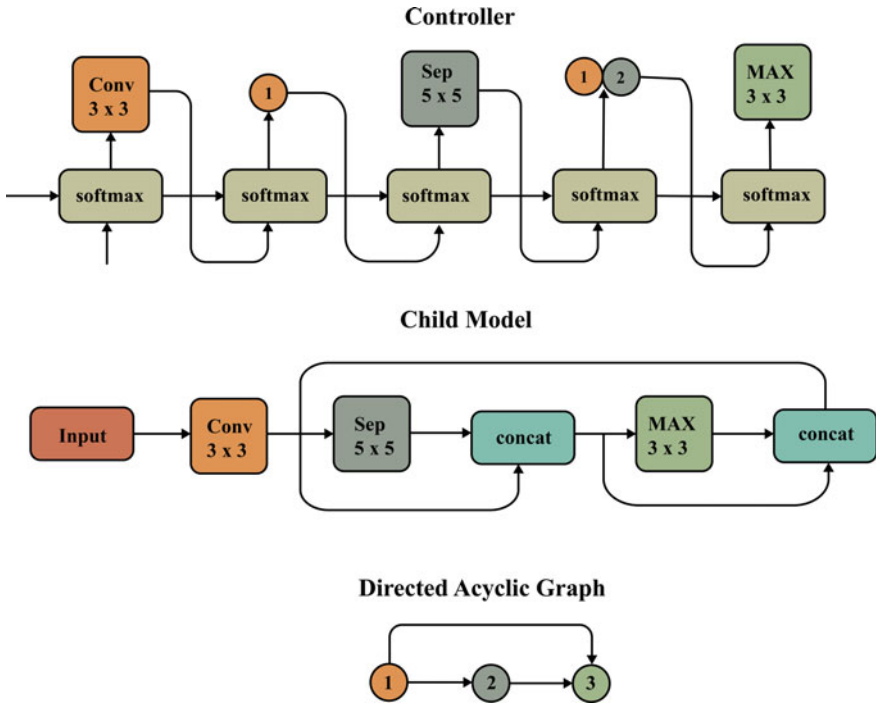


Fig. 5 RNN controller generating child model and layers in generated neural network architecture

Fig. 5 shows the working of ENAS with the Reinforcement Learning algorithm. The controller generates a set of architectures through the search space, and the child model represents a sampled architecture. The Directed acyclic graph with 3 nodes where the nodes denote the layers in a convolutional neural network and the arrow denotes the path of computations. The generated architectures are evaluated during the evaluation stage. The accuracy obtained from the architectures is treated as a reward to update the controller’s parameters and generates a new set of architectures with better efficiency.

DARTS uses a NASNet search space, gradient-based optimization algorithm to search the architecture and train from scratch approach to evaluate the searched architectures. In the DARTS method, the search space is represented in the form of a Directed acyclic graph, which comprises N nodes in the ordered sequence where each node x^i represents a feature map in Convolution Neural Network [29]. The directed edges connect the nodes with weights where each edge represents [22] operation $o^{(i,j)}$. The outputs from two previous layers of the convolution cells form the input to the current cell. Reduction operations are performed on every intermediate node to obtain output from the cell. The output of each cell is calculated by performing

the reduction operation to intermediate nodes. All the intermediate nodes will be computed with Eq. 4 using its predecessor nodes, as

$$x^{(j)} = \sum_{i < j} o^{(i,j)}(x^{(i)}) \quad (4)$$

A unique zero operator is included in the directed edges for indicating a lack of connectivity between two nodes. Now the task remaining is to learn the operations on the edges of the cell. Initially, the edges are not associated with each other as shown in Fig. 5. But later, each edge is associated with some mixed operation like convolution, max pooling, etc. The search space is made continuous by relaxing the categorical choice of each operation to a softmax for overall possible operations. As shown in Eq. 5, let O be a set of the candidate operations (like convolution, max pooling, zero) where each of the operations represents function in which will be applied for x^i .

$$o^{-(i,j)}(x) = \sum_{o \in O} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in O} \exp(\alpha_{o'}^{(i,j)})} o(x) \quad (5)$$

where operation mixing weights for nodes in the pair which are parameterized by a vector $\alpha^{(i,j)}$ with the dimension as $|O|$. After the search, the most likely operation $o^{(i,j)} = \operatorname{argmax}_{(o \in O)} \alpha_o^{(i,j)}$, where α is referred to as architecture, is replaced with each of the mixed operations $o^{-(i,j)}$ to obtain a discrete search space. Then the bilevel optimization [2] comes into consideration, where the lower-level variable is for the minimization [10] of the training loss L_{train} as shown in Eq. 7 for the associated weight and the upper-level variable is for the minimization [22] of the validation loss L_{val} as shown in Eq. 6 for the architecture search.

$$\alpha_{\min} L_{val}(w^*(\alpha), \alpha) \quad (6)$$

$$\text{s.t. } w^*(\alpha) = \operatorname{argmin}_w L_{train}(w, \alpha) \quad (7)$$

Fig. 6 illustrates DARTS algorithm with directed acyclic graph consisting of four nodes where each node represents a feature map in a convolutional network. Each operation is described in the form of blocks and is associated with some weights. The weights associated with the operations are optimized after each epoch with the help of a bilevel optimization problem. At the end of all the epochs, only the most likely or the optimized operations (dark lined blocks) are selected to determine the final architecture.

NSGA-Net uses a population-based Evolutionary algorithm with crossover and mutation to search the architectures, and it also considers multiple objectives. The train from scratch approach followed in DARTS is employed to evaluate the searched architectures. In the NSGA-Net method, the first stage is encoding [23], where the

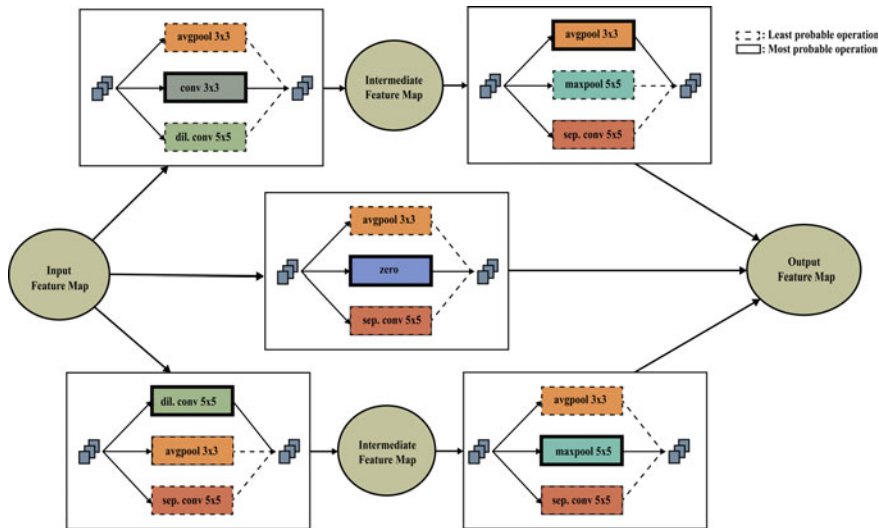


Fig. 6 Continuous relaxation of DARTS search space describing feature map of Directed Acyclic Graph in a convolutional neural network

architectures available from search space are converted to the binary-encoded string. Then Phenotype of architecture is built by the use of Genotype (Genotype is a series of rules for constructing neural networks and Phenotype is a neural network encoded by the Genotype). Later the complete information of a network is encoded by a gene and the network is represented in terms of stages. If the network consists of S stages where the s th stage contains K_s nodes, which is denoted by v_{s,k_s} , $k_s = 1, 2, 3, \dots, K_s$. Each node that is ordered in each stage corresponds to convolution and batch normalization and ReLU, after summing up element-wise of all of its input nodes. The fully connected part of the network is not encoded. In every stage, the usage of the bits [24] for encoding the inter-node connections are $1 + 2 + \dots + (K_s - 1) = \frac{1}{2} K_s(K_s - 1)$.

The connections are represented as $(v_{s,i}, v_{s,j})$ where i, j is bit representation. Fig. 7 illustrates the three-stage network, and the binary string represents the encoded version of each stage. The nodes present outside each stage represent the input and output nodes. A pooling layer is added after every stage. Each stage consists of several nodes, which represent the operations in a convolutional neural network. If there is an edge between the nodes, it is represented through 1 and if there is no edge, it is represented through 0. For obtaining efficient neural network architectures, Genetic operations like selection, mutation, and crossover are utilized.

Initialization of search algorithm is done using N number of randomized individuals. For all the T generations initialized, genetic [23] operations were applied. A process named Russian roulette [25] is performed, which helps in determining [36] the survival of the individuals. Every individual of the next generation, i.e.,

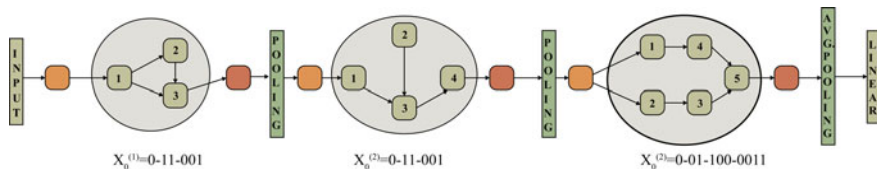


Fig. 7 A three-stage network and its encoded binary string

$M_{t,n}$ is independently determined by a non-uniform way of sampling over the set $\{M_{t-1,n}\}_{n=1}^N$. The probability [35] obtained by the sampling $M_{t-1,n}$ is proportional to $r_{t-1,n} - r_{t-1,0}$ where $r_{t-1,n}$ is the recognition rate, which is defined by the fitness function assigned to all the individuals and $r_{t-1,0} = \min_{n=1}^N \{r_{t-1,n}\}$ is a value obtained by minimum fitness function from the previous generation. The individual having the largest probability is selected by this process and the worst one is being eliminated. The method of crossover [23] involves simultaneously changing two individuals. It is better to form a stage that is a basic unit for crossover instead of individual bit, which is inflected by the need for retaining the local structures within each of the edges. The process of mutation involves flipping of each bit of the individual independently with the probability [35] of q_M . Generally, since the configured probability is very small (0.05), the mutation will likely not change an individual so much. This small probability [19] will also help retain the good properties of the individual and give the chance to try new possibilities. After the final set of architectures is obtained, it tries to find the architectures with better trade-off value by comparing the accuracy against the other parameters like model size, latency, and power consumption.

4 Results and Discussion

To generate a neural architecture model for the classification of Cultural Heritage sites, all the considered NAS algorithms are supplied with the same data set and configured to train on NVIDIA DGX-1, Tesla V100 GPUs. The ENAS [28] method is applied to search normal and reduction convolutional cells for micro search space. The algorithm is configured to search architecture for 150 epochs, batch size of 64 with Adam optimizer and a learning rate of 0.00035 [28]. The method took 48 GPU hours to complete the architecture search. During each epoch, the ENAS [28] method did not optimize the produced architectures resulting in poor training accuracy of 45.02% and test accuracy of 32.835%. After every epoch, based upon the accuracy of the produced architectures, the parameter of the RNN controller started getting updated instead of searching architectures that have better accuracy compared to present searched architectures. Hence, because of this issue, the accuracy after each epoch remained constant, and there was barely an increase in the accuracy at the end of 150 epochs. The DARTS [22] method is configured to search architecture

for 150 epochs, batch size of 64. The search algorithm took approximately 26h to search for the architectures. The evaluation method is configured to train the searched architectures for 150 epochs with a batch size of 64 on two GPUs. The DARTS method resulted in a training accuracy of 98.89% and a test accuracy of 88.625%. In the DARTS method, instead of producing new architectures after each epoch, by searching in a continuous space, DARTS can optimize the architectures, which resulted in a better accuracy and reduced training time.

The NSGA-Net [24] method is configured with a number of phases $np = 3$ and the number of nodes in each phase $n0 = 6$. The number of architectures to be produced in each generation is set to 40. The algorithm is trained to search architecture for 150 epochs with batch size 64. The algorithm took approximately 5h to produce one single architecture, so for 40 architectures, it took roughly a week (197h). The searched model showed training accuracy of 91.625% and test accuracy of 69.92%. Due to the inclusion of genetic algorithms, the search space was widened since it resulted in many diverse architectures because the architectures were encoded to binary strings; even a flip in the single bit resulted in a new architecture. Hence, this method gave a better accuracy when compared to ENAS, since ENAS produces a new set of architectures after every epoch.

The DARTS method that uses gradient-based optimization produced the architectures with the highest accuracy as shown in Table 1 with less time. ENAS and NSGA-Net did not give a better accuracy compared to DARTS and also took more time in searching the architectures. DARTS and NSGA-Net methods optimized the architecture after every epoch as shown in Fig. 8, whereas ENAS produces a new set of architectures after every epoch. Hence, the DARTS method is well suited if model with highest accuracy is the only criterion. If multi-objectives are involved in addition to accuracy such as model size, latency and power consumption then the NSGA-Net (evolutionary) method performs very well.

Table 1 Performance analysis of ENAS, DARTS, and NSGA-Net methods for classification of cultural heritage site images

NAS method	Search method	Test accuracy (%)	Params (in millions)	Search cost (GPU hours)
ENAS	Reinforcement learning	32.83	4.25	48
DARTS	Gradient based optimization	88.625	3.36	46
NSGA-Net	Evolutionary algorithm	69.92	4.5144	192

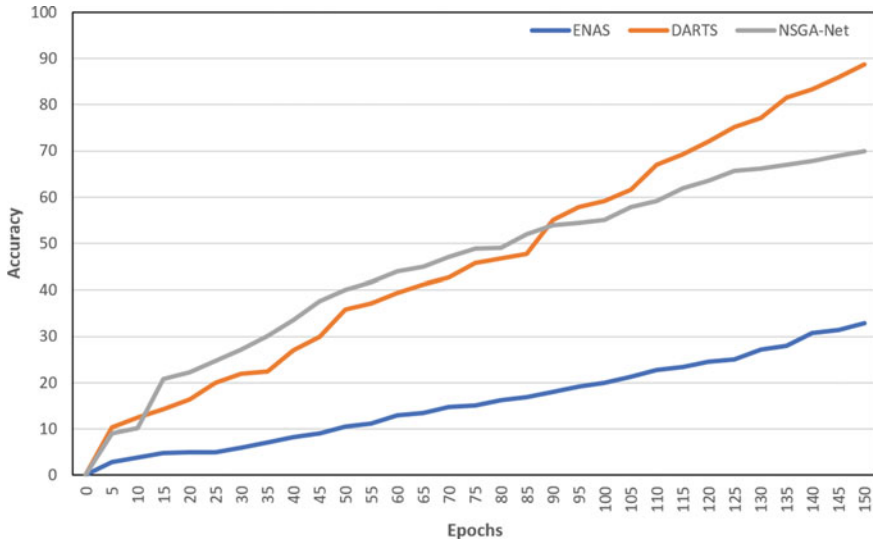


Fig. 8 ENAS, DARTS, and NSGA-Net methods comparison for accuracy versus epochs during architecture search

5 Conclusion

The engineers require substantial resources for the hand design of neural networks with reasonable performance, and it is always time-consuming, computationally expensive and error-prone method. NAS automates this process intending to find the optimal network that outperforms the hand-designed model. It is challenging to infer why NAS methods work well on standard datasets and perform poorly when attempt is made to transfer the same NAS method to real-time/custom datasets. In this work, we proposed a custom image data set based on Indian heritage sites using crowdsourced framework and performed a comparative performance analysis of three NAS algorithms, ENAS, DARTS, and NSGA-Net, for the image classification task. The DARTS method showed the highest accuracy of 88.625, and NSGA-Net is well suited if multi-objectives are considered for generating the neural architecture in addition to accuracy.

References

1. What is automated machine learning (automl)? (Nov 2020). <https://docs.microsoft.com/en-us/azure/machine-learning/concept-automated-ml>
2. Anandalingam G, Friesz TL (1992) Hierarchical optimization: an introduction. *Ann Op Res* 34(1):1–11

3. Baker B, Gupta O, Naik N, Raskar R (2016) Designing neural network architectures using reinforcement learning. [arXiv:1611.02167](https://arxiv.org/abs/1611.02167)
4. Baker B, Gupta O, Raskar R, Naik N (2017) Accelerating neural architecture search using performance prediction. [arXiv:1705.10823](https://arxiv.org/abs/1705.10823)
5. Bidari I, Chickerur S, Ranmale H, Talawar S, Ramadurg H, Talikoti R (2020) Hyperspectral imagery classification using deep learning. In: 2020 fourth world conference on smart trends in systems, security and sustainability (WorldS4). Ieee, pp 672–676
6. Brock A, Lim T, Ritchie JM, Weston N (2017) Smash: one-shot model architecture search through hypernetworks. [arXiv:1708.05344](https://arxiv.org/abs/1708.05344)
7. Cai H, Chen T, Zhang W, Yu Y, Wang J (2018) Efficient architecture search by network transformation. In: Proceedings of the AAAI conference on artificial intelligence, vol 32
8. Cai H, Zhu L, Han S (2018) Proxylessnas: direct neural architecture search on target task and hardware. [arXiv:1812.00332](https://arxiv.org/abs/1812.00332)
9. Chollet F (2017) Xception: deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1251–1258
10. Colson B, Marcotte P, Savard G (2007) An overview of bilevel optimization. *Ann Op Res* 153(1):235–256
11. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee, pp 248–255
12. Elsken T, Metzen JH, Hutter F (2019) Neural architecture search: a survey. *J Mach Learn Res* 20(1):1997–2017
13. Ha D, Dai A, Le QV (2016) Hypernetworks. [arXiv:1609.09106](https://arxiv.org/abs/1609.09106)
14. Hutter PDF (Feb 2021) Automl. <https://www.automl.org/automl/>
15. Kulkarni U, Meena SM, Joshua P, Rodrigues K, Gurlahosur SV (2020) Integrated crowdsourcing framework using deep learning for digitalization of Indian heritage infrastructure. In: 2020 IEEE sixth international conference on multimedia big data (BigMM). Ieee, pp 200–208
16. Kulkarni U, Meena S, Gurlahosur SV, Bhogar G (2021) Quantization friendly mobilenet (qf-mobilenet) architecture for vision based applications on embedded platforms. *Neural Netw* 136:28–39
17. Kulkarni U, Meena S, Gurlahosur SV, Mudengudi U (2019) Classification of cultural heritage sites using transfer learning. In: 2019 IEEE fifth international conference on multimedia big data (BigMM). IEEE, pp 391–397
18. Liang H, Zhang S, Sun J, He X, Huang W, Zhuang K, Li Z (2019) Darts+: improved differentiable architecture search with early stopping. [arXiv:1909.06035](https://arxiv.org/abs/1909.06035)
19. Liu C, Dollár P, He K, Girshick R, Yuille A, Xie S (2020) Are labels necessary for neural architecture search? In: European conference on computer vision. Springer, pp 798–813
20. Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li LJ, Fei-Fei L, Yuille A, Huang J, Murphy K (2018) Progressive neural architecture search. In: Proceedings of the European conference on computer vision (ECCV), pp 19–34
21. Liu H, Simonyan K, Vinyals O, Fernando C, Kavukcuoglu K (2017) Hierarchical representations for efficient architecture search. [arXiv:1711.00436](https://arxiv.org/abs/1711.00436)
22. Liu H, Simonyan K, Yang Y (2018) Darts: differentiable architecture search. [arXiv:1806.09055](https://arxiv.org/abs/1806.09055)
23. Lu Z, Deb K, Goodman E, Banzhaf W, Boddeti VN (2020) Nsganetv2: evolutionary multi-objective surrogate-assisted neural architecture search. In: European conference on computer vision. Springer, pp 35–51
24. Lu Z, Whalen I, Boddeti V, Dhebar Y, Deb K, Goodman E, Banzhaf W (2019) Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the genetic and evolutionary computation conference, pp 419–427
25. Lux I, Koblinger L (1991) Monte Carlo particle transport methods: neutron and photon calculations (boca raton, fl: Chemical rubber company)
26. Meena SM, Abhishek NK, Ravikumar A, Kulkarni U, Gurlahosur SV, Uma M (2021) Crowd source framework for Indian digital heritage space. In: Data analytics for cultural heritage: current trends and concepts, p 123

27. Panda R, Merler M, Jaiswal M, Wu H, Ramakrishnan K, Finkler U, Chen CF, Cho M, Kung D, Feris R et al (2020) Nastransfer: analyzing architecture transferability in large scale neural architecture search. [arXiv:2006.13314](https://arxiv.org/abs/2006.13314)
28. Pham H, Guan M, Zoph B, Le Q, Dean J (2018) Efficient neural architecture search via parameters sharing. In: International conference on machine learning. PMLR, pp 4095–4104
29. Real E, Aggarwal A, Huang Y, Le QV (2019) Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence, vol 33, pp 4780–4789
30. Real E, Liang C, So D, Le Q (2020) Automl-zero: evolving machine learning algorithms from scratch. In: International conference on machine learning. PMLR, pp 8007–8019
31. Ren P, Xiao Y, Chang X, Huang PY, Li Z, Chen X, Wang X (2020) A comprehensive survey of neural architecture search: challenges and solutions. [arXiv:2006.02903](https://arxiv.org/abs/2006.02903)
32. Stanley KO, Clune J, Lehman J, Miikkulainen R (2019) Designing neural networks through neuroevolution. *Nature Mach Intell* 1(1):24–35
33. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. *Evolut Comput* 10(2):99–127
34. Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach Learn* 8(3):229–256
35. Xie L, Yuille A (2017) Genetic cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1379–1388
36. Xu K, Srivastava A, Sutton C (2019) Variational Russian roulette for deep Bayesian nonparametrics. In: International conference on machine learning. PMLR , pp 6963–6972
37. Yu K, Sciuto C, Jaggi M, Musat C, Salzmann M (2019) Evaluating the search phase of neural architecture search. [arXiv:1902.08142](https://arxiv.org/abs/1902.08142)
38. Zoph B, Le QV (2016) Neural architecture search with reinforcement learning. [arXiv:1611.01578](https://arxiv.org/abs/1611.01578)
39. Zoph B, Vasudevan V, Shlens J, Le QV (2018) Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8697–8710

Heritage Representation of Kashi Vishweshwar Temple at Kalabgoor, Telangana with Augmented Reality Application Using Photogrammetry



Tejas Pawar, Aman Sharma, and Shiva Ji

1 Introduction

Heritage interpretation is an educational activity that uses authentic materials, first-hand experience, and illustrative media to reveal meanings and relationships rather than merely communicating factual information. Visual or firsthand interactions help people connect to information better than theory [8]. AR technology has become a well-accepted technology among the scientific community and public, which combines real and virtual objects and mixes them into the real environment. In virtual heritage, this technology is used for improving the visitor experience of a cultural heritage site. Heritage interpretations can have many forms, material workshops, heritage walks, conjectural models, Virtual Reality (VR) experiences, and AR applications. In the current scenario in India, we see workshops and heritage walks on a broader scale. VR and AR experiences are scarce and often seen in museums. High-quality experiences are achieved with high-quality and precise documentation. Today, superior methods are used to perform measurements and digital documentation. Photogrammetry is a popular tool in the Architecture, Engineering & Construction (AEC) sector [3].

In the field of cultural heritage, virtual modelling and 3D reconstruction are standard tools for recreating, analysing, and visualising large objects (for example, archaeological sites and architectural buildings) as well as small objects (for example, sculptures, ceramic tiles, silver, marble, and wooden artefacts) [6]. At this time, various technologies can be used to create accurate photo-realistic 3D models, and Photogrammetry is extensively used as it is effortless to use. Whereas, precise modelling of existing 3D data is typically complex and costly since “reality” is

T. Pawar (✉) · A. Sharma · S. Ji
Indian Institute of Technology Hyderabad, Sangareddy, Kandi, Telangana 502285, India
e-mail: md21resch11004@iith.ac.in

complicated in and of itself; the more complex the thing, the more complex the model. Photogrammetry models are frequently created to visualise the historical state of monuments and authentic real-life models.

1.1 Photogrammetry

Photogrammetry and Augmented Reality are tools that have become a keen interest of professionals in all sectors. Photogrammetry procures measurements of size, shape, position, and texture from high-resolution photographs. Photographs are captured from all angles of the structure with a high-resolution camera. These photos help create a 3D model with real-time texture. In its most basic form, a pair of overlapping images are utilised to construct a three-dimensional model, which may then be quantified using proper instruments [9]. These proportions were traditionally depicted on maps and plans as elevations, facades, and/or contours. Photogrammetry is the science of using photos to derive measurements of an object's size, shape, location, and texture. In its most basic form, a pair of overlapping photos is utilised to generate a three-dimensional model, which may then be quantified using suitable equipment [2]. Photogrammetry has a long history of use as a tool to aid in the documentation of cultural assets and is well-established as a measuring science. The data sources begin with photogrammetric data, which includes terrestrial, aerial, and satellite pictures; second, spatial (GIS) data, which includes maps, vector files, and point event locations, as well as lines (counter lines and roads) [2]. Scientific advancements have made the processes far more adaptable in their use, opening up new possibilities for portraying structures as diverse as aboriginal rock painted shelters, historically significant buildings, and Ruins.

1.2 Augmented Reality

AR is widely being used in many applications such as education, entertainment, virtual heritage, simulation and games. In virtual heritage, AR is used to enhance the overall experience of the visitor of a cultural heritage site. Furthermore, the interactive, realistic and complex AR system can enhance, motivate and stimulate students' understanding of certain events, especially for the traditional notion of instructional learning that has proven inappropriate or difficult [7]. Museums have been at the forefront of experimenting with how these new technologies may be utilised as educational aids, touting these breakthroughs as part of their democratisation goal. As a result, we may observe a shift from traditional audio guides to PDAs (Personal Digital Assistants) and eventually to mobile applications. The increasing use of mobile devices for various purposes has increased in the application-development sector. People can use this type of software to acquire critical information and communicate with others creatively [5]. Multiple virtualisations have influenced AR

to be utilised quickly and pleasantly because of improved device capabilities, 3D sensor equipment, and graphics technologies, allowing it to reach a broader market [1]. When it comes to using this tool in educational settings, recent research has shown that AR has helped students better understand reality, examine elements from various perspectives, and construct scenarios that promote simulation or information contextualisation, to mention a few advantages [4].

In the field of heritage, certain qualities of AR can be beneficial, particularly in terms of acquisition, management, and distribution. There are three aspects of AR experience. First, due to the current scenario's lack of modelling, it aids in the reduction of time and ultimate expenses associated with acquisition, modelling, and management. Second, it can create hybrid settings (real and virtual), combining past (non-existing portion, virtually modelled) and present (actual part, not modelled) scenarios to increase comprehension of heritage. Third, AR improves user immersion over VR systems since users can move around, see objects in their actual size, and explore them more naturally, allowing applications to be developed in real-time, on-site.

1.3 Historical Background

Kashi Vishweshwar temple, Kalabgoor, Telangana, is a masterpiece constructed solely out of black granite. The whole site is spread on an approximate 10,000 sq. ft. area. It is believed that temple was built by the Kakatiya dynasty of Telangana and was built in the eleventh century CE. Kashi Vishweshwar temple's architecture is similar to Warangal's Thousand Pillars Temple in Telangana state. The sculptures on these pillars are excellent examples of sculptural mastery, and Architectural Elements are also Exquisite in aesthetic features.

2 Methodology

In the AR application for heritage interpretation, augmentation was applied to a part of Kashi Vishweshwar Temple. The structure in focus was the Nandi Mandapa of Kashi Vishweshwar Temple. The Mandapa was chosen for its excellent condition and details, which covered up to 88. A high-resolution precision camera, Nikon D5600, was used to click photographs covering the maximum surface area of the Mandapa. Shadows in photos hamper the capturing of proper texture in pictures. A cloudy day was preferred to click the photographs to avoid significant shadows. The desired weather also provided precise detail of textures. The whole process was completed within one day. Figure 1 describes a sample of photos captured for this research. Camera properties for these photos are as follows:

Camera Model: Nikon D5600, Exposure Time: 1/400 s, ISO speed: ISO-200, Focal Length: 28 mm, Max aperture: 4.1



Fig. 1 Nandi Mandapa of Kashi Vishweshwar Temple, Kalabgoor

Reality Capture software was used to rebuild the 3D model using the raw data acquired realistically as shown in Figs. 2 and 3. This software was used as it provides additional tools to enhance the texture. The software aligns the photos that are obtained. In the event of any discrepancies, the images were aligned manually using a reference point.

The final product is available in two formats: basic and dense model. The dense model mesh comprises 72.5 million triangles, whereas the simple model mesh is



Fig. 2 Raw data (Textured) obtained after uploading to reality capture

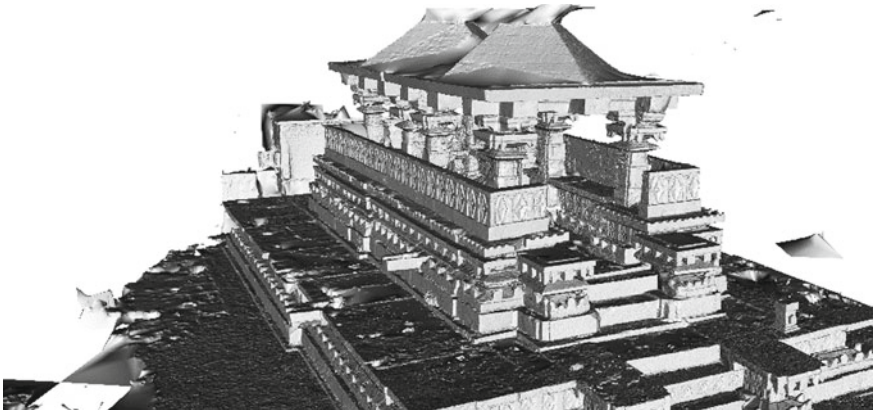


Fig. 3 Raw data (Mesh Model) obtained after uploading to reality capture

made up of 3 million triangles, as shown in Fig. 4. The simple model was chosen because of its smaller file size and ease of use. The high-resolution texturing on the preview model gave it a more realistic appearance. After unwrapping the model, 167 textures with a resolution of 4096×4096 pixels were obtained, as shown in Fig. 5.

MeshLab software was used to access the file, which was exported in wavefront object format. Unwanted surfaces were modified in the mesh lab to improve the model.

The final polished 3D model was then imported into the Unity Augmented Foundation Android Platform. While importing the 3D model into Unity, the Pixels of the material image are enhanced and changed to $16,384 \times 16,384$ pixels, as shown in Fig. 6a. Also, the unlit texture is applied as material, as shown in Fig. 6b, to retain the same exposure value. An application on the android platform was created for accessing the final product. In Unity, both the possibilities of Floor based tracker and Image-based tracker were explored. In-Floor based tracking, this app detects the Floor and places the model. This model is provided with interactive features such as Pinch to scale, Drag to translate and Drag to Rotate on Axis. And in the Image-based tracker Plan of Nandi mandapa was drawn and was set as an Image tracker to show the AR model in Unity.

3 Results and Analysis

Tests were conducted for augmented reality on the floor-based tracker and Image-based tracker. The AR model was provided with interactive accessibilities like Pinch to scale, drag translate and drag-rotate on-axis. The application developed was for 32-bit and 64-bit supported Android phones.

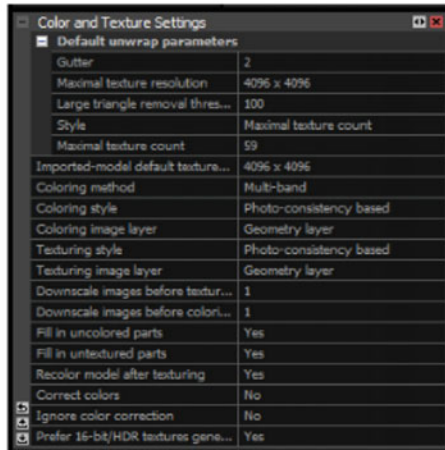
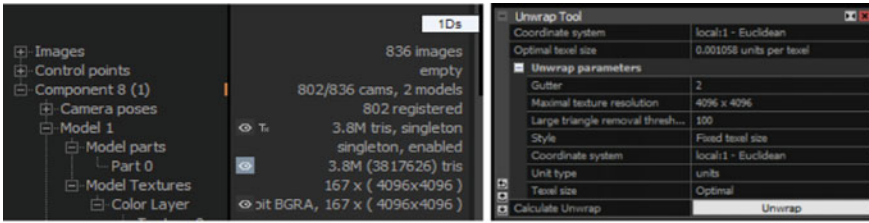
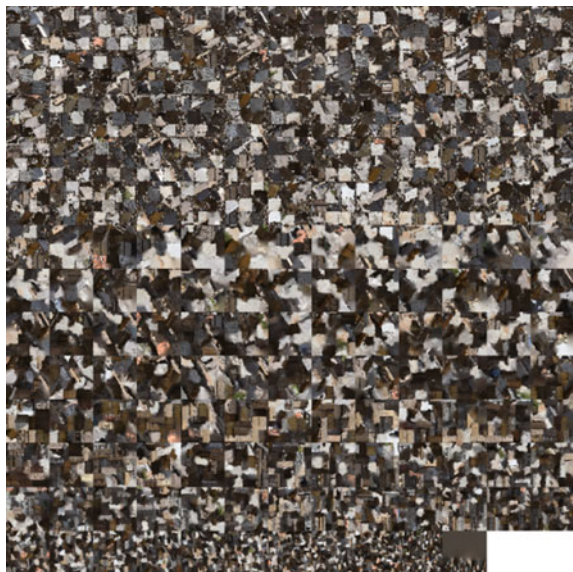
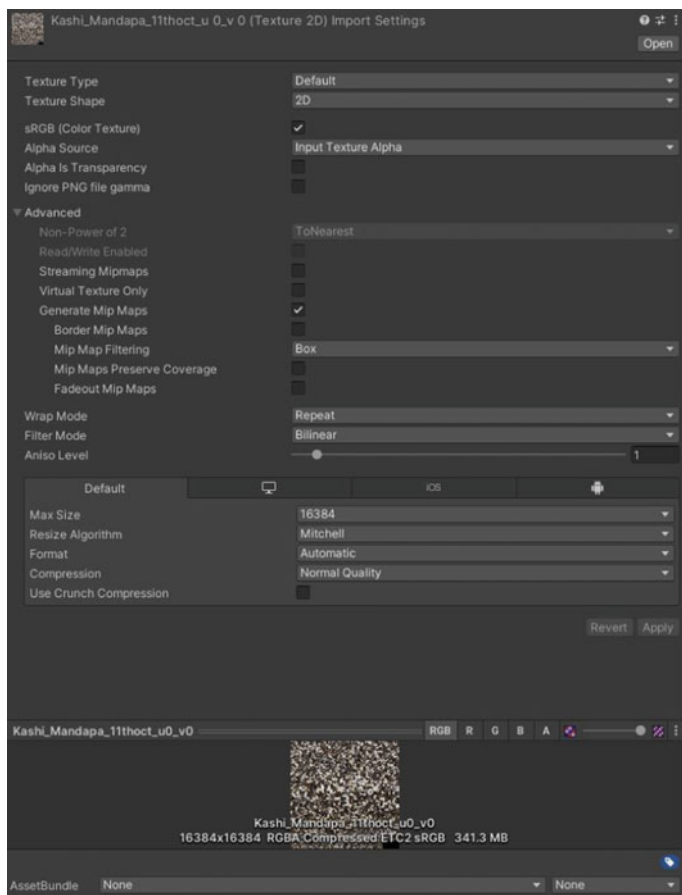


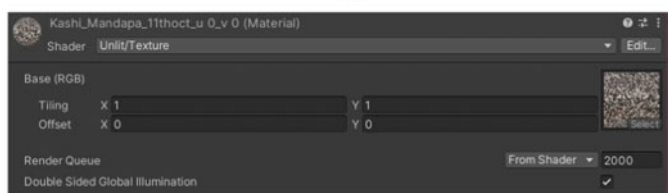
Fig. 4 Settings used in Reality Capture for mesh modelling, unwrapping and texturing the model

Fig. 5 167 Texture obtained from the data of photographs





(a)



(b)

Fig. 6 Texture enhancement in unity

Due to the high-resolution texture wrapped around the model, the object appeared to be more realistic. In Fig. 7a, the 3D virtual model reconstructed from Photogrammetry is placed inside the natural environment, and in Fig. 7b Realistic appearance of the 3D model can be observed along with the original photograph in the background. Also, in Fig. 7c 3D virtual model is placed alongside the existing Nandi Mandapa.

This research used Image in Fig. 8 as Image Tracker for the Augmented reality app. In this research, the possibility of distortion of the Image was checked. The hand-drawn plan of Nandi mandapa, as shown in Fig. 9, was tested on the app. Proportions were maintained while drawing this plan, and as a result, it was found that the Augmented reality app is able to detect Fig. 9 as an image tracker. However, input was given in Fig. 8.



Fig. 7 AR virtual model placed in real-world

Fig. 8 Image-based tracking with the plan of the Nandi Mandapa

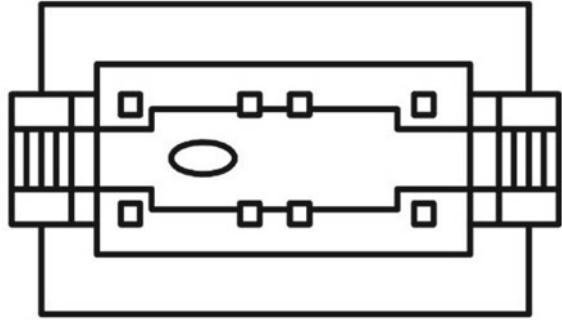
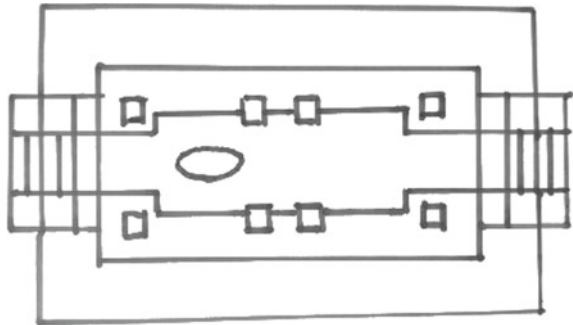


Fig. 9 Image-based tracking with the hand-drawn plan of the Nandi Mandapa was found successful with the generated application



4 Conclusion

In this study, it is observed that augmented reality can be used as a powerful tool in historical interpretation, and the end-user experience can be enhanced and can be more immersive. In addition, the possibility of making AR more interactive gives users the freedom to have better interaction with the model and improve the heritage interpretation. Photogrammetry based AR models of heritage structures can be explored by the user from micro-scale to macro scale. Photogrammetry gives the hyper-realistic appearance of a 3D model, which may also be used in the depiction of a conjecture of dilapidated sites or ruins. Also, another aspect of this study in which Image-based tracking was explored it is found that images with slight distortion but with the same proportions can be detected as image markers. This can help people to draw their own image tracker markers anywhere and get access to the model. These markers can be set as any iconography or pattern associated with the heritage structure so people can have better knowledge about it and can relate to it.

Acknowledgements This work is carried out under the research project titled "Creating Digital Heritage of Representative Architectural Marvels from Each State of North East India", PI: Dr. Shiva Ji, IIT Hyderabad, India.; funded by the Department of Science and Technology, Ministry of Science and Technology, Govt. of India, under ICPS/IHDSR scheme.

References

1. Adhani NI, Rohaya D, Rambli A (2012) A survey of mobile augmented reality applications
2. AL-Ruzouq R (2012) Photogrammetry for archaeological documentation and cultural heritage conservation. In: Special applications of photogrammetry. InTech. <https://doi.org/10.5772/35314>
3. Davila Delgado JM, Oyedele L, Demian P, Beach T (2020) A research agenda for augmented and virtual reality in architecture, engineering and construction. *Adv Eng Inform* 45. <https://doi.org/10.1016/j.aei.2020.101122>
4. Luna U, Rivero P, Vicent N (2019) Augmented reality in heritage apps: current trends in Europe. *Appl Sci (Switzerland)* 9(13). <https://doi.org/10.3390/app9132756>
5. Maples DH, Dima DM (2021) Affectual dramaturgy for augmented reality immersive heritage performance. *Body Space Technol* 20(1):25–36. <https://doi.org/10.16995/bst.368>
6. Merchán MJ, Merchán P, Pérez E (2021) Good practices in the use of augmented reality for the dissemination of architectural heritage of rural areas. *Appl Sci* 11(5):2055. <https://doi.org/10.3390/app11052055>
7. Noh Z, Sunar MS, Pan Z (2009) A review on augmented reality for virtual heritage system. *Lecture notes in computer science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol 5670. LNCS, pp 50–61. https://doi.org/10.1007/978-3-642-03364-3_7
8. Portalés C, Lerma JL, Navarro S (2010) Augmented reality and photogrammetry: a synergy to visualise physical and virtual city environments. *ISPRS J Photogramm Remote Sens* 65(1):134–142. <https://doi.org/10.1016/J.ISPRSJPRS.2009.10.001>
9. Portalés C, Pérez C (2009). Photogrammetry and augmented reality for cultural heritage applications

Augmented Data as an Auxiliary Plug-In Toward Categorization of Crowdsourced Heritage Data



Shashidhar Veerappa Kudari, Akshaykumar Gunari, Adarsh Jamadandi, Ramesh Ashok Tabib, and Uma Mudenagudi

1 Introduction

In this paper, we propose a novel training mechanism to mitigate problems such as data sparsity, high inter-class variance, and low intra-class variance which leads to poor clustering performance. Traditional clustering algorithms such as K-means, Gaussian mixture models (GMM) [3], and spectral clustering [8] rely largely on the notion of *distance*; for example, K-means [11] uses Euclidean distance to assign data points to clusters. Recent advances in deep learning have led to emergence of clustering techniques parameterized by deep neural networks [2, 13, 14, 17, 18] attempting to jointly learn representations, and perform clustering relying on tools like Stochastic Gradient Descent and backpropagation with a clustering objective function. This introduces challenges in choosing an appropriate neural network architecture, and a right clustering objective function. Recent methods [5, 16], attempt to circumvent these problems, limited literature show investigations on the effect of data sparsity and high intra-class variance, usually found in crowdsourced cultural heritage datasets. The apparent architectural differences arise due to data acquisition methods and cultural similarities might lead to assignment of false clusters. In this paper, we empirically demonstrate the use of different transformations such as random—

S. Veerappa Kudari (✉) · A. Gunari · A. Jamadandi · R. A. Tabib · U. Mudenagudi
KLE Technological University, Hubballi, India
e-mail: shashidharvk100@gmail.com

A. Jamadandi
e-mail: adarsh.cto@tweaklabsinc.com

R. A. Tabib
e-mail: ramesh_t@kletech.ac.in

U. Mudenagudi
e-mail: uma@kletech.ac.in

scaling, rotation, and shearing as data augmentation techniques toward increasing the data density, yielding superior clustering performance.

Crowd-sourcing facilitates desired data at scale and involves task owners relying on a large batch of supposedly anonymous human resources with varying expertise contributing a diversified amount of data. In our case, we are interested in obtaining a large image corpus of Indian Heritage Sites with the hindsight of large scale 3D reconstruction toward digital archival and preservation. An essential step in this pipeline is to formulate an efficient deep clustering method toward mitigate the issues outlined above. Toward this-

- We propose a novel training strategy to circumvent the problem of poor clustering performance by
 - introducing data augmentation as an auxiliary plug-in for deep embedded clustering
 - to densify data and facilitate better feature representation considering limited data.
 - to address data with high intra-class and low inter-class variance.
 - to augment data using affine transforms (rotation, scaling and shearing).
 - incorporating Consistency Constraint Loss (CCL) with Mean Squared Error (MSE) Loss to handle introduced transformations.
- We demonstrate our proposed strategy on a crowdsourced Indian heritage dataset and show consistent improvements over existing works.

In Sect. 2, we discuss contemporary works related to clustering. In Sect. 3, we propose a strategy to circumvent the problem of poor clustering performance. In Sect. 4, we discuss the experimental setup carried out on Indian Heritage Dataset. In Sect. 5, we demonstrate results through quantitative and qualitative metrics, and conclude in Sect. 6.

2 Related Works

In this section, we discuss contemporary works addressing clustering using deep features. Classical clustering techniques such as K-means [11], Gaussian Mixture Models (GMM) [3], and Spectral clustering [8] are limited by their distance metrics and perform poorly when the dimensionality is high. Toward this, recent techniques such as Deep Embedded Clustering (DEC) [16], Improved Deep Embedded Clustering [5] extract deep features toward categorization in lower dimension embedding space.

Recent advances in deep neural networks have ushered in a strategy of parameterizing clustering algorithms with neural networks. Deep Embedded Clustering (DEC),

proposed by authors in [16], pioneered the idea of using deep neural networks to learn representations and solve for cluster assignment jointly. The method involves using Stochastic Gradient Descent coupled with backpropagation to extract deep features while simultaneously learning the underlying representations. However, as authors in [5] point, the choice of clustering loss tends to distort the feature space, which consequently affects the overall clustering performance. To mitigate this, the authors propose an under-complete autoencoder to preserve the data structure, leading to improved clustering performance. Inspired by these works, we propose a method to improve Clustering performance by densifying the data distribution. We hypothesize that data distribution sparsity is a significant deterrent in clustering. The problem is further exacerbated when the data exhibits high intra-class variance. We empirically show that using data augmentation as an auxiliary plug-in helps in improving cluster performance. Extensive experiments on cultural heritage dataset show consistent improvements over existing methods.

3 Categorization of Crowdsourced Heritage Data

The crowdsourced heritage data arrives in the incremental fashion, where the number of classes and number of images belonging to class are obscure. More likely, we observe that images belonging to a particular class may arrive in large number while very few samples may arrive for some other classes. This brings the problems of class imbalance and data sparsity. Due to data sparsity, deep learning techniques used for the feature representation of the images fail in their task, making the clustering performance poor. Deep learning architectures like Convolutional Autoencoders (CAE) are sensitive to these problems. Toward this, we attempt to mitigate the data sparsity issue via data augmentation (Fig. 1).

We increase the density of the data, by performing three kinds of data transformations, i.e., random rotation, random shear, and random scaling on the original data, as this would generally make the model more robust in terms of learning. These techniques tend to provide more generic and genuine data. There are many other augmentation techniques as described in [1, 10, 12, 15] used to increase the data density and class imbalance [6, 19].

Convolutional Autoencoder (CAE) has proven to be effective in case of classification, clustering, and object detection. We combine the augmented data with original data to train CAE to generate embeddings toward clustering.

Considering x_i , $i \in \{1, \dots, m\}$ as an image, the transformation t_j , $j \in \{1, \dots, s\}$ applied on x_i generates transformed image x_i^j , represented as $x_i^j = T(x_i)$. The total number of images after augmentation are N , where $N = s \times m$.

The traditional objective function used for training the CAE is Mean Squared Error(MSE) between the input x_i and decoder output x_i' which is as

$$MSE(x, x') = \frac{\sum_{i=1}^N (x_i - x_i')^2}{N} \quad (1)$$

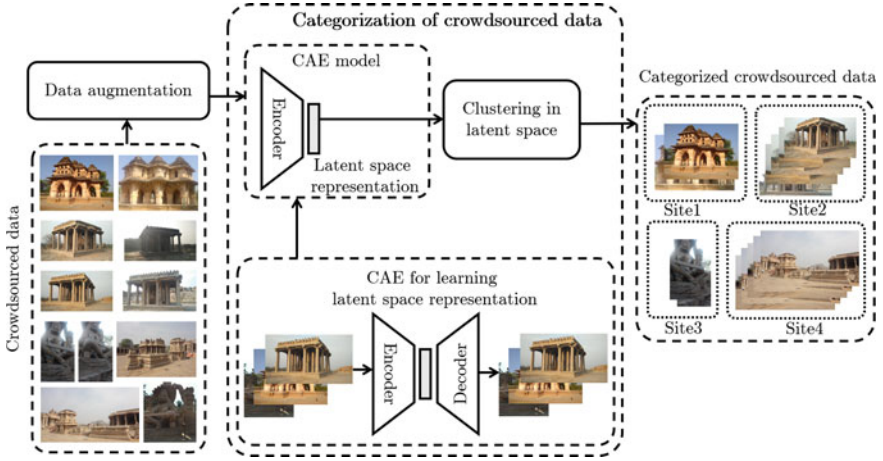


Fig. 1 Categorization of crowdsourced heritage data

The MSE loss for training CAE limits information about the relation between original data and the augmented data. To overcome this, we incorporate the Consistency Constraints [9]. The Consistency Constraints are seen to be effective in the Semi Supervised Learning (SSL). A Consistency Constraint Loss (CCL) can be incorporated by enforcing the predictions of a data sample and its transformed counterpart (which can be obtained by randomly rotating, shearing, or scaling the images) to be minimal. The CCL Loss is defined as follows:

$$L_{CCL} = \frac{1}{NK} \sum_{i=1}^N \sum_{k=1}^K \| p(k|i) - p^t(k|i) \| \quad (2)$$

where N represents the total number of data points and the K represents total number of clusters, $p(k|i)$ represents the probability of assignment of each image x_i , and $p^t(k|i)$ represents the probability of assignment of randomly transformed image x_i^t to cluster k . $p(k|i)$ is parameterized by assuming they follow the Student's T distribution as follows:

$$p(k|i) \propto \left(1 + \frac{\| z_i + \mu_k \|^2}{\alpha} \right)^{-\frac{\alpha+1}{2}} \quad (3)$$

Here z_i is the feature representation of the image x_i , μ_k represents the cluster center of cluster k . If U represents the cluster centers then $U = \{\mu_k, k = 1 \dots K\}$ which are initialized by K-means and tuned as the training progress. The overall objective function of CAE is now defined as

$$Loss = MSE(x, x') + L_{CCL} \quad (4)$$

We use K-means technique to quantify our results and depict how augmentation can improve the performance of the clustering. We show how data augmentation can improve the performance of the existing state of art methods Deep Embedded Clustering (DEC) and Improved Deep Embedded Clustering (IDEC), where CAE is used as the initial feature extractor. We provide the extensive ablation study of these methods over the combinations of different CAEs trained.

4 Experiments

4.1 Dataset

We extensively experiment on crowdsourced Indian Digital Heritage (IDH) Dataset. The dataset is collected through a platform sourced by crowd. We consider 10 classes of this dataset with 150 images per class toward experimentation as these 10 classes consists of high intra-class and low inter-class variance. The considered dataset undergoes augmentation like random rotation, random shearing, and random scaling. Random rotation of images is performed over the range of 0–90°, random shear is performed over 50° of transformation intensity and random scaling is performed over the scale of 0.5–1.0. We generate around 6000 images through these transformations. The same dataset is used throughout the experimentation to maintain the uniformity in comparison of results in different experiments in different environments.

4.2 Training Setup

- Runtime Environment: Nvidia GP107CL Quadro P620
- Architecture: Autoencoder

– Encoder:

Contains 4 VGG Blocks

VGG Block has 2 Convolution Layers followed by a maxpooling layer

Batch normalization layer was used at the end of each layer before the activation function

Activation Function: ReLU

– Decoder:

Decoder part of the model consists of convolution transpose layers with batch normalization layer at the end of each layer before the activation function

Activation Function: ReLU, Sigmoid (Output Layer)

- Batch Size: 16
- Learning Rate: 0.001
- Number of Epochs: 500 (CAE), 2000 (DEC and IDEC)
- Optimizer: Adam

4.3 Evaluation Metrics

Toward evaluation of proposed strategy and comparison with state-of-the-art methods, we use Unsupervised Clustering Accuracy (ACC), Normalized Mutual Information (NMI), Adjusted Rand Index (ARI).

4.3.1 Unsupervised Clustering Accuracy (ACC):

It uses a mapping function m to find the best mapping between the cluster assignment output c of the algorithm with the ground truth y which can be defined as

$$ACC = \max_m \frac{\sum_{i=1}^N 1\{y_i = m(c_i)\}}{N} \quad (5)$$

For the given image x_i , let c_i be resolved cluster label and y_i be the ground truth label, m is the delta function [7] that equals one if $x = y$ and zero otherwise. m maps each cluster label c_i to the equivalent label from the datasets. The best mapping can be found by using the Kuhn-Munkres algorithm [4].

4.3.2 Normalized Mutual Information (NMI)

It measures the mutual information $I(y, c)$ between the cluster assignments c and the ground truth labels y and is normalized by the average entropy of both ground labels $H(y)$ and the cluster assignments $H(c)$, and can be defined as

$$NMI = \frac{I(y, c)}{\frac{1}{2}[H(y) + H(c)]} \quad (6)$$

4.3.3 Adjusted Rand Index (ARI)

It computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. It is defined as

$$ARI = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex} \quad (7)$$

5 Results and Discussions

In this section, we discuss the results of the proposed strategy toward categorization of crowdsourced Indian Heritage (IDH) dataset and compare the results with state-of-the-art methods.

We measure the clustering performance by reporting the unsupervised clustering accuracy, NMI and ARI. From Table 1 we observe, CAE trained with data augmentation yields better performance over CAE trained without augmentation. We see an improvement of **2.74%** when trained with MSE loss and an improvement of **15.21%** when trained with a combination of MSE and CCL, as CCL loss mainly depends on augmented data. This improvement is significant in the context of clustering data with high intra-class variance.

To discern the effect of individual augmentation techniques (rotate, scale and shear), we choose samples in the combination of—{ori, rot}, {ori, sher} and {ori, scal}. The results are presented in Table 2. We observe, set {ori, rot} shows poor performance compared to augmentations consisting of {ori, sher} and {ori, scal}. We hypothesize that the performance drop for {ori, sher} can be attributed to the fact that, the CAE is not equipped with appropriate symmetry inductive bias that enables it to learn rotation-invariant features.

Table 1 Performance of CAE trained with and without augmented data. CAE-WAug represents CAE model trained without augmented data and CAE-Aug represents CAE model trained with augmented data

	MSE			MSE + CCL		
	ACC	NMI	ARI	ACC	NMI	ARI
CAE-WAug	0.5424	0.4880	0.3377	0.4035	0.3752	0.1880
CAE-Aug	0.5698	0.5327	0.4000	0.5566	0.4936	0.3336

Table 2 Effect of augmentation on performance of the original data. Original, rotated, sheared, scaled data are represented as ori, rot, sher and scal, respectively

	CAE – MSE			CAE – (MSE + CCL)		
	ACC	NMI	ARI	ACC	NMI	ARI
ori + rot	0.4355	0.3168	0.2155	0.3491	0.2681	0.1493
ori + sher	0.5189	0.4480	0.3144	0.4413	0.3574	0.2091
ori + scal	0.5183	0.4168	0.2846	0.4346	0.3419	0.2151

5.1 Ablation Study

In this section, we perform ablation study using DEC [16] and IDEC [5] with and without considering augmentation. In Table 3, we provide the ablation study of DEC which is unsupervised clustering technique that jointly optimizes the cluster centers and the parameters of the CAE. KL-divergence between the auxiliary and target distribution optimizes the objective function. From Table 3, we infer that providing CAE with the augmented data followed by DEC considering original data increases the accuracy by **5.61%**. While providing augmented data to DEC, with CAE being trained with original data hinders the performance. Hence, only CAE is trained with original and augmented data ensuring the objective is met.

In Table 4 we provide the ablation study of the Improved Deep Embedded Clustering (IDEC). IDEC is an improvement over IDEC, which not only jointly optimize the cluster centers and parameters of the CAE, but also preserve the local structure information. They use KL-divergence between the auxiliary and target distribution as their objective function along with the MSE loss of the CAE. From Table 4 it can be observed that providing CAE with augmented data with MSE+CCL loss, then providing the trained CAE to IDEC, where IDEC is trained on original data improves the performance by depicting the increase in accuracy by **7.43%**. While training the IDEC with augmented data with CAE trained on original data only hinders the performance.

From the experiments we observe, it is better to train the CAE with MSE+CCL as the integrity loss, with augmented data. The CAE trained in such an environment is incorporated for initial feature representation to the IDEC by providing original data, to perform better than other methods.

Table 3 Comparing results of proposed methodology with DEC [16]. CAE-WAug and CAE-Aug refers to CAE trained without and with augmentation, respectively. DECWAug and DEC-Aug refers to DEC trained without and with augmentation respectively. We show how combination of augmentation applied to CAE and DEC may affect the clustering performance

Loss →	MSE			MSE + CCL		
	ACC	NMI	ARI	ACC	NMI	ARI
DEC [16] CAE – WAug + DEC – WAug	0.4113	0.3874	0.2271	0.3625	0.3625	0.8196
CAE – WAug + DEC Aug	0.3096	0.3013	0.1530	0.2927	0.2210	0.1201
CAE – Aug + DEC WAug	0.4674	0.4876	0.3076	0.4492	0.4665	0.2716

Table 4 Comparing results of the proposed methodology with IDEC [5]. CAE-WAug and CAE-Aug refers to CAE trained without and with augmentation respectively. IDECWAug and DEC-Aug refer to IDEC trained without and with augmentation, respectively. We show how the combination of augmentation applied to CAE and IDEC may affect the clustering performance

Loss →	MSE			MSE + CCL		
Method ↓ , Metric →	ACC	NMI	ARI	ACC	NMI	ARI
IDEC [5]	0.5098	0.4903	0.3020	0.3625	0.4340	0.2511
CAE – WAug + IDEC – WAug						
CAE – WAug + IDEC Aug	0.3611	0.3300	0.1703	0.3514	0.3335	0.1856
CAE – Aug + IDEC WAug	0.4983	0.4942	0.3130	0.5841	0.4340	0.3662

6 Conclusions

In this paper, we have defined data augmentation as an auxiliary plug-in for deep embedded clustering that densifies data helping in accurate clustering performance. We have demonstrated how data augmentation helps to increase the data density yielding superior clustering performance when the data is considerably less in amount. Extensive experimentation is done on setting up the right objective function. Our main objective is to cluster data with very less inter-class variance and very high intra-class variance. We have demonstrated our experiments on crowdsourced heritage dataset. We also show, how certain augmentation techniques uphold the clustering objectives (such as random shear and random scale), while some of them hinders the same (random rotation). We demonstrate our results on Indian Digital Heritage (IDH) dataset to show our methodology shows better performance to state-of-the-art clustering algorithms. Deploying clustering algorithms for critical applications warrants circumspection and is still a work in progress and we believe our work is a step in this direction.

Acknowledgements This project is partly carried out under the Department of Science and Technology (DST) through ICPS program—Indian Heritage in Digital Space for the project “Crowd-Sourcing” (DST/ ICPS/ IHDS/ 2018 (General)) and “Digital Poompuhar” (DST/ ICPS/ Digital Poompuhar/ 2017 (General)).

References

1. Bloice M, Stocker C, Holzinger A (Aug 2017) Augmentor: an image augmentation library for machine learning. *J Open Sour Softw* 2 . <https://doi.org/10.21105/joss.00432>
2. Caron M, Bojanowski P, Joulin A, Douze M (2019) Deep clustering for unsupervised learning of visual features
3. Chen J, Zhang L, Liang YC (May 2019) Exploiting Gaussian mixture model clustering for full-duplex transceiver design. *IEEE Trans Commun* 1. <https://doi.org/10.1109/TCOMM.2019.2915225>
4. Cui H, Zhang J, Cui C, Chen Q (Jan 2016) Solving large-scale assignment problems by Kuhn-Munkres algorithm. <https://doi.org/10.2991/ameii-16.2016.160>
5. Guo X, Gao L, Liu X, Yin J (2017) Improved deep embedded clustering with local structure preservation. In: *Proceedings of the twenty-sixth international joint conference on artificial intelligence, IJCAI-17*, pp 1753–1759. <https://doi.org/10.24963/ijcai.2017/243>
6. Guo X, Zhu E, Liu X, Yin J (2018) Deep embedded clustering with data augmentation. In: Zhu J, Takeuchi I (eds) *Proceedings of The 10th Asian conference on machine learning. Proceedings of machine learning research*, 14–16 Nov 2018, vol 95, pp 550–565. PMLR. <http://proceedings.mlr.press/v95/guo18b.html>
7. Gupta SC (1964) Delta function. *IEEE Trans. on Educ.* 7(1):16–22. <https://doi.org/10.1109/TE.1964.4321835>,
8. Hamad D, Biela P (2008) Introduction to spectral clustering. In: *2008 3rd international conference on information and communication technologies: from theory to applications*, pp 1–6. <https://doi.org/10.1109/ICTTA.2008.4529994>
9. Han K, Vedaldi A, Zisserman A (2019) Learning to discover novel visual categories via deep transfer clustering
10. Inoue H (2018) Data augmentation by pairing samples for images classification. <https://openreview.net/forum?id=SJn0sLgRb>
11. Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY (2002) An efficient k-means clustering algorithm: analysis and implementation. *IEEE Trans Pattern Anal Mach Intell* 24(7):881–892. <https://doi.org/10.1109/TPAMI.2002.1017616>
12. Mikołajczyk A, Grochowski, M (2018) Data augmentation for improving deep learning in image classification problem. In: *2018 international interdisciplinary PhD workshop (IIPHDW)*, pp 117–122. <https://doi.org/10.1109/IIPHDW.2018.8388338>
13. Mrabah N, Khan NM, Ksantini R, Lachiri Z (2020) Deep clustering with a dynamic autoencoder: from reconstruction towards centroids construction
14. Ren Y, Wang N, Li M, Xu Z (2018) Deep density-based image clustering. *CoRR* **abs/1812.04287**
15. Shorten C, Khoshgoftaar T (2019) A survey on image data augmentation for deep learning. *J Big Data* 6:1–48
16. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis
17. Yang J, Parikh D, Batra D (2016) Joint unsupervised learning of deep representations and image clusters
18. Zhan X, Xie J, Liu Z, Ong YS, Loy CC (Jun 2020) Online deep clustering for unsupervised representation learning. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*
19. Zhong Z, Zheng L, Kang G, Li S, Yang Y (2017) Random erasing data augmentation

Evolution of Bagbazar Street Through Visibility Graph Analysis (1746–2020)



Shilpi Chakraborty  and Shiva Ji 

1 Introduction

Cultural assimilation in the urban setting occurs through activities aimed at expanding the accessibility of offerings associated with urban development and initiatives and practical performances aimed at enhancing communal interaction by the use of communal places. Space syntax provides a theoretical paradigm that engages in exchange throughout and within the urbanization, spatial analysis, and cultural engagement, with the ability to both stimulate and create analytical views to urban centers [16].

The quality of urban environments has worsened as the need for individual and commodity mobility has increased. As a consequence of the growth and expansion of digitization, transportation has transformed individuals' lives, allowing them to go to distinct destinations in minimal time. Consequently, there has been a substantial transformation in the cultural and physical structure of the neighborhood, with more excellent utilization of vehicle sources of mobility, allowing for the expansion of productive activity in areas far from homes [25]

Hillier and Hanson advocate for a particular architectural understanding of spatial arrangement that allows for correlative assessment of structures and towns over period and area, avoiding the unintentional estimation of societal narratives onto built styles through numerous different by effective confrontation, less rhetorical viewpoints [11]. There are four prominent techniques for spatial background: First, and most importantly history as “backdrops”: study in this field typically offers historical material to establish the place(s) of a study case-secondly, chronology as

S. Chakraborty (✉) · S. Ji

Department of Design, Indian Institute of Technology Hyderabad, Sangareddy, Kandi, Telangana 502284, India

e-mail: md20resch11004@iith.ac.in

S. Ji

e-mail: shivaji@des.iith.ac.in

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022
U. Mudenagudi et al. (eds.), *Proceedings of the Satellite Workshops of ICVGIP 2021*,
Lecture Notes in Electrical Engineering 924,
https://doi.org/10.1007/978-981-19-4136-8_5

syntactic “developmental stages” in community patterns and architectural categorizations. Thirdly, there are “syntactic, morphological narratives,” which put the social aspects of geographical habitation layout and physical structure to the forefront, having the primary goal of comprehending the current physical environment. Baker’s term spatial narratives is employed beyond qualifier to characterize the fourth type, focuses on how sociological situations arrange and emerge structured over time and space, and extends above morphology that describes specific socioeconomic with cultural elements of urban society. It’s worth noting that neither of these classifications is to be strict; therefore, there’s a lot of crossover across them.

The development of livability in a community so that socioeconomic products do not endanger the presence of organic materials and assure their constant growth is a method. The linkage of the majority of the definition of the planning stage is to cognitive argumentation frameworks and natural phenomenon evaluation. Various rationalist and empirical techniques investigate the structure, form, and placement of cities related to urbanization. The consideration of structure and space in settlements changes depending on social, economic, or geographical aspects. The broad geographical layout of a neighborhood’s social and financial activity, which connects to urban morphology regarding economic elements is urban fabric [3, 26]. “Unit for Architectural Studies” created by Bill Hillier in the mid-1970s, was a significant incentive. Besides from an exhilarating compendium of highly influential scientific publications from the 1970s, among the time of the facility’s inception [11, 15], three books by Bill Hillier and Julienne Hanson: “The social logic of space” (Hillier and Hanson 1984), “Space is the machine” (Hillier 1996b), and “Decoding homes and houses” (Hillier 1996c) (Hanson 1998). The second volume, “Space is the Machine,” traces the development of space syntax from the 1980s to the earlier 1990s, concentrating on the theory’s structural and practical aspects.

The spatial syntax approach considers the significance of space as well as the connections between space and movements. Based on writers, some of these approaches analyze space in terms of its essential qualities. Numerous people were debating the remote locations. It is this, instead of the interactions among towns and buildings, that space syntax is primarily [13].

2 Space Syntax Method

Space syntax is distinct from traditional urban morphology in that it concentrates upon open area networks to achieve a kind of spatial depiction. (Hiller, B, 1996). Space syntax philosophers refer to collections of simulated interactions among spaces as “visibility,” and the utilization of these words is to describe something they believe to be the inherent features of space [7]. Turner claimed a connection among isovist and visibility graphs, with utilization in the analysis of spatial cognition, based on a significant amount of research on the comparability and correlation among spatial syntactic paradigm, spatial representation, and cognitive map. By equating the space syntax assessment and image map of various cities, Shokouhi [27] proposed that the cohesive framework constituted

of the town's crucial components, the natural institution of structures, and the consistency among landmark graphic cross-functional is the critical component to urban perception. In an attempt to analyze the spatial patterns of communities, Tao [28] merged the axis map and the visual map. Space syntax displays graphic spatial information, which is then utilized to explain environmental variables that influence users' spatial behavior patterns and experiences. These quantified information reveal the connections between urban roads, blocks, and buildings [32]. Syntactic metrics are often employed to quantify the compositional characteristics of urban and architectural contexts. Several demonstrations are to correspond with motion and user activity in several contexts, notably museums [29].

Koohsari [17] compared the conceptual framework and the space syntax framework on the historical area of Taiping Street in Changsha, Hunan Province, and discovered it as the space syntax paradigm had a significant level of connection with the nodal and transportation system in the conceptual framework. Urban scholars whose concern in the link connecting the physical qualities of cities and the social interactions of cities takes them into conceptual and analytical territory that has yet to explore in the historical study [19, 23]. As a preliminary step, developing a better technique of defining urban space since it appears throughout historical cartographic materials would be beneficial, as it would aid in deciphering the historical documentation of what individuals performed in these areas.

Cartography is one of the few generally dependable tools for urban scholars to reach the past's physical existence despite its inherent problematic characteristics as historically distorted realities. However, a requirement of vital equilibrium is comprehending the theological influences that create the physical reality with the psychologist Gibson refers to as life's physical "capabilities" [9]. According to Hillier and Hanson [14], the promise of space syntax for historians is in giving a beginning for recognizing and, more significantly, defining the character of such spatial "capabilities." The attraction for space syntax scholars is to a preceding version of nineteenth-century city scholars and historical researchers who used cartography to illustrate survey data to try to investigate queries like the housing dispersion of immigrant communities and housing discrimination, as well as quite broadly to such academics interested in the town's urban fabric as well as its depiction [6, 8, 20, 24]. We concentrate on syntactic measurements that can be determined using "visible graphs" in this paper [22, 30]. A visibility graph is created by concatenating a grid on a top-down representation of a space (e.g., from a CAD drawing) utilizing space syntax software (DepthmapX). Topological techniques for assessing grid cell closeness and inter-visibility are to derive syntactic measurements for every grid cell.

3 Kolkata: Bag Bazar Street

Kolkata, India's West Bengal state capital, is the world's fourteenth biggest city and India's first metropolitan unit (Fig. 1). The town runs north-south along the Hooghly (Ganga) River on the eastern edge of the famed river delta Bengal valley.

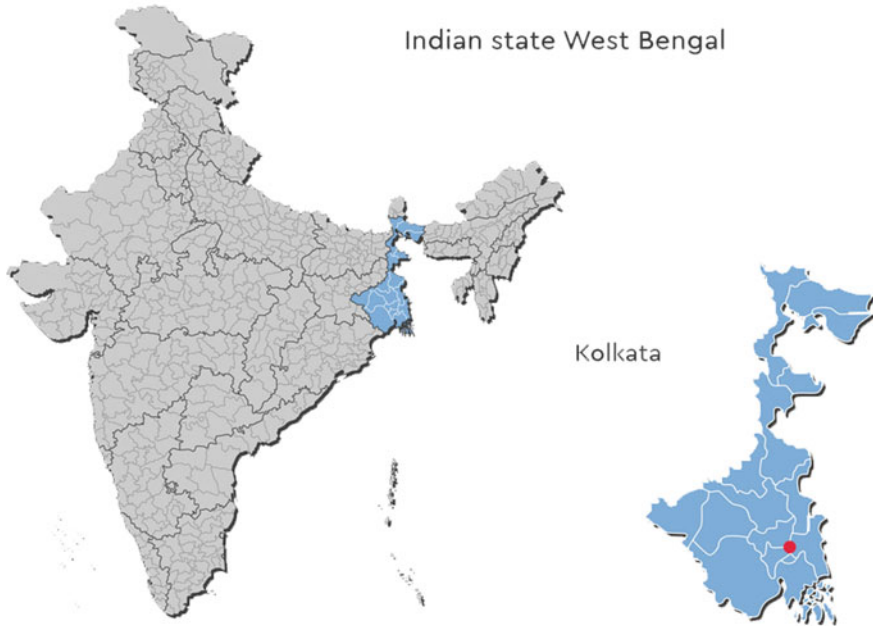


Fig. 1 India map with West Bengal highlighted (left); Kolkata in West Bengal (Right) [31]

A large portion of the area used to be a swamp, gradually restored to suit the area's burgeoning population. The remnant undeveloped part of the East Kolkata Wetlands with classification as a "vitaly significant wetland." Kolkata's history began in 1690, with the establishment of the East India Company to export.

BagBazar street is one of Kolkata's three significant arteries, also known as the grandmother road in ekanto poribar (joint family). Bagbazar, a suburb in North Kolkata, has long acted as a bastion for the Bengali aristocracy and adjacent Shyambazar. Bagbazar has made essential contributions to Kolkata's growth and development. When Kalikata became populated, the English gradually abandoned Sutanuti as a residential place. Perin's Garden, a recreational resort on the town's northern outskirts, remained, where promised personnel of the British East India Company would generally join the spouses for an evening stroll and moonlight celebration. Nonetheless, it fell out of favor after 1746 and sold for Rs. 25,000 in 1752. Captain Perin was the lucky recipient of several ships. Colonel C.F. Scott began manufacturing gunpowder on the property in 1754 (Fig. 2).

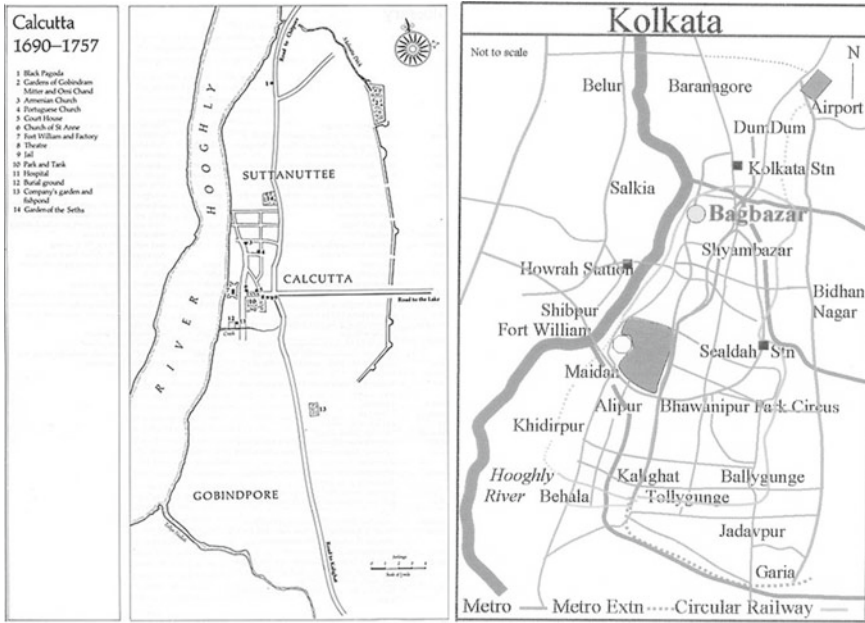


Fig. 2 Bag Bazar Street in 1746 (Left) [18]; in 2020 (Right) [33]

4 Methodology

4.1 Data and Survey

The utilization of primary and secondary questionnaires is to gather data for this research. Observations, conversations, statistics, and streets data are all part of the preliminary study’s objectives (Table 1). In this article, we deliver a quantitative analytic approach for various attributes of urban area of Bagbazar street based on analyses performed with our proposed spatial modeling method (Space Syntax Method).

Table 1 Data analysis assessment methods

Sl. no	Goals	Tools	Findings
1.	Exploring the development of Bag Bazar Street in Kolkata	Descriptive Qualitative Analysis	Development of Bag Bazar Street, Kolkata
2.	Visibility Graph Analysis of Bag Bazar Street, Kolkata	Space Syntax Analysis	The Degree of development of Bag Bazar Street, Kolkata

Qualitative analytic approaches will be employed for the first aim, while quantitative analytical techniques for the second. The table below lists information analysis methodologies, study purposes, study tools, and expected research findings.

4.2 Analysis Method

Understanding the urban setting as a collection of structures linked by a space network cycling between modules is the first element in establishing Space Syntax from the standpoint of urban scale. The system connects each set of road parts. The structure is the optimum consequence of all aspirations to general purposes within the geographical matrix. This is the power that connects all. It has a distinct morphology or connection arrangement. The cartographic maps help in understanding the growth of the Kolkata as shown in Fig. 3, 4 and 5.

The geographical and geometrical matrices research helps analyze the context of urban settings and their potential impact on interpersonal behavior and commercial activity. Connectedness, integration, management, and choice are among the metrics

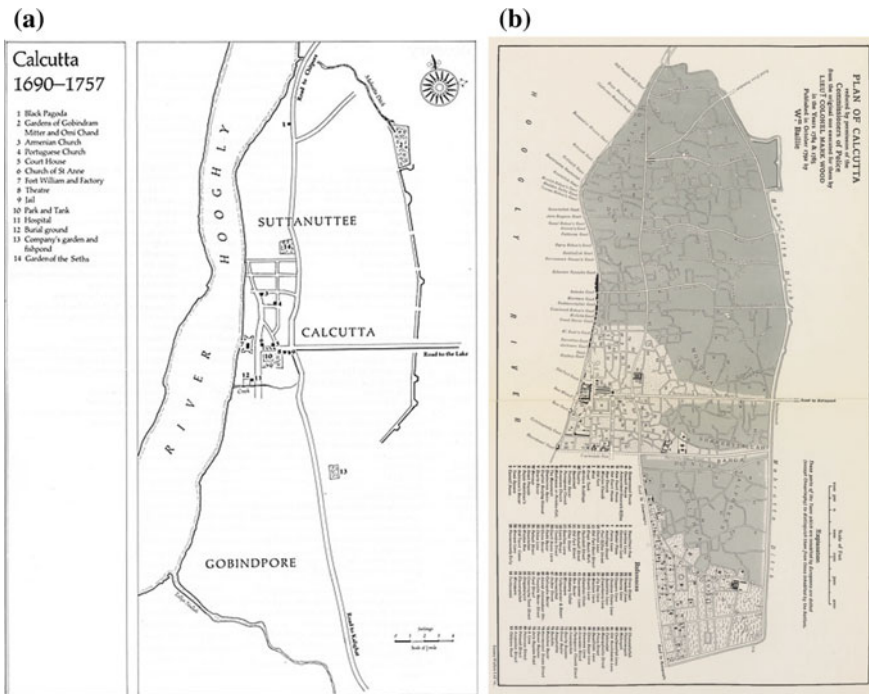


Fig. 3 Calcutta in a 1746 (Left) [18], b 1785 (Right) [4]

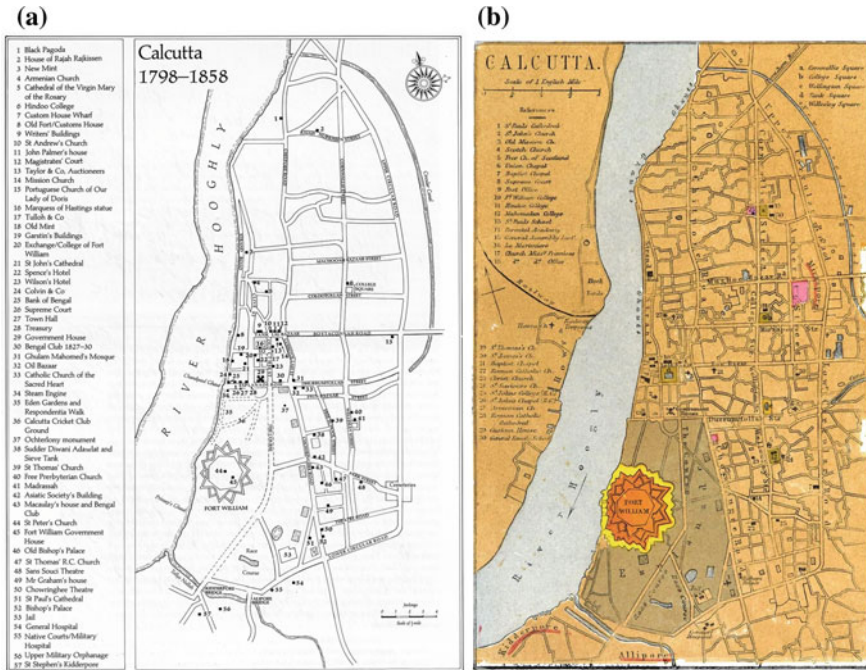


Fig. 4 Calcutta in a 1825 (Left) [18], b 1865 (Right) [4]

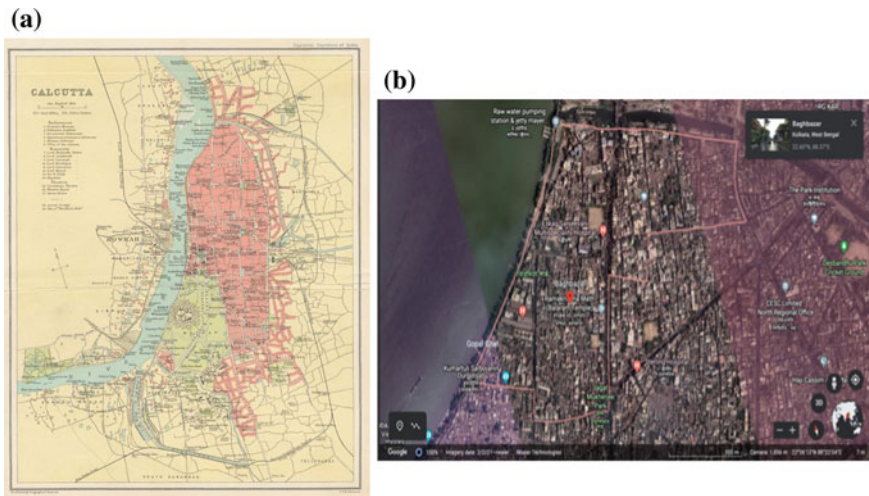


Fig. 5 Kolkata in a 1931 (Left) [5], b 2020 (Right) [1]

established by Space Syntax researchers to define conduct. The utilization of links between some of them to develop particular navigational layout features [2, 12].

Planners, on either hand, consider the region as a whole; nevertheless, they seldom pay attention to particular urban areas. On the other hand, metropolitan designers in our neighborhood, but they are scared to grow outside to the size of a whole metropolis. As a corollary, it has a lot of potential since an advising instrument in large metropolitan strategic planning, as it may help in zoning planning and determining the implications.

5 Result and Discussion

5.1 General Description

Historical cartographic maps are being amassed (1746–2020) For assessment. These maps differed substantially and depicted the plan of a city in a specific year. The circumstances that evolved as a result of Bag Bazar Street’s planning are in Fig. 6. The authors reconstructed these maps and put them through DepthmapX for evaluation. The evolution of Bag Bazar street is into six stages (Fig. 7). Bag Bazar street has developed from a single street to an entire neighborhood as a result of urbanization.



Fig. 6 The timeline of events led to evolution of Bagbazar street. (Source Author)

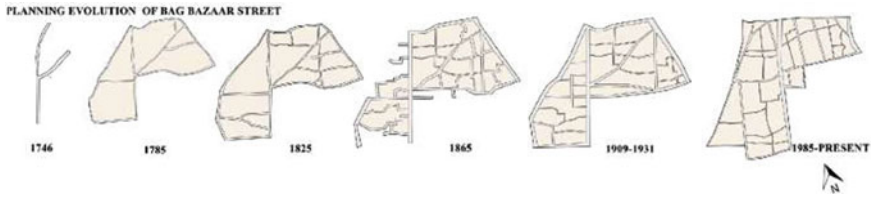


Fig. 7 Planning evolution of Bag Bazaar Street. (Source Author)

It’s possible that following 1985, development slowed owing to a shortage of space and congestion. Hence the determination of the evolution of Bag Bazar street is by the amalgamation of information through timeline and cartographic maps.

5.2 Axial Analysis

Hillier differentiated external and internal aspects of space when it came to accuracy in respect to geographical components. Geometry, structure, roughness, solidity, and patterning of space are all fundamental characteristics of space. Since they are simple to notice, such qualities are straightforward to express using terms like “an open area” or “a short street.” External aspects of space, on the other hand, deal with unseen geographical connections. The size and shape of the items are irrelevant. Because it solely considers whether another space connecting to adjacent spaces by respecting lateral shifts and angular variation inside a complex system, space syntax exclusively operates for external characteristics [10].

Space syntax deals with the crucial components of the axial line using the reasoning of external spatial characteristics. In its simplest terms, the axial map, an axial line, is a targeting reticle that depicts mobility routes and shapes. The axial map, which considers the entire distance of a road, is a system connected to all the various avenues that intersect it. This enabled us to get beyond the restrictions of traditional transportation concepts, which defined a road network as a set of nodes in road junctions connected with sections of the road among them [21].

The extent to which space is connected When assessing the level of connection of a region, it is critical to look at the element of space consistency. The primary data in this study is in terms of an axial map. Axis lines show linkages that can occur in linked places such as highways and other open areas. Bag Bazar Street has a medium connectivity rating of with highest number of axial lines that is 304 from 1985 to the present (Table 2). DepthMap, a tool for visibility research of urban fabric, was utilized in this research as open-source software. Researchers can use many metrics for analysis with DepthMap, as shown in Table 3 and relation among them in Fig. 8.

Table 2 Axial lines of Bag Bazar Street

Year	Axial lines
1746	22
1785	130
1825	178
1865	252
1909–1931	184
1985-Present	304

Table 3 Attributes of visibility graph analysis

Measures	Metric	Characteristics
Size	Connectivity	The amount of near neighbors with whom a place is immediately correlated. This is a fixed local metric
Global	Harmonic Mean Depth	The mathematical mean of the inverses of depths through one space to all the others
	Mean Depth	The average depth measurement computed through one grid unit over the next
Nomalized Depth	Integration [HH]	Hillier and Hanson’s method gives the fraction of any matrix cell’s visibility to the visibility of all grid points in the area. It also depicts an axial line’s topological closeness toward other lines
	Integration [P-Value]	It aids in the resolution of the size issue when evaluating structures of varied sizes and matrix sizes
	Integration [tek1]	Visibility Graph Analysis integration should be normalized, according to Teklenburg, in order to compare the integration produced from different analyses
	Choice	An axial line’s distance covered across all axial lines. It denotes the amount of shortest routes that link one axial line to another. The shortest topographically computed path is the difference of two axiallines
	Intensity	Calculates the amount of change of entropy compared to total depth to determine the compared asymmetrical geographical system
Complexity	Relavitized Entropy	A computation of the anticipated depth value range
	Entropy	The investigation of the depth parameters of a grid unit and its neighbors
Control	Control	Identifying dominating regions visually
	Controllability	Identifies locations that may be conveniently checked when walking

The visibility graph analysis of space syntax method as shown in Figs. 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 and 20 through the analysis in Table 4, the evolution bag bazaar street with the lowest transformation is in the year 1865 and the highest in 1985 present.

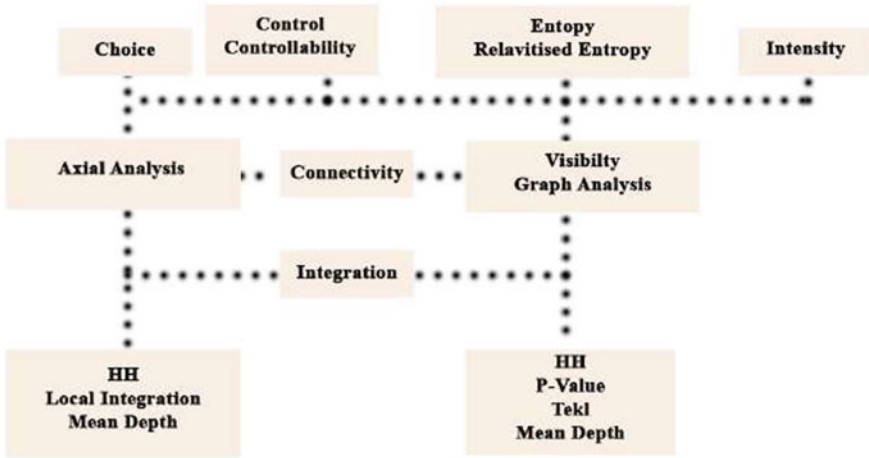


Fig. 8 Space syntax: axial analysis and Visibility graph analysis

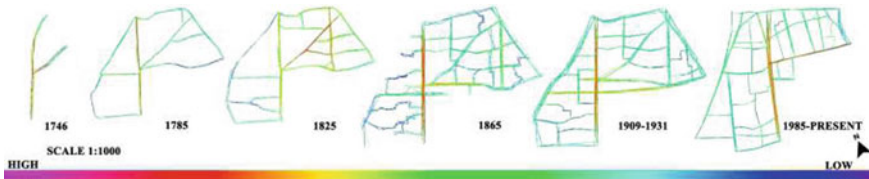


Fig. 9 Degree of connectivity map of evolution of Bag Bazar Street. (Source Author)



Fig. 10 Harmonic mean depth map of evolution of Bag Bazar Street. (Source Author)



Fig. 11 Mean depth map of evolution of Bag Bazar Street. (Source Author)



Fig. 12 Integration (HH) of evolution of Bag Bazar Street. (Source Author)



Fig. 13 Integration (P-value) of evolution of Bag Bazar Street. (Source Author)



Fig. 14 Integration (tekl) map of evolution of Bag Bazar Street. (Source Author)

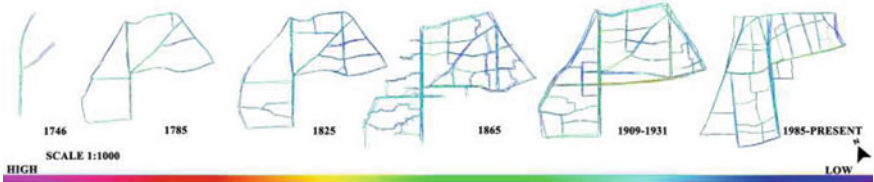


Fig. 15 Choice map of evolution of Bag Bazar Street. (Source Author)



Fig. 16 Intensity map of evolution of Bag Bazar Street. (Source Author)



Fig. 17 Relativized entropy map of evolution of Bag Bazar Street. (Source Author)

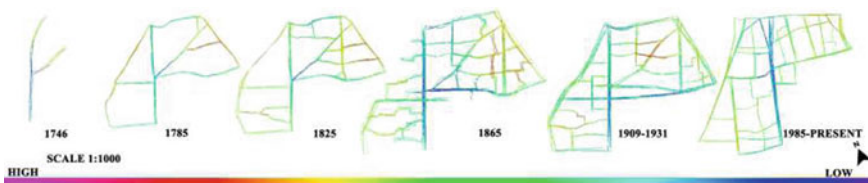


Fig. 18 Entropy map of evolution of Bag Bazar Street. (Source Author)

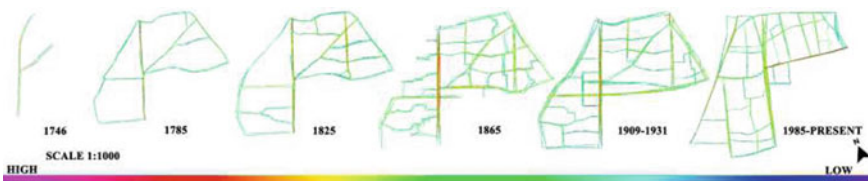


Fig. 19 Control map of evolution of Bag Bazar Street. (Source Author)



Fig. 20 Controllability map of evolution of Bag Bazar Street. (Source Author)

Table 4 Visibility graph analysis

Attributes	Year				
	1746	1785	1825	1865	1985–2020
Connectivity	High	Medium	Medium	Low	Medium
Harmonic Mean Depth	Low	Medium	Medium	Low	Medium
Mean Depth	Low	Medium	Low	Low	Low
Integration [HH]	Medium	Medium	Low	Low	Medium
Integration [P-Value]	Medium	Low	Low	Low	Medium
Integration [tek]	Low	Medium	Medium	High	Medium
Choice	Low	Low	Low	Low	Low
Intensity	Medium	Low	Low	High	Medium
Relativized Entropy	Low	Medium	Low	Low	Medium
Entropy	Low	Low	Medium	Low	Medium
Control	Medium	Medium	Medium	Medium	Medium
Controllability	High	Medium	Medium	Medium	Medium

6 Conclusion

It was feasible to focus on providing a unique viewpoint on urban procedures than those enhanced by conventional periodizations of historical context, which prefer to deliver the urban fabric as pretty stagnant, by putting the morphological background of sociocultural space at the forefront of the research approach and investigating how the situation faced of sociocultural provisions have altered. The study investigated axial analysis and visibility graph analysis through 12 attributes. The focus of this study is on the connection of Bag Bazar Street's growth in Kolkata. There are six stages to evolution. To compare the links, amalgamation of a qualitative study with attributes derived by various spatial syntactic-based studies. The Space Syntax aids in the proper understanding of Bag Bazar street's development while also generating any assumptions. Space Syntax analysis aims to provide a credible indicator for the evolution of space from a heritage city for digital archiving. Finally, the development of Bag Bazar Street's attributes changes throughout time (1746–2020). From 1985 to 2020, Bag Bazaar Street had the highest transformation rate.

References

1. Bagbazar street in 2020, Googleearth.com. Accessed 1 October, 2021
2. Al-Sayed K, Turner A, Hillier B, Iida S, Penn A (2014) Space syntax methodology. Bartlett school of architecture. UCL, London, UK
3. Anderson WP, Kanaroglou PS, Miller EJ (1996) Urban form, energy and the environment: a review of issues, evidence and policy. *Urban Stud* 33(1):7–35
4. Baillie W (1792) Plan of calcutta. <https://www.wdl.org/en/item/16800/view/1/44/>. Accessed 3 October, 2021
5. Bartholomew JG (1931) The imperial gazetteer of India, new (revised) edition, published under the authority of the government of India. <https://archive.org/details/imperialgazettee05grea>. Accessed 1 October, 2021
6. Cannadine D, Johnson J, Pooley C (1982) Residential differentiation in nineteenth-century towns: from shapes on the ground to shapes in society. In: *The structure of nineteenth century cities*, pp 235–51
7. Duan J, Hillier B et al (2007) Urban space
8. Dyos H (1966) The growth of cities in the nineteenth century: a review of some recent writing
9. Gibson JJ (1979) The theory of affordances. The ecological approach to visual perception
10. Hillier B (1999) Space as a paradigm for understanding strongly relational systems. *Proceedings of the space syntax second international symposium*, vol 2, pp 56–1
11. Hillier B (1973) In defense of space. *RIBA J (Royal Institute of British Architects Journal)*, 539–544
12. Hillier B (1996) Cities as movement economies. *Urban Des Int* 1(1):41–60
13. Hillier B, Hanson J (1989) The social logic of space. Cambridge University Press
14. Hillier B, Hanson J, Peponis J (1984) What do we mean by building function? E and FN Spon Ltd
15. Hillier B, Leaman A, Stansall P, Bedford M (1976) Space syntax. *Environ Plan B Plan Des* 3(2):147–185
16. Josna Raphael P, Kasthurba A (2015) Understanding the cultural influence through space syntax on the spatial configuration of temple towns of Kerala, India. *J Civ Eng Environ Technol* 2(5):461–468
17. Koohsari MJ, Sugiyama T, Mavoia S, Villanueva K, Badland H, Giles-Corti B, Owen N (2016) Street network measures and adults' walking for transport: application of space syntax. *Health Place* 38:89–95
18. Losty JP (1990) Calcutta: city of palaces. A survey of the city in the days of the East India Company (1690–1858)
19. Morris R (2000) The industrial town. *The English Urban Landscape*, vol 198
20. O'Brien J, Griffiths S (2017) Relating urban morphologies to movement potentials over time: a diachronic study with space syntax of liverpool, UK. In: *Proceedings of the 11th space syntax symposium*, vol 11. Instituto Superior Técnico, pp 98–1
21. Penn A (2020) Exploring the frontiers of space: bill hillier (1937–2019)
22. Penn A, Turner A (2003) Space layout affects search efficiency for agents with vision. In: *Proceedings 4th international space syntax symposium London*. <http://www.spacesyntax.net/SSS4>
23. Pooley C (2000) Patterns on the ground: urban form, residential structure and the social construction of space
24. Pooley CG (1977) The residential segregation of migrant communities in mid-victorian liverpool. *Trans Inst Br Geogr*, 364–382
25. Santilli D, D'Apuzzo M, Evangelisti A, Nicolosi V (2021) Towards sustainability: new tools for planning urban pedestrian mobility. *Sustainability* 13(16):9371
26. Schwab W (1992) Urban sociology. *Engelwood cliffs*
27. Shokouhi M (2013) Legible cities: the role of visual clues and pathway configuration in legibility of cities

28. Tao Y (2012) Digital city and space syntax: a digital planning approach. *Planners* 28(4):24–29
29. Turner A (2001) Depthmap: a program to perform visibility graph analysis. In: *Proceedings of the 3rd international symposium on space syntax*, vol 31. Citeseer , pp 31–12
30. Turner A, Doxa M, O'sullivan D, Penn A (2001) From isovists to visibility graphs: a methodology for the analysis of architectural space. *Environ Plan B Plan Des* 28(1):103–121
31. vectorstock: India map (2020). <https://cdn3.vectorstock.com/i/1000x1000/35/97/india-country-map-west-bengal-state-template-vector-29083597.jpg>. Accessed 14 October, 2021
32. Volchenkov D, Blanchard P (2008) Scaling and universality in city space syntax: between zipf and matthew. *Phys A Stat Mech Appl* 387(10):2353–2364
33. Wikipedia: bagbazar map (2007). <https://en.wikipedia.org/wiki/Bagbazar>. Accessed 3 October, 2021

Mapping Archaeological Remains of 14th Century Fort of Jahanpanah Using Geospatial Analysis



Gaurav Kumar Pal and M. B. Rajani

1 Introduction

Delhi has been the centre of political history for more than a thousand years. The present Delhi is an amalgamation of seven historical cities: Qila Rai Pithora (QPR) (extension of Lal Kot), Siri, Tughlaqabad, Jahanpanah, Firozabad, Dinpanah (Purana Qila), Shahjahanabad [1]. All the seven cities were fortified settlements; remains of many of them still exist. Few of the forts have lost their contours owing to the growth and rapid urbanization of Delhi after India gained independence. The fortifications like Tughlaqabad, Firozabad and Purana Qila have passed the test of time and stand resolutely under the protected eyes of ASI (Archaeological Survey of India) [2]. Whereas only parts of QPR, Siri, and Shahjahanabad are protected but large portions still exist as unprotected remains and encroached spaces and the contours of Jahanpanah's fort are largely obscured.

The present study is focused on the fourth historical city, Jahanpanah. It was built in 1327 AD by Muhammad Bin Tughlaq of the Tughlaq Dynasty (1320–1413 AD), the successor of Ghiyasuddin Tughlaq who built Tughlaqabad. Muhammad Bin Tughlaq followed the Tughlaq dynasty architecture with the construction of Daulatabad Fort in Daulatabad, Maharashtra and Jahanpanah in Delhi. After the reign of Ghiyasuddin Tughlaq, Muhammad Tughlaq first shifted his capital to Daulatabad but later came back to Delhi and enclosed QPR and Siri Fort with two fortification walls [3] (see Figs. 1 and 2). This enclosure came to be known as Jahanpanah. Siri Fort is towards the north-east of Jahanpanah while QPR is at the south-west. Along the southern part of the fortification, an irrigation structure known as Satpula (a weir creating a reservoir directing water from the south into the fort boundary) was constructed at

G. K. Pal (✉) · M. B. Rajani
National Institute of Advanced Studies, IISc Campus, Bangalore, India
e-mail: palgaurav@nias.res.in

the same time in order to regulate water for agriculture [4]. The Satpula Bridge is still intact, but the same cannot be said for the entire fortification wall of Jahanpanah [5]. The remains of the wall lay in isolated patches in the urban sprawl of South Delhi covering the areas of IIT (Indian Institute of Technology), Khirki Village, Panchsheel Park, Malviya Nagar, and Adchini [6]. However, these references are not enough to discern the exact contours of the fortification wall and hence there is no clear evidence for the exact locations of the Jahanpanah walls. Jahanpanah remains had only one inscription in the ASI protected monuments list, the bastion, where a part of the Jahanpanah wall meets QPR's [7] (see Fig. 2).

This paper tries to identify the exact location of the walls of Jahanpanah and map the accessible remains using geospatial data and analysis. Remote Sensing (RS) and Geographical Information System (GIS) has been increasingly used now in the field of archaeology and for mapping heritage sites [8]. For this study, old maps of Delhi and Corona Satellite images of 1965 were used to analyse and identify



Fig. 1 Map, The seven cities (1867)



Fig. 2 Overlay of georeferenced fort walls of south Delhi; see Table 1 for numbered locations

the exact location of where the fortifications would have been, and then through analysis of multi-date satellite images of recent years specific locations (unbuilt patch like green areas, parks) which potentially still may have some remains were selected. Subsequently, field visits to the selected location can be carried out for making ground observations and geotagging the feature found along the erstwhile fortification walls.

2 Methodology

The first step was to find reliable spatial information (location and shape) of the Jahanpanah fort. This was achieved through finding old maps from archival sources and digital libraries. The maps of the colonial period are preferred due to their accuracy in scale compared to earlier maps. A map of The Seven Cities (1867) [9] was found which marks the fortifications of the seven historical settlements of Delhi (Fig. 1). In addition Corona Satellite image of 1965 was accessed in order to identify remains of the Jahanpanah wall as this showed the view of the landscape before rapid urbanization occurred in Delhi [10]. Corona Satellites were used by the US Spy Intelligence (1960–1972), under the name Discoverer Program for the public. In 1995, these images were declassified for global change research. Map of Seven Cities of Delhi and the Corona Satellite image were georeferenced with the present Google Earth image (5th June 2021) of Delhi. Georeferencing Corona images is relatively

Table 1 Monuments inside Jahanpanah

S. no (corresponding to Fig. 2)	Monument	Period
1	Bijay mandal	Early 14th c
2	Begumpur mosque	Mid. 14th c
3	Serai shahi mahal	Early 16th c. (not clear)
4	Kharbuze ka gumbad	Before 1397
5	Lal gumbad	1397
6	Khirki mosque	Late 14th c
7	Satpula bridge	1343

easier than old maps as features are identifiable with current Google Earth Imagery. The shapes of road intersections are readily discernible as compared to maps before the nineteenth century.

A total of 32 GCP points (Ground Control Points) were taken to georeference The Seven Cities (1867) in Thin Plate Spline resampling with a mean error of 0.0024. Thin Plate Spline transformation was preferred over different Polynomial transformations as it gives localised warping than complete image warping as in Polynomial methods. The distortions in the latter gave a mean error of more than 40, which doesn't give an ideal accuracy for the features to be overlaid in the current Google map. After georeferencing, these maps were overlaid on the current Google Earth image in order to get the exact location of the Jahanpanah fort wall (Fig. 2). Monuments which lay inside the Jahanpanah fort have also been marked, where Bijay Mandal shows the possible citadel area of Jahanpanah during Mohammad Bin Tughlaq's reign [11] (Table 1). Field work has to be carried out on the possible remains of the wall as seen on different timelines on Google Earth.

Map features: The map, The Seven Cities (1867) was chosen as it contains the layout of all the seven cities which makes it easier to study Jahanpanah as it has multiple features that can be correlated. This map has fewer details as compared to other contemporary maps, which show other historical cities of Delhi in greater detail. However, this map had more details on the contours of the fortification of Jahanpanah. Old Delhi (Lal Kot and QPR area), Siri, Tughlaqabad and Jahanpanah are prominently marked whereas the City of Sher Shah and Ferozabad are coincided by dashed outlines. Directional orientation of the map is marked indicating a North-South alignment with scale. The map demarcates the landscapes of Delhi showcasing Aravalli ridges on the north-west and west of Lal Kot and QPR and the River Yamuna on the east. Two medieval routes are marked from Shahjahanabad: one from Ajmeri Gate to Mehrauli via Jahanpanah and QPR; another from Delhi Gate to Mathura by the route passing through the City of Sher Shah and detours to Tughlakabad. Significant monuments, mosques, temples, tombs and places are labelled, many of which fall inside the boundaries of the seven walled cities, thereby adding to the heritage value of these cities. Drainage network of the city is seen on the map where one of the channels is going through Satpula Bridge and further to the north-east. It

clearly marks the railway lines that were laid out in the colonial period. Although the map doesn't feature any legends, the labels and symbols used are self-explanatory.

2.1 Analyses of Geospatial Data

Overlaying the georeferenced 1867 map of The Seven Cities on Google Earth showed exact locations of the possible remains of the fort wall. The wall on the eastern side goes through vegetative land via Jahanpanah City Forest and meets Siri Fort crossing the Chirag Delhi drainage channel which goes till Satpula Bridge. As evident around the southern fortwall, the Press Enclave Marg road has emerged with time retaining the shape of the past wall [12]. It also remains consistent and accurate where the wall of QPR meets Jahanpanah as recorded on ASI Bhuvan [2]. The western portion of the fort wall goes adjacent with the moat along the northern boundary of QRP.

The findings made through overlaying georeferenced Corona Satellite images are worth noting. On close analysis, a circular shape of bastion can be observed where the eastern wall meets the southern wall of Jahanpanah. There are many semicircular shapes indicating bastions on the southern wall which can be observed in the Corona image (Fig. 3). There is a clear difference in the remains of the fort wall in the 1965 image and the present Google Earth image of Delhi, testifying the fact that urbanization with little regard to heritage sites have engulfed many portions of the fort remains.

2.2 Geospatial Studies of Archaeological Sites

The impact of such a geospatial database on archaeological sites and remains shows a clear picture of the past and present land use. Use of GIS studies adds to the historical data as the extent of the study can be analysed further, even where textual data suggests lesser information. The present example of Jahanpanah where its remains have further deteriorated can be seen both in Corona satellite images and current Google Earth images. However, such geospatial studies of fort extents and monuments of different regions can add to the heritage repository and open up new avenues for authorities to use this data in a way which can be added to the city development keeping ASI's prescribed protection boundaries intact. Just like in the case of Jahanpanah, there are several sites which have gone missing due to construction activities, largely due to a lack of spatial data on the context of the site. One similar case study is that of Chikkajala Fort in Karnataka, where unprotected sites of the fort were demolished for the construction of NH-7, clearly seen in the Google Earth imagery from 2004–2018 [13].

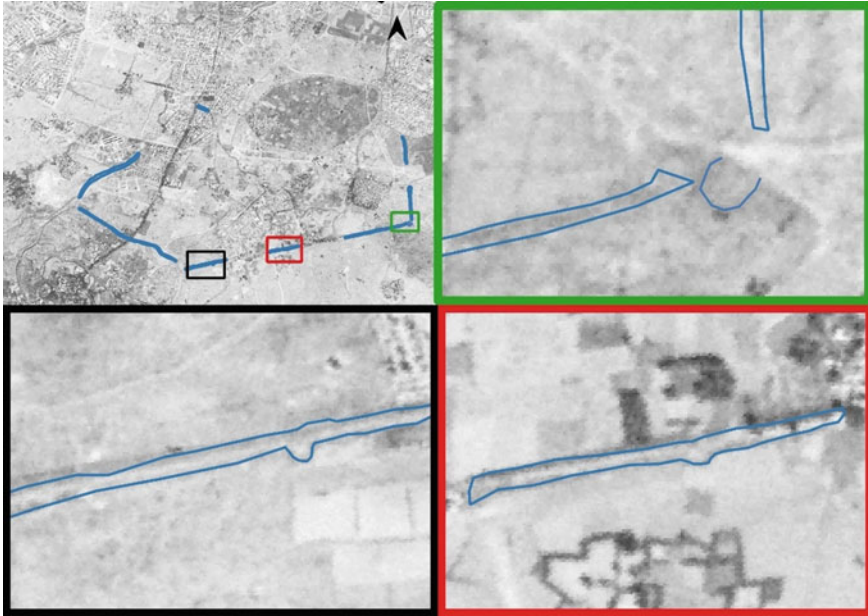


Fig. 3 Remains of Jahanpanah wall identified in corona satellite image (1965)

3 Conclusion

The geospatial analysis of the Jahanpanah wall throws light on new avenues for discovery. Use of old maps (of Colonial period) in today's spatial context offers new insight to the historical information. There are maps available which show drainage channels flowing south to north at Satpula Bridge, highlighting the emphasis on hydraulic structures for irrigation during Tughlaq and later periods [14]. Such channels have been in use in later dynasties as well, which can be further analysed using geospatial tools. This paper marks the fort line of Jahanpanah which is testified in the Corona image. It offers exact locations of Jahanpanah fort remains, which can be accessed for ground truthing on the field. The remains and mounds which can be found using this study will further add to the scholarship on the seven historical cities of Delhi.

Acknowledgements The research reported in this paper was conducted as part of a project funded by the Indian Space Research Organization, Department of Space, Government of India, under the RESPOND Scheme. The authors thank ISRO for the same.

References

1. Delhi-a Heritage city. UNESCO World Heritage Centre (2012). <https://whc.unesco.org/en/tenativelists/5743/>. Last Accessed 18 Sept 2021
2. ASI BHUVAN https://bhuvan-app1.nrsc.gov.in/culture_monuments/
3. Singh P (1971) The ninth Delhi. *J R Soc Arts R Soc Encourag Arts Manuf Commer* 119(5179):461–75. <http://www.jstor.org/stable/41370745>
4. Kumar R (1989) Irrigation technology in medieval india: a study of satpula weir. In: *Proceedings of the Indian history congress*, vol 50, pp 850–54. <http://www.jstor.org/stable/44146147>
5. https://bhuvan-app1.nrsc.gov.in/culture_monuments/
6. Sharma YD (2001) *Delhi and its neighbourhood*. New Delhi: archaeological survey of india. 25:7
7. https://web.archive.org/web/20170320190049/http://ignca.nic.in/img_0002_as_delhi.htm#. http://ignca.nic.in/img_0002_as_delhi.htm#
8. Rajangam K, Rajani MB (2017) Applications of geospatial technology in the management of cultural heritage sites – potentials and challenges for the Indian region. *Current Science*, Current Science Association, 113:10 pp 1948–60. <http://www.jstor.org/stable/26494844>
9. Guerrieri PM (1867) *The seven cities*, Maps of Delhi, Delhi. 17 2017
10. USGS <https://earthexplorer.usgs.gov/>
11. Peck L Delhi (2005) *A thousand years of building*. Jahanpanah. New Delhi, Roli Books Pvt Ltd pp 58–69
12. Rajani MB (2020) *Springer remote sensing/photogrammetry patterns in past settlements: geospatial analysis of imprints of cultural heritage on landscapes*. Singapore, Springer. p 28
13. Rajani MB (2020) *Springer remote sensing/photogrammetry patterns in poast settlements: geospatial analysis of imprints of cultural heritage on landscapes*. Singapore, Springer. p 148
14. Wadhawan PS (1988) A lesser known monument of the 14th century. *Proceedings of the indian history congress*, 49:684–695. <http://www.jstor.org/stable/44148473>

Spatial Analysis and 3d Mapping Historic Landscapes—Implications of Adopting an Integrated Approach in Simulation and Visualization of Landscapes



Mythrayi Harshavardhan 

1 Introduction

With the advent of technology, digital platforms have expanded, and new tools have paved the way for cutting edge development in many fields, including heritage conservation and management UNESCO emphasizes putting more efforts “to protect the world’s cultural and natural heritage”. This action plan comes in the wake of the Sustainable development goals (SDGs) for 2030 [1]. Cultural heritage has a socio-economic and political impact, and it is crucial to preserve it. Both tangible and intangible heritage fortify cultural diversity, local identity and community values [2]. Rapid urbanization and industrialization have contributed to the destruction and loss of built heritage, thereby posing a big challenge to preserve them. It has become vital to document heritage sites using efficient methods like digitizing data. The importance of having a national database for Heritage sites has been demonstrated by [3].

This paper focuses on different tools and approaches used to create 3D visualization of historical landscapes and architectural features embedded in them for a few heritage sites at Badami, Karnataka, India. GIS software like QGIS and ERDAS LPS Photogrammetry help in spatial analysis using multiple methods that include spatial interpolation, data exploration and overlay analysis which can be integrated with other modelling software for better results. The heritage sites explored in this paper are *Yellamma Temple*, *Upper Arali Tirtha* and an aqueduct in Badami.

The documentation of these sites has been carried out using the following approaches: (1) 3D models of the landscape were created using stereoscopic satellite images, and (2) 3D architectural models were made using two methods: (a) measure

M. Harshavardhan (✉)

School of Humanities, National Institute of Advanced Studies, IISc Campus, Bengaluru, India
e-mail: mythrayi.h@nias.res.in

drawings on AutoCAD and Google Sketch Up, and (b) processing digital photographs using Photogrammetric software like Agisoft Metashape. The methods adopted are both spatially (can be adopted anywhere and for any scale) and thematically (can be used for various types of built structures and forms) relevant. This study focuses on using specific digital platforms for different contexts to create a database of 3D landscapes, which can further be used by a variety of professionals in varied fields. The results from the present study find application in varied fields such as heritage conservation and management, urban planning, archaeology, and other related fields. There is a burgeoning literature on the subject. However, there is a dearth of information on specific methods to document unique sites and landscapes. The availability of resources coupled with lacunae in the knowledge of appropriate technology for data acquisition for heritage and archaeological documentation is a major challenge to overcome.

There is a huge potential for digitizing heritage structures and contributing to the database, which would lead to efficient information dissemination. However, there is still a paucity of information on the approach adopted for data acquisition, investigation, and documentation of heritage sites and structures. Some issues that can crop up while using digital platforms for documentation are- limited availability, reliability, accessibility resources, and the technical know-how, which can often limit the output. There is also a need to utilize the appropriate tools and techniques to get accurate results and interoperability of the data with other platforms used by professionals working at the interplay of heritage management and digital technologies [2].

2 Digitizing Heritage—A Brief Overview

Digital technology discussed in this paper implies the tools, devices, systems and resources used to collect, analyze and process spatial data. Digital technology helps in converting existing resources into a virtual analogue format. Spatial data discussed in this paper analyses location data attributes interrelated with geographical data. Spatial analysis can deliver more organized and suitable data to the end-user [4]. GIS is a platform to record, organize, retrieve and analyze spatial data. GIS tools have been vital in the spatial and temporal analysis of data by enabling the overlay of various attributes and helping cohesive analysis and visualization. Software like QGIS and ArcGIS has been a base platform to perform various image processing techniques and data interpretation to study archaeological and heritage sites. This section outlines a general approach adopted to analyze and visualize the data. This study explores three different sites to elucidate the different methods adopted for documenting these sites.

Historical features on the landscape can be traced from remote sensing data and prove useful in exploring heritage sites [5]. The availability of satellite images, digital maps, and elevation models has equipped one with the toolkit to analyze and document historic landscapes. 3D simulation and visualization of landscapes help perceive the study areas in greater depth and provide multiple dimensions to the

data layers. Digital terrain models prove useful in analyzing slope, relief, aspect ratio, etc. Yet another advantage with the virtual 3D models is the ability to visualize areas inaccessible on foot or require closer investigation when one is not on the field [6].

The case studies used in this paper adopt different integrated techniques for effective documentation of the heritage sites. Spatial data can be acquired by digitizing historic maps and other satellite imagery, collecting field survey points/Ground Control Points (GCPs), drone imagery, and generating Digital Elevation Models (DEMs). DEM can be obtained from processing spatial data and can enable users to visualize the terrain parameters such as slope, contours, aspect ratio and relief features. Furthermore, 3D simulation or reconstruction of landscapes using Photogrammetry can be done more accurately by adding finer details like monuments and other important built forms.

3 Context

3.1 *Badami and Its Immediate Environs*

In ancient times, Badami, known as *Vatapi*, was a former capital of the Early Chalukyas (5th- 8th Century) and was established by *Polekeshi I* [7]. The settlement of Badami is nestled amidst red sandstone cliffs in a horseshoe shape with a large reservoir called *Agastya Tirtha* on its eastern side (see Fig. 1). Badami exalted the status of a capital city till the reign of the *Rashtrakutas* [8]. However, the heritage structures found around the contemporary settlement of Badami are a testament to its glorious past. Badami invokes some popular images of the heritage structure strewn across the rocky landscape, which are the cave temples carved on the faces of the cliffs, the vestiges of the fortification that encloses some iconic temples and pavilion structures carved in stone. These well-known structures are- the *Upper Sivalaya*, the *lower Sivalaya*, *Malegitti Sivalaya*, Tipu's battery, granaries, and a *dargah*. At the lower level, where the settlement is located. There are another group of temples that are of importance in this landscape. The *Bhutanatha* group of temples is located at the eastern edge of *Agastya Tirtha and Yellamma Gudi*¹ on its western edge.

Some preliminary steps for analysis of Badami and its immediate environs involved using maps published by the Survey of India (SOI), which were used to identify some salient landscape features and built forms. Following are the maps used: (1) The 1932 map (Indexed 48 M/9) is a one-inch map or scaled to 1:63,360, first edition; (2) map surveyed in 1975 (Indexed 48 M/9), of 1:50,000 scale, first edition; (3) map updated in 2003–04, of 1:50,000 scale. Field surveys of identified locations, whose coordinates were obtained from geoportals (such as Google Earth and BHUVAN), and the SOI maps, at this stage of data collection. A few previously unknown and unclear sites were recorded using handheld GPS devices. These devices

¹ *Gudi* is a temple in the local language, Kannada.

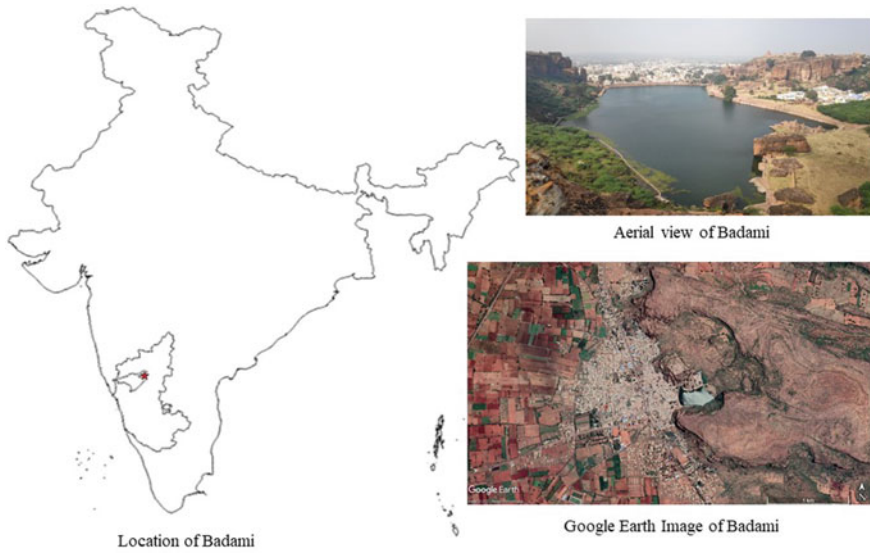


Fig. 1 Showing the location of Badami. *Figures compiled by author*

record the elevation, latitude and longitude, important to geotag locations. Terrain models were generated for hydrological analysis [9]. The present study extends the work by exploring methods to create digital architectural models and contextualizing and embedding them in landscape 3D models. This has been done for the three chosen heritage structures. The sites have slightly varying scales, and hence different methods have been applied to visualize the structures in the context of landscape on a digital platform. There is a multitude of sources available for satellite data. However, this paper discusses the historic imagery freely available on google earth and SOI maps, and the principal datasets used for this study were high-resolution images from CARTOSAT-1 (IRS-P5).

Thematic maps have been generated from the existing base maps. The layers of information include adding vector data (points, lines and polygons) as geoTIFF, .shp, .kmz or .kml files. The GIS platform provides flexibility to work on the available data on multiple digital platforms like Google Earth, QGIS and ArcGIS. It enables the many layers of spatial and temporal data integration and analysis. Some basic image processing was performed to integrate the layers of raster data. Two or more images are brought to a common map projection [10] and can be pinned together digitally using Ground Control Points (GCPs) in both images that are spread out evenly. Map projections result from mathematical calculations performed for projecting

the earth's curvature onto a flat surface. The study uses multiple images, georeferenced to World Geodetic System (WGS) 84 datum² and ellipsoid.³ The images are coordinated by geo-referencing them, locating the image on a map coordinate system by assigning latitude and longitude to each pixel. This process is done by adding Ground Control Points (GCP) (points for which geographical coordinates can be attributed) to the target image from sources such as Google Earth, Google Map and Bhuvan (sources from where corresponding coordinates can be acquired). Higher accuracy can be achieved with more GCPs that are evenly distributed. The images are then transformed by resampling techniques, where the size, orientation, and scale are modified accordingly. The transformation set used in this study is Nearest Neighbor, and the resampling techniques used are first-order and third-order polynomial transformations [5].

A Digital Elevation Model (DEM) is vital raster imagery to analyze the terrain and produce relief maps. Terrain parameters like slope, contours, and elevation can be studied. DEMs also prove useful in studying topography, geomorphological analysis, ortho-rectification of satellite imagery, and creating hydrology maps [12]. It has helped evaluate the unique terrain of Badami (see Fig. 2).

CARTOSAT-1 produces stereoscopic imagery of the earth in the visible region of the electromagnetic spectrum. The stereoscopic images are black and white and were taken by two panchromatic cameras with a spatial resolution of 2.5 m and a swath of 30 km. The two panchromatic cameras are mounted with a $+26^\circ$ (Fore) -5° (Aft) tilt, enabling the capture of two different images of the same area. The stereo pairs are acquired with a time difference of 50 s due to the cameras pointing in forwarding and backward directions. This feature of the satellite facilitates the generation of three-dimensional maps, Digital Elevation Models (DEMs), and Digital Terrain Models (DTMs) that find application in various fields [7–14].

The CARTOSAT-1 stereo pairs have helped visualize the topography of Badami and its undulating terrain. With the CARTOSAT stereo pair images, it is possible to generate anaglyphs. Anaglyphs also help visualize the terrain but without the need to generate DEM. However, the anaglyph images must be viewed through a pair of anaglyph glasses with two different colours, one for each eye (see Fig. 3). The stereo images are superimposed to produce depth, where the left image is projected in red and the right image in cyan (green and blue), creating an RGB colour composite image. The anaglyph glasses enable viewing the stereoscopic images integrated without any glitch. In the anaglyphs generated from CARTOSAT-1 images, the north is rotated to the left as the images are captured by the satellite, which orbits in the north–south direction. An anaglyph image does not support viewing 3D from different angles, distances, and heights in an anaglyph image. The anaglyph image can be viewed in 3D only from the top-down nadir direction [12].

² Datum of a location is a set of reference points i.e., latitude, longitude and height calculated for that particular location on the earth's surface. There are local and regional datums used for certain regions [11].

³ The earth's surface is not a perfect ellipsoid, hence mathematical calculations have been developed for different ellipsoids to find a fit that would help in establishing coordinate points for different locations [11].

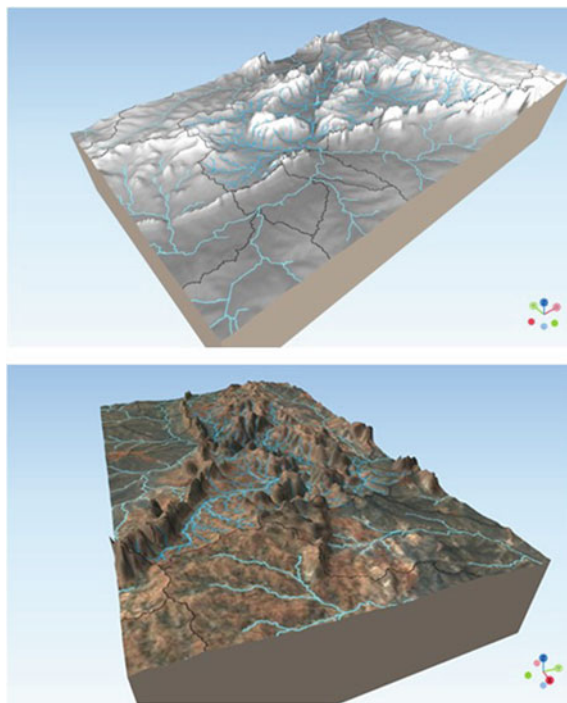


Fig. 2 3D simulation of the landscape of Badami and its immediate environs. *Source figures compiled by author, and the image on the right is google earth imagery draped over DEM*

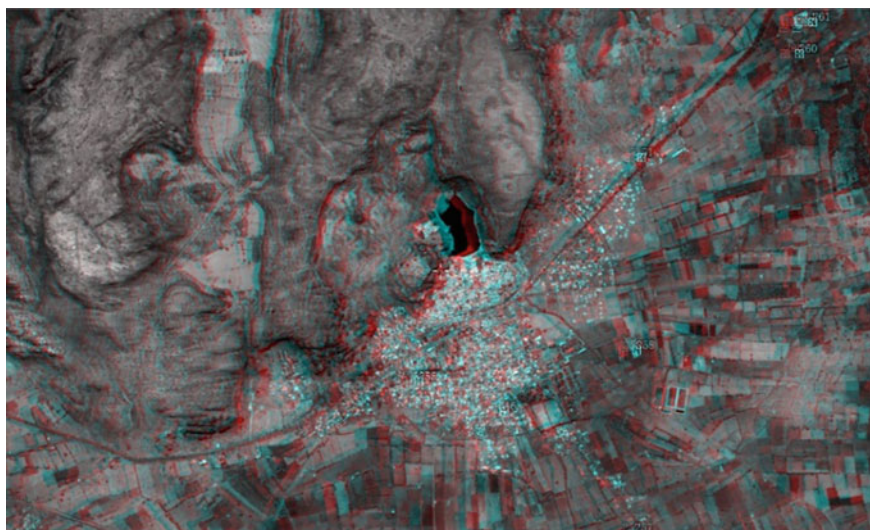


Fig. 3 Anaglyph produced with CARTOSAT-1 stereo pair. *Source author's own*

This paper explores sites from a closer perspective, and the virtual reconstruction of the landscape has been accomplished primarily using the QGIS and ERDAS Imagine Photogrammetry. The architectural monuments and other built forms have been modelled using the Agisoft Metashape, Google SketchUp, and AutoCAD. The built structures have then been incorporated into the landscape model in the Virtual GIS module of ERDAS Imagine software for cohesive viewing of the historic landscape in its entirety. Other sophisticated methods have been adopted to generate accurate building models using Light Detection and Ranging (LIDAR), where data is recorded with the help of a pulsed laser and an airborne or ground-based system for generating precise three-dimensional models of sites [15]. However, in the absence of availability of LiDAR data, this study uses Photogrammetry techniques to generate 3D models of built structures for visualization purposes. Sites 1 and 2 use similar methods for the documentation, but the typology of built forms and scales vary.

ERDAS Imagine software allows the transformation of raw satellite data into usable digital formats for mapping, and raster data analysis, which can further be used in 3D visualization. The acquired raw geospatial data can be processed and analyzed using various processes such as analytical triangulation, terrain model generation, ortho-photo production and producing ortho-mosaics. The various workflows used on this platform are very flexible and help produce accurate results. It supports an interactive/dynamic platform for virtual fly-throughs and rendering of 3D models. This technology uses non-intrusive and secure methods from data acquisition to generation.

This study uses Agisoft Metashape to document heritage structures and other built forms. The software generates highly detailed and photorealistic models by processing images into 3D spatial data in a dense point cloud, textured polygonal models, georeferenced ortho-mosaics and Digital Surface Models (DSM)/Digital Terrain Models (DTM). The linear workflow is intuitive and easily provides a user-friendly platform to process models. Buildings can be visualized in their entirety or ruinous state; they can be reconstructed with exceptional details, and fly-throughs that enable virtual tours of heritage sites are possible.

An aerial LiDAR covering the landscape and buildings would have collected 3D data of all elements from one source. However, since this study uses satellite data for landscape topography and photographs for capturing buildings, we found much disparity in spatial resolution, scaling and 3-axis orientation of data from the two sources. This provided an opportunity to explore the challenges and find optimal solutions that are discussed below.

3.1.1 Site 1

Beyond the *Bhutanatha* temple complex, a trail leads to a site located on the summit of the hill, called *Arali Tirtha*. This site is marked by two shrines- one small cubical shrine made of sandstone blocks and a larger shrine in the form of a natural cavern with a rock overhang on one side with a spring (perennial water source) in front of it. This cavern is adorned with sculptures and carvings of many Hindu deities [16].

A few hundred meters uphill towards the east of this site is a cistern, the source of a stream that gathers water during a heavy spell of rain and flows to the edge of the cliff, splitting into two waterfalls called *Akka-Thangi* falls [9]. Further exploration of the site around the cistern led to finding a large temple complex (marked on the 1975 SOI map of the region- Indexed 48M9) with pillared mandapas in a ruinous state. *Arali Tirtha* has been extensively studied and documented by various scholars. However, the site uphill from *Arali tirtha- Upper Arali Tirtha* is lesser-known, and hence, this paper explores the methods used to document this site. *Upper Arali Tirtha* is located on a cliff summit that is difficult to access. Hence a digital model of landscape with the built form helped accurately capture and later analyse the setting (see Fig. 4).

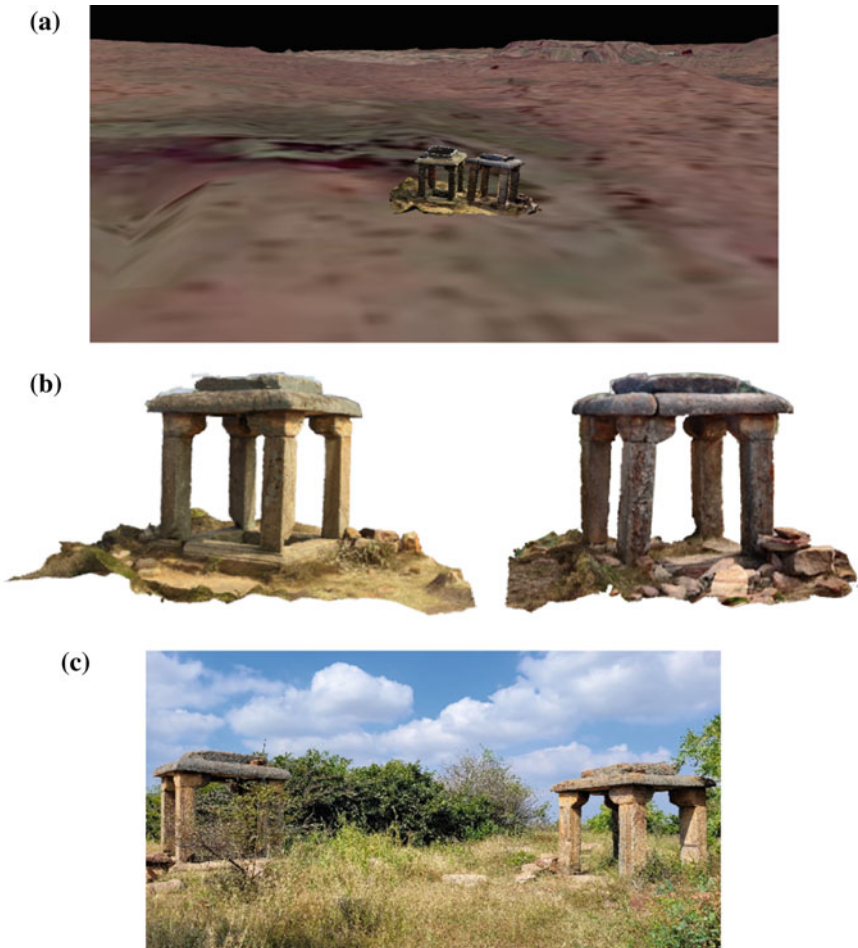


Fig. 4 *Upper Arali Tirtha* **a** Terrain model of the site with the built forms **b** 3D Models of the built forms **c** Field photograph of the structures. *Source author's own*

3.1.2 Site 2

Yellama Temple on the western banks of Agastya Tirtha is located on a bund. Evidence indicates that this temple was established in 1139 during the Later Chalukyan period [17]. This temple was originally dedicated to Vishnu, but now it enshrines a popular female deity. The ornate temple is approached through a collonaded open mandapa that leads to the main sanctuary with a tall shikhara rising above it (see Fig. 5). This structure has an ornate facade with complex architectural details. It was possible to recreate this model on a digital platform, capturing the intricate details.

3.1.3 Site 3

The former rulers located the former Chalukyan capital at a strategic locale due to the surrounding sandstone cliff's geographic advantage that provides natural defenses [7]. The semi-arid climatic conditions and the region's unique geology led to various types of water harvesting systems devised in the past to sustain the settlement [18, 19].

Harshavardhan and Suganya [20] establish the presence of an aqueduct that once connected a cistern in the north-eastern part of the fort and another cistern located a few hundred meters away in the same direction. The paper also dates the structure as possibly belonging to Vijayanagara Period and discusses the intricate water networks in the landscape, which were engineered in the past that formed a cascading system by connecting other water harvesting systems. A Badami map drafted in 1818 by John Jeffery O'Donnoghue, a British military engineer, marks the aqueduct and several tanks in the region [20, 13]. Figure 6 shows the location of this aqueduct with respect to the adaptation of the 1818 sketch of the Badami Fort and Pettah region marking important features (Image on the right). This 1818 map has some planimetric inaccuracies that may have occurred during the survey or reproduction of the map [8, 21].

4 3D Landscape Model Generation

The sites were ground validated from the points available on 1930 and 1975 SOI maps. The SOI maps and Google Earth imagery were georeferenced and overlaid on QGIS to analyze the context. Field points were added to the base maps, and vector layers of polygons were added to demarcate the study areas.

CARTOSAT-1 images were processed using a series of workflows that finally helped generate a DEM 3D simulation of the landscape. On the ERDAS IMAGINE Photogrammetry software, the image pairs are first placed together with tie points that can be automatically generated or with the help of Ground Control Points

(a)



(b)



Fig. 5 *Yellamma Temple* **a** Field Photograph **b** 3D Model of the built form. *Source author's own*

(GCPs) using the triangulation method.⁴ Once the images are triangulated, generating anaglyphs and DEMs with additional data processing is possible. Terrain editing, DTM extraction, and ortho-rectification of the processed images can also help achieve accurate results. Ortho-photos are created by minimising the distortion in the image pairs that can occur due to the topographic relief displacement, sensor orientation

⁴ A method by which a location of a point is determined by measuring angles formed between that point and the points on either side of a fixed baseline, at one end.

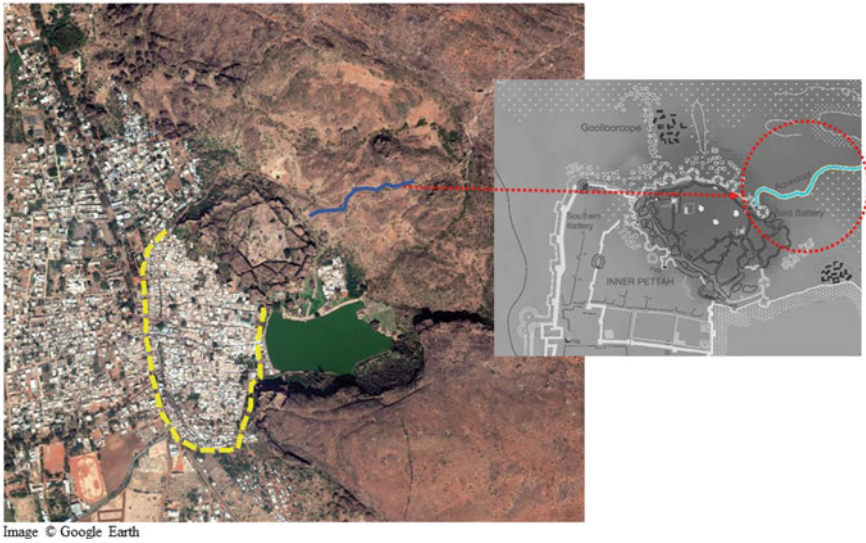


Fig. 6 Google imagery showing traces of aqueduct (left) and an adaptation of the 1818 map of Badami that marks the aqueduct (right) [13].⁵ Source figure's compiled by author

or other technical errors. The resulting images are planimetrically correct, where the ground objects are represented in their true X and Y positions. The terrain model is draped with raster images like Google Earth or ortho-images to add more details and help better visualize the terrain. The multiple layers used in Virtual GIS need to be reprojected to ensure they have the same coordinate reference system (CRS).

4.1 Building Model Generation of Sites 1 and 2

The architectural models of *Upper Arali Tirtha* and *Yellamma Temple* were created using Metashape. The models were generated by importing a series of photographs (for this study, photographs were taken from smartphones) of the built forms on the software. With the help of a linear workflow, one can convert the photographs into point cloud data, which further renders a photorealistic model with accurate details and textures (see Figs. 4 and 5). These models are to scale, which, when imported onto the terrain model (with file extensions of 0.3ds and .dxf) on the ERDAS Imagine platform, requires the user to adjust various parameters like location, scale, and orientation (see Fig. 7). This must be done manually to situate the model correctly on the landscape. These parameters can be modified using the model attributes of Pitch (Lateral axis), Yaw (Vertical axis), and Roll (Longitudinal axis). Elevation

⁵ <https://artsandculture.google.com/asset/badami-karnataka-john-jeffery-o-donnoghue/5gH1cgapvbkzg?hl=en>

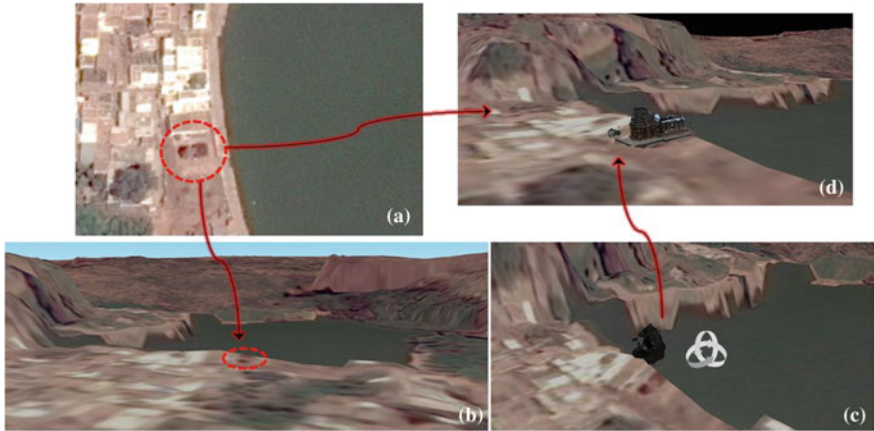


Fig. 7 *Yellamma Temple* **a** Aerial view showing the location of the built form on Google satellite image **b** Perspective View Badami landscape marking the location of the built form **c** Architectural model imported on terrain model in ERDAS Imagine platform before correction of 3-axis orientation parameters **d** Terrain model of the site with the built form. *Source author's own*

parameters of above ground level (AGL) and above sea level (ASL) should also be altered to locate the models correctly on the terrain model. Figure 7 illustrates the process with the *Yellamma Temple* model (site 2).

4.2 Building Model Generation of Site 3:

The method used to generate a 3D model of site 3 (a 900 m long and 600 mm wide linear structure) differs from sites 1 and 2 due to the different nature and scale of the site. The scale of this site 3 is relatively larger than the other two sites. Unlike the other two sites, it was not ideal to use Metashape to generate an entire aqueduct model as it is a linear structure spanning a few hundred meters. However, parts of the structure can be modelled using the software. Hence, the techniques used to document this site included DEM generation and developing a 3D model of the aqueduct using Google Sketchup and AutoCAD.

The 1818 map was georeferenced and overlaid with a georeferenced historical google earth imagery of different periods on QGIS to compare and identify features on the two maps. This was done to factor in the seasonal changes, which revealed the presence of water bodies in some seasons. This process led to tracing the remains of the aqueduct, which subsequently led to ground validating the structure and recording a few field points and elevation at different points along the length of the aqueduct. This data was essential in generating a profile of the aqueduct (see Fig. 8). A conjectural model of the aqueduct was drawn out in AutoCAD. The drawing was

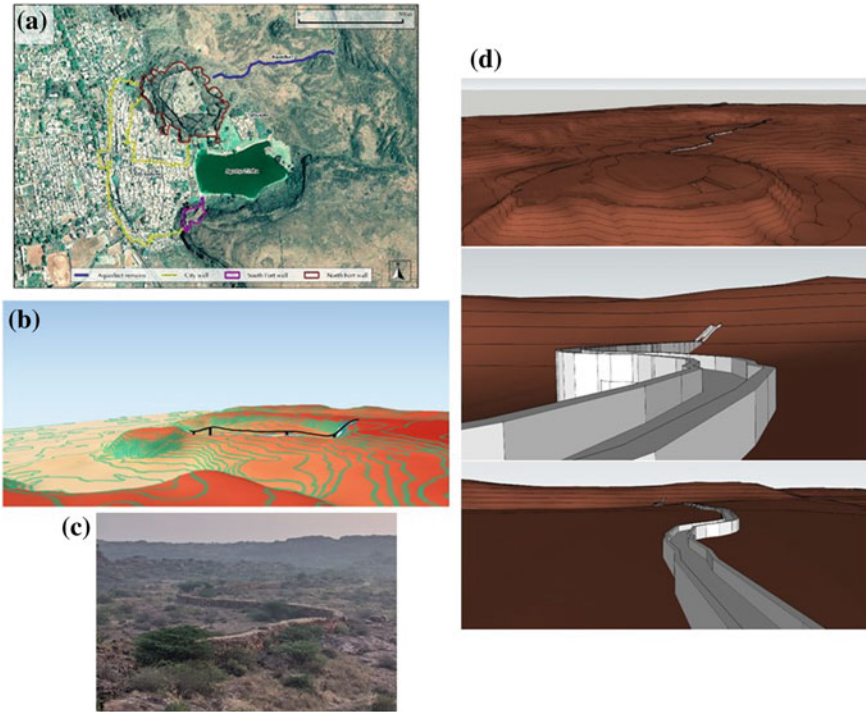


Fig. 8 Model of terrain and Aqeduct, Badami **a** Google earth map showing the location of the aqeduct *Figures Compiled by the Author. Source of the base map- Google Earth* **b** Simulation of the landscape showing the elevation of the aqeduct **c** A view of the remains of the aqeduct structure on the ground, photograph taken from the North Fort **d** Perspective views of the model of the aqeduct

then imported into Google SketchUp (in the form of .dwg and .dxf files), where a schematic model of the aqeduct was generated.

SRTM DEM (1 ARC Second) was acquired from the USGS Earth Explorer website. And this allows the users to download a DEM of 30 m resolution. From the DEM, it is possible to create contours, which can then be imported into SketchUp to create a contour model on which the aqeduct structure would be located (see Fig. 8). The resultant schematic model enables one to study the landscape, although this method is a more manual way of creating the terrain model and the built form.

5 Concluding Remarks

This study uses different approaches to create 3D models for simulation and visualization of historical landscapes. Satellite imagery and remote sensing data have proven useful in generating terrain models for a region like Badami with dramatic

landscapes and undulating topography. The methods used in this study have their advantages and drawbacks. The main advantage of the methods used in this is satellite imagery can depict heritage and archaeological sites synoptically, giving a clearer understanding of the spatial extent of a region. The satellite images obtained are often for relatively larger areas where one can better analyze the site and its surroundings. Through the acquired satellite data, it is possible to analyze slope, aspect, contours, elevation, relief analysis, and drainage patterns. The tools are flexible, user-friendly and simplify complex processing functions. It can be combined or replaced with other techniques. The methods used to survey and document sites are non-intrusive and enable large data acquisition. A few drawbacks involve using high-end modelling software that is expensive for DEM generation and terrain modelling. However, there are a few alternative tools and freely available data to overcome this drawback. Limited resources and satellite imagery availability can have problems such as poor resolution or cloud cover that can inhibit clear viewing and can often prove to be a major challenge to overcome. Since this study discusses the reconstruction of terrain models and architectural models on separate modelling platforms of ERDAS Imagine and Agisoft Metashape, respectively, one might encounter a few difficulties, such as the location, scale, orientation, etc., while creating a consolidated model. Yet another disadvantage is the lack of technical knowledge and support to use appropriate methods for data acquisition and documentation of historic landscapes. A single method cannot help get the accuracy in the documentation of sites. It is imperative to combine different techniques for better results in digitizing heritage [22].

With heritage structures and historic landscapes undergoing tremendous stress because of various natural and anthropogenic factors, there is an urgent need for digital documentation. The application of digital technologies can help preserve and showcase heritage sites by creating a database that can be used in the future. This method of documentation is faster and more efficient than manual documentation techniques.

Acknowledgements The research reported in this paper was conducted as part of a project funded by the Department of Science and Technology, Government of India, under the scheme Indian Heritage in Digital Space. I thank the Principal Investigator of the project Dr M.B. Rajani at the National Institute of Advanced Studies, for her valuable inputs and discussions on the paper. I thank Ms Kuili Suganya for her insights about the site.

References

1. UNESCO (1972) Convention concerning the protection of the world cultural and natural heritage. Last accessed <http://whc.unesco.org/en/conventiontext/>
2. Trillo C, Aburamadan R, Mubaideen S, Salameen D, Chikomborero B, Makore N (2020) Towards a systematic approach to digital technologies for heritage conservation. insights from Jordan. *Preserv Digit Technol Cult* 49(4):121-138. Last accessed <https://doi.org/10.1515/pdtc-2020-0023>

3. Gupta E, Das S, Suganya K, Balan C, Kumar V, Rajani MB (2017) The need for a national archaeological database. *Curr Sci* 113(10)
4. Paramasivam CR, Venkatramanan S (2019) An introduction to various spatial analysis techniques, chapter 3. In MV Venkatramanan S (Eds) GIS and geostatistical techniques for groundwater science pp. 23–30. Last accessed <https://doi.org/10.1016/C2017-0-02667-8>
5. Gupta E, Rajani MB (2020) Geospatial analysis of historical cartographic data of kollam fort. *J Indian Soc Remote Sens*, 48: 1567–1581. <https://doi.org/10.1007/s12524-020-01181-w>
6. Rajani MB (2007) Bangalore from above: an archaeological overview. *Curr Sci* 93(10):1352–1353
7. Michell G (2011) Badami, Aihole, Pattadakal. Mumbai: JAICO Publishing House
8. Menon SM (2013) Treasure trove of temple architecture: lesser-known monuments of Badami, *Travel & flavors*, pp 84–87
9. Suganya K, Harshavardhan M, Rajani MB (2022) The significance of ancient water systems and the sacred groves in the landscape of badami, karnataka: a geospatial study. In: Dhyani MBS(Eds) Blue-Green infrastructure across asian countries, improving urban resilience and sustainability (p. (Chapter 17)). Springer Nature. Singapore
10. Robinson AH, Morisson JL, Muehrcke PC, Kimerling AJ, Guptill SC (2004) Elements of cartography, Wiley, INC. Singapore
11. Rajani MB (2020) Patterns in past settlements: geospatial analysis of imprints of cultural heritage on landscapes: Springer Nature
12. Mandyam BR, Patra SK (2009) Space observation for generating 3D perspective views and its implication to the study of the archaeological site of Badami in India. *J Cult Herit* 10(1):e20-e26. <https://doi.org/10.1016/j.culher.2009.08.003>
13. Nanda V, Johnson A (2015) Cosmology to cartography- a cultural journey of Indian maps: from the collections of kalakriti archives, Hyderabad and national museum. New Delhi, National museum
14. National Remote Sensing Centre. <https://www.nrsc.gov.in/> Last accessed <https://www.nrsc.gov.in/>
15. National ocean service. <https://oceanservice.noaa.gov>. Last access <https://oceanservice.noaa.gov/facts/lidar.html>
16. Nuess J (1993) Aralikatti- A forgotten sculptured cave near Badami. *Berliner indologische studien (BIS)* 7:173–206
17. Padigar SV (2012) Heritage series: badami. Department of archaeology, museums and heritage, Bangalore. (2012)
18. Joshi RV (1955) Pleistocene studies in the malaprabha basin. Poona; Dharwar: deccan college postgraduate and research institute; Karnataka University.
19. Agarwal A, Narain S (1997) Dying wisdom- rise fall and potential of india's traditional water harvesting systems. New Delhi: centre for science and environment
20. Harshavardhan M, Suganya K (2020) Water harvesting systems of the past- a case study of badami, India. National seminar on 'recent advances in geospatial technology & applications'. Dehradun: Indian institute of remote sensing, pp. 170–175. Last accessed <https://www.iirs.gov.in/national-seminar>
21. Davidson TE (1986) Computer correcting historical maps for archaeological use. *Historical Archaeology*, 20(2):27–37
22. Kovacs F (2015) Documentation of cultural heritage; techniques, potentials, and constraints. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W7. <https://doi.org/10.5194/isprsarchives-XL-5-W7-207-2015>

23. Bhatawdekar S, Jaiswal RK (2018) Cartography (high-resolution) observing system. Comprehensive remote sensing, 1. <https://www.sciencedirect.com/topics/earth-and-planetary-sciences/cartosat>
24. European space agency, <https://earth.esa.int>. Last accessed <https://earth.esa.int/eogateway/missions/irs-p5>
25. Satellite imaging corporation. <https://www.satimagingcorp.com>. Last access <https://www.satimagingcorp.com/satellite-sensors/other-satellite-sensors/cartosat-1/>

Medical Image Processing (MedImage)

HSADML: Hyper-Sphere Angular Deep Metric Based Learning for Brain Tumor Classification



Aman Verma and Vibhav Prakash Singh

1 Introduction

Brain tumors are a malicious accumulation of anomalously growing cells in the brain parenchyma and the regions in the vicinity. Early-stage diagnosis of brain tumors helps medical experts to take timely medication. Along with the diagnosis, it is of utmost importance that the correct class of the tumor is also identified. The task of brain tumor classification can be categorized as a three-class detection problem wherein the classes are Meningioma, Glioma and Pituitary Tumor. Computer Aided Diagnosis (CAD) based approaches have become prevalent in classification and more recently machine intelligence-based solutions are being employed to augment CAD frameworks. To be specific, deep learning-based methods have established dominance in the domain owing to their robust performances [1]. These approaches utilize deep neural network architectures such as deep CNN to capture complex nonlinear decision boundaries. Multiple attempts have been done to enhance classification performances, in [2] authors used a 3-stage approach to classify the grade of the tumors, first the brain tumors were segmented, then from the segmented images, data augmentation was done after which a CNN was employed for feature extraction cum classification. Transfer Learning approaches involving ImageNet [3] pre-trained models have been actively employed to achieve state-of-the-art performance [4]. To overcome overfitting GAN based brain MRI data augmentation was done

A. Verma (✉)

Department of Electronics and Communications, National Institute of Technology Raipur, Raipur, India

e-mail: aman.verma.nitr@gmail.com

V. P. Singh

Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology Allahabad, Prayagraj, India

e-mail: vibhav@mnnit.ac.in

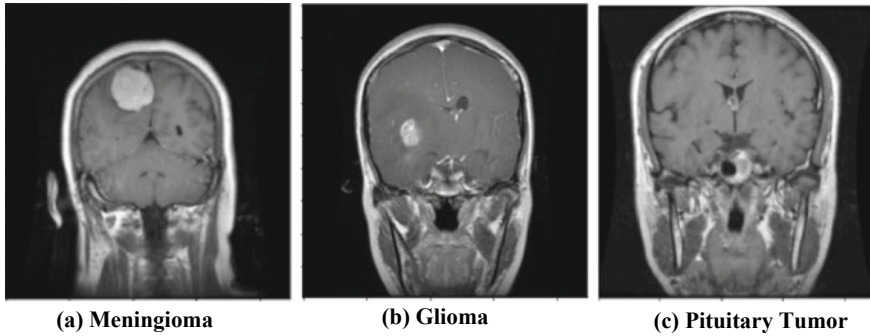


Fig. 1 MRI scan slices from each of the brain tumor types

in [5] while approaches involving Capsule Networks have been recently proposed [6]. BrainMRNet [7] has been recently introduced, it uses Attention Mechanism [8] to bring robustness in features. Even though there has been significant progress so far, there is a wide scope for improvements. Firstly, there is a need to improve the generalizability of the models and secondly, extraction of more prominent features is also essential. High inter-class similarity along with high-intra-class variability degrades the performance. Foremost cause of the same is that the visual morphology and view-of-capture of the MRI-slices among the classes are quite similar. Figure 1 depicts structural similarities between the three brain tumor types. Moreover, the major tumor-type differentiating information remains concealed with limited spatial space. Thus, this research attempts to overcome above mentioned problems and tries to improve feature discrimination to enhance the performance and generalization of brain tumor classification models.

Deep metric learning has been a natural choice in multiple domains [9–11] to enforce inter-class separability and intra-class similarity. Usually, SoftMax based classifications are performed but it is to be noted that SoftMax loss maximizes class distances so as to have features before the last fully connected layer linearly separable as it is a softened version of the max operator. This enforces low-inter-class separation which causes minimal intra-class similarity. Deep Metric Learning (DML) on the other hand metricizes the distance between training examples which makes examples belonging to the same class being clustered while the different ones moving towards respective class centers leading to inter-class separation. DML is applied via training deep learning models with distance aware losses. Triplet Loss [12] has been one of the most widely used DML loss, for any example(anchor) it samples a positive(same class) and negative(different-class) example, computes the distance between embedding of anchor and positive and anchor and negative, then it imposes a Euclidean margin between them to improve inter-class distances and intra-class closeness. Though it has been successful, selection of inadequate margin and sampling wrongly may lead to unstable training. Concepts of adaptive margins and hard-negative mining [13] have been introduced to alleviate the problems in prior but issue related sampling sustains. Other loss functions such as Center-Loss [14]

supervise SoftMax loss through another term that minimizes intra-class variance, but in such losses SoftMax loss empirically dominates. An advancement of triplet loss has been the Quadruplet Loss [15] which samples another negative with a target to set minimum inter-class distance greater than maximum intra-class distance, but again negative sampling remains an issue of concern. Angular Loss [16] improvised over the previous works by enforcing angular margin and hence utilizing gradients from all negative, positive and anchor, this was lacking in Euclidean margin-based triplet loss strategies but negative mining must be done here also.

To alleviate the shortcomings of this paper, proposed the HSADML framework—a hyper-sphere manifold metric-based learning for brain tumor classification. The proposed approach utilizes SphereFace Loss [17] which surpasses the tedious nature of triplet loss-based sampling and enforces angular separation using angular margin. To the best of knowledge this is the first work on brain tumor classification to utilize DML. Following is the summary of key contributions:

- HSADML Framework to enhance the generalizability of brain tumor classification models.
- HSADML framework uses SphereFace loss that alleviates the issue of triplet sampling but at the same time develops discriminative representation using angular margin -based Hyper-Spherical DML.
- The proposed approach achieves state-of-the-art results on a benchmark dataset. Extensive experimentation done to verify the methodology suggests the same.

This paper has been organized in four sections The current section gave an introduction and literature review. Proposed framework is explained in the next section, while in the third section—Experiments, Results and Discussions, experimental analysis has been done. Finally, the paper is concluded in Sect. 4 with future research dimensions.

2 Methodology

HSADML Framework has two major components first being the SphereFace Loss Function and the second being the backbone network as diagrammatically illustrated in Fig. 2. In this study, MobileNet [18] has been utilized as the backbone network with the reason being the lightweight nature of the same. All the components of the HSADML framework have been explained in subsequent subsections.

2.1 SphereFace Loss

It is visually challenging to identify the type of brain tumor mainly because of two reasons—(1) The affected region being quite small, at a large scale bears resemblance to all the three- classes. (2) Multi-view data appends morphological similarity which

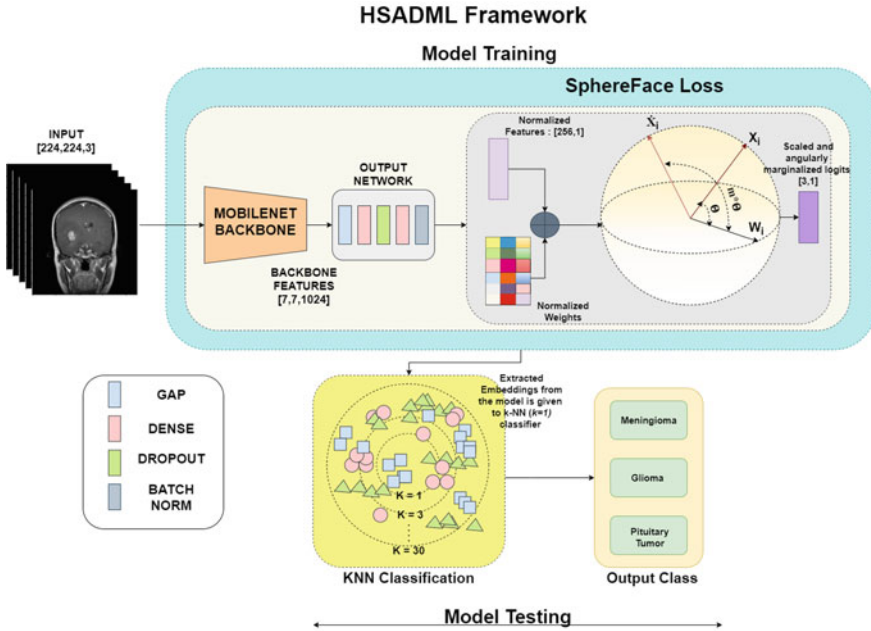


Fig. 2 HSADML framework has two phases—the training phase and the testing phase. In training phase, the deep learning model is trained via angular metric learning using SphereFace loss. While in testing phase k-NN ($k = 1$) based testing is done

further enforces inter-class compactness and intra-class discrepancy. SoftMax Loss function on the other hand when employed in classifiers generates linearly separable decision boundaries using the features of the last dense layer. Mathematically, the SoftMax Loss function is defined as follows:

$$\mathcal{L}_{Softmax} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{(W_{y_i}^T x_i)}}{e^{(W_{y_i}^T x_i)} + \sum_{j \neq i} e^{(W_{y_j}^T x_i)}} \quad (1)$$

Here, $x_i \in \mathbb{R}^d$ with d being the dimension of feature embedding. Let there be n classes $-[y_1, y_2, \dots, y_n]$ and the batch-size is N , $W \in \mathbb{R}^{d \times N}$ represents the weights while $W_{y_i} \in \mathbb{R}^1 \times d$ is the weight column corresponding to class y_i . For the sake of brevity, the bias term hasn't been included. For SoftMax Loss linear-vector space is considered for classification where the class boundaries are bounded within a certain defined region, this restricts the generalizability of the model. Whereas, if the classification is done in angular some specific sector gets assigned to a particular class which makes the region of existence for that class unbounded within the sector. HSADML framework on the other hand utilizes SphereFace Loss Function to facilitate angular metric learning. SphereFace Loss projects the feature embeddings onto a hyper-sphere manifold by normalizing both weights and feature embeddings. Then,

it metricizes the angular distance between class centers 0 and the feature embeddings. By imposing an angular margin, it makes training harder but ensures intra-class compactness and inter-class variance. $W_{y_j}^T x_i$ defines the dot product between W_{y_j} and x_i , thus SphereFace Loss formulates following Euclidean to Angular Space Transformation— $\|W_{y_j}\| \|x_i\| \cos \theta_{y_j i}$. Here, $\theta_{y_j i}$ denotes the angle between W_{y_j} and x_i . The class center and the feature embeddings are L_2 normalized which makes the transformation dependent only on $\theta_{y_j i}$. The normalization step projects the embeddings on a d dimensional hyper-sphere. Finally, angular margin m is introduced to make training tougher but this results in formation intra-class clustered and inter-class separated embeddings. Formally, SphereFace Loss can be defined as follows:

$$\mathcal{L}_{SPF} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{\|s\| \Psi(\theta_{y_i})}}{e^{\|s\| \Psi(\theta_{y_i})} + \sum_{j \neq i} e^{\|s\| \cos(\theta_{y_j i})}} \tag{2}$$

Here,

$$\Psi(\theta_{y_i}) = (-1)^k \cos(m\theta_{y_i}) - 2k \tag{3}$$

$\|s\|$ is a scaling factor, $k \in [0, m - 1]$, $\theta_{y_j i} \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m} \right]$, $\theta_{y_j i} \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m} \right]$ and $m \geq 4$. Function $\Psi(\cdot)$ has been utilized to mitigate instability-intricacies during CNN training.

Angular margin makes the training more challenging; by increasing the angular distance of feature embedding with its respective ground-truth class center by a factor of m . This results in model learning more robust embeddings that stand-out in magnifying inter-class separation while minimizing intra-class separation. Due to classification now being done angularly on a hyper-spherical manifold, the generalization of the model is further enhanced. Furthermore, the loss function does not inculcate triplet/quadruplet sample mining.

2.2 Backbone Architecture

SphereFace Loss requires feature-rich embeddings to operate smoothly over the hyper-spherical deep metric learning. Thus, the HSADML framework incorporates the use of ImageNet pertained model of MobileNet. Specifically, MobileNet architecture has been chosen for the base-network because of the low-computational burdens that it lifts. The depth wise separable and point-wise convolutions drastically reduce the number of parameters in MobileNet while supporting good performance. Accredited to the same, the inference time of the MobileNet model is quite low. This all ensures that there is scalability in the approach for practical deployments in medical diagnosis.

The enriched representation extracted out of the MobileNet model is fed to Output Network wherein the feature-maps are firstly average pooled globally (GAP Layer), then the pooled embeddings are passed through a series of dense layers and dropout layers. Finally, after the final dense layer which is *He Instantiated* [19] and linearly activated, a Batch Normalization layer [20] is applied which helps in making the training stable. These are the final embeddings which are passed onto the SphereFace Loss function. In this case dimensionality of the final embeddings was chosen to be (256,1).

2.3 Classification and Testing

After the model has been trained under SphereFace Loss, embeddings from the final dense layer are extracted. Then using cosine distance as a distance metric k-NN ($k = 1$) algorithm is employed so as to test the model. The extracted embeddings are L_2 normalized, and instead of directly applying cosine distance metric, Euclidean metric over normalized embeddings was utilized. The normalization step ensures that there is proportionality between the respective Euclidean and Cosine distances. Mathematically, let there be two normalized embeddings x_{i1} and x_{i2} , then

$$\text{cosdist}(x_{i1}, x_{i2}) = 1 - x_{i1}^T x_{i2} \quad (4)$$

Similarly,

$$\text{eucdist}(x_{i1}, x_{i2}) = [2 * (1 - x_{i1}^T x_{i2})]^{\frac{1}{2}} \quad (5)$$

This makes,

$$\text{eucdist}(x_{i1}, x_{i2}) \equiv \text{cosdist}(x_{i1}, x_{i2}) \quad (6)$$

Here, $\text{cosdist}(\cdot)$ represents the cosine distance and $\text{eucdist}(\cdot)$ the Euclidean distance.

k-NN with the first nearest neighbor being a very basic classifier, HSADML based extracted embeddings are also classified using more sophisticated classifiers. Specifically, SVM with Gaussian and Polynomial kernel, Random Forest and k-NN (with best possible nearest neighbor combination considering up to 30 neighbors) have also been employed to facilitate robust classification of the extracted feature embeddings. For training all the classifiers, embeddings are extracted from both the training and testing set after which the classifier model is trained with the embeddings from the training set and for inference-evaluation testing set extracted embeddings are used.

2.4 Implementation and Network Training

The input to the model was 3-channel stacked MRI images with image dimensions being (224,224,3). Prior to inputting the images rescaling of pixel-intensities are rescaled to the range [0,1]. Data Augmentation strategies involving the use of random rotation, random brightness increment-decrement, random flipping and zooming-in are applied over the inputs. The augmented data is given to the MobileNet which produces 1024 feature-maps each having height and width as (7,7) respectively. In the Output Network, the first dense layer transforms the feature dimension from 1024 to 256 and then a Dropout layer [11] is applied with the rate of 0.2. For the SphereFace Loss function, experimentation on margin is done and the empirically found optimal margin i.e., $m = 5$ is considered while the scaling factor was set to 30. Stabilization in model performance was seen with the application of BatchNormalization after the final dense layer. The model was trained with Adam Optimizer and for 275 epochs with a custom learning rate schedule.

$$lr = \begin{cases} 1e - 4 & \text{if } epoch < 125 \\ 1e - 5 & \text{if } 125 \leq epoch < 175 \\ 1e - 6 & \text{otherwise} \end{cases}$$

This schedule was designed so as to slower the learning rate when convergence has been attained. For the case of SphereFace Loss function, the model converged started from about 100 epochs but loss kept optimizing till the end and consequently, the embeddings kept becoming better. Loss optimization for the proposed HSADML model has been illustrated in Fig. 3. The following curve suggests smooth convergence.

3 Experiments, Results and Discussions.

3.1 Dataset and Experimental Protocol

All the experimentation on the HSADML framework has been done using the Benchmark Dataset of Figshare [21]. It contains 3064 T1 weighted FLAIR images of Brain MRI with 708, 1426 and 930 belonging respectively to Meningioma, Glioma and Pituitary tumor classes. As suggested in [7] all the results were reported on 70–30% training–testing data-split, but to elucidate the robustness of the approach results over 30–70% training–testing data-split is also presented. These experiments involved the usage of a hold-out evaluation strategy wherein the testing data of 70–30% data-split became the training data of 30–70% data-split; vice-versa is also true. Along with

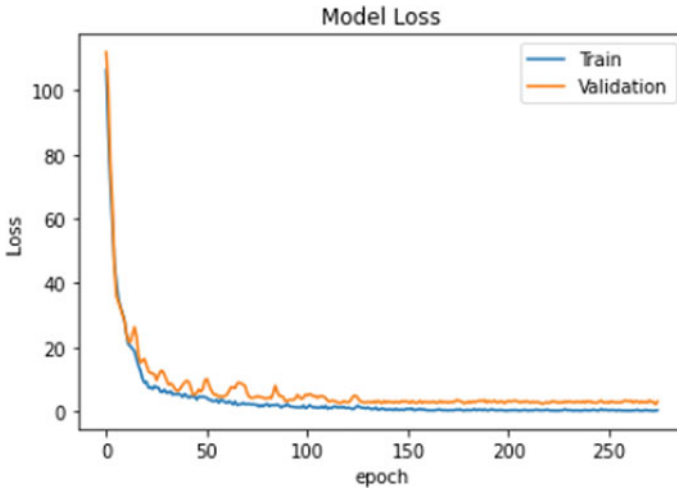


Fig. 3 Loss optimization during training of HSADML framework

analyzing the performance of the model over different data-settings, experimentation over the SphereFace Loss and different classifiers have been done but these experiments involved only the use of 70–30 data-split.

3.2 Performance Metrics

Models involved in this study are compared both graphically and numerically. For numeric performance evaluation following metrics have been used:

- **Accuracy:** Both intra-class and overall accuracy of the models have been computed. Let True Positives, False Negatives, True Negatives and False positive be represented by TP, FN, TN, FP. Then

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

- **F1-Score:** F1-Score represents the harmonic mean computed over both Precision ($\frac{TP}{TP+FP}$) and Recall ($\frac{TP}{TP+FN}$). It is a good metric for data-imbalance and also finds its significance for the cases where FN and FP have different prominence. It has also been computed over per-class and overall. Mathematically.

$$\frac{2}{F1} = \frac{1}{Precision} + \frac{1}{Recall}$$

Table 1 Performance analysis of HSADML framework for different data-split

Training dataset (in %)	Testing dataset (in %)	Accuracy (in %)				F1-score					MCC
		Me	Gl	Pt	Avg	Me	Gl	Pt	Macro avg	Micro avg	
70	30	95.85	99.76	99.28	98.69	0.9719	0.9964	0.9841	0.9841	0.9869	0.9796
30	70	89.00	97.01	97.99	95.47	0.9104	0.9667	0.9687	0.9486	0.9547	0.9289

- **Matthews Correlation Coefficient (MCC):** MCC takes into account entire confusion matrix and is a good evaluation measure for imbalanced-data.

$$MCC = \frac{TP * TN - FP * FN}{[(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)]^{\frac{1}{2}}}$$

- **Area Under Curve (AUC):** AUC value is the area under the ROC curve. Higher the AUC better the True Positive Rate and Lower the False Positive Rate.

3.3 Experimental Analysis

This section discusses on the results of various experiments and analysis over them. Firstly, the performance evaluation of the HSADML Framework over 70–30% and 30–70% training–testing data-split has been done. The results have been tabulated in Table 1. For the sake of simplicity, we refer each class by an abbreviation i.e., Meningioma (Me), Glioma (Gl) and Pituitary Tumor (Pt). The proposed framework at 70–30% data setting achieves overall high accuracy of 98.69% while maintaining a decent.

MCC score of 0.9762. The average accuracy of the model is significantly high for the classes Gl and Pt but a little less for Me which turns out to be the most challenging class. F1-Score on class as well as on an overall basis suggest significant performance. In the 30–70% data-split, training data is quite limited in amount. Although in this scarce training data experiment, performance is retained. Average accuracy achieved in this experiment is 95.47% while the MCC is 0.9289. Figure 4 shows the Confusion Matrix of both the models trained under both the data-splits. From the matrix, high-end performance over the Gl and Pt class can be noted but there remains a gap of improvement for the Me class. More specifically, recall of the models for the Me class is comparatively lesser than the precision. It can be inferred that the base-network utilized in the study is MobileNet, and hence when some more advanced backbone network shall be used, further performance gain is expected.

To further explore the capabilities of the proposed model, advanced classifiers have been tested with 70–30% Data-Split’s model extracted embeddings. Specifically, SVM with Gaussian and Polynomial kernel, Random Forest and k-NN (k = 1) (k = 1 was the best possible nearest neighbor combination with considering up to 30 neighbors) has also been employed. The results of the same have been tabulated in

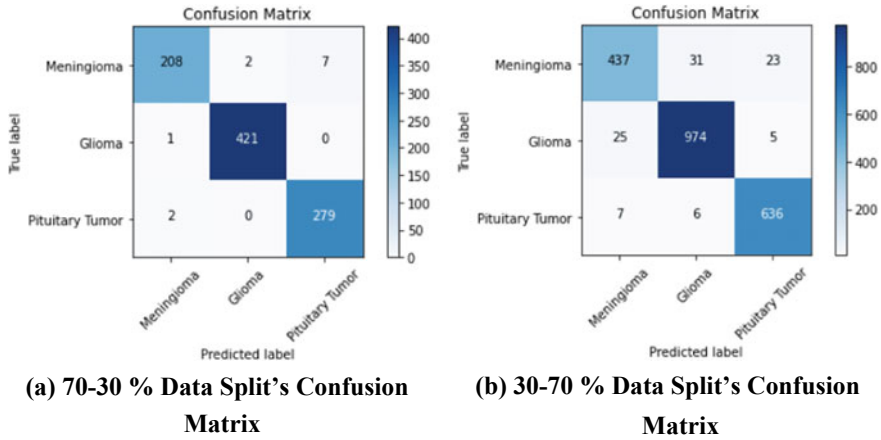


Fig. 4 Confusion matrices of HSADML framework

Table 2. For both k-NN and SVM with gaussian kernel, the model gave the same classification. Random Forest classifier using 400 estimators each utilizing 32 features showed a sustainable performance while SVM with a polynomial kernel of order 256 too obtained decent performance. The k-NN achieving the highest performance scores illustrates feature-level discriminability amongst various classes. SphereFace loss enjoyed greater stability in training than the SoftMax Loss model. It is clear from Table 3 that the proposed HSADML Framework facilitates comparatively higher performance than the other models. The obtained result further justifies on the use of angular margin. In terms of average accuracy, the proposed approach is superior than its counterparts of SoftMax Loss and Modified SoftMax Loss by 0.22% and 0.33% respectively. In a similar manner, the performance of SphereFace Loss is also better than the Triplet Metric Loss, although the model trained with triplet loss has better performance for the Meningioma class. In Table 4, for each class Area Under Curve (AUC) value calculated via ROC curves has been tabulated. It is worthwhile mentioning that AUC of SphereFace Loss model is the highest on an overall basis as well as for class Gl, while the same is comparable to the best for Me and Pt classes. Further, potential in the approach can be concluded in Fig. 5 which shows t-SNE Comparison amongst different Loss functions; from the figure, intra-class compactness and inter-class separability can be well spotted for the SphereFace Loss. For the case triplet loss, intra-class variation appears to be higher while accredited to angular metric, intra-class clustering is observed for the SphereFace Loss. Hence, it can also be concluded that with training in the hyper-spherical domain but without any angular margin, intra-class separation and inter-class discrepancy are not mitigated in comparison to SphereFace Loss.

For understanding the behavior of the angular margin, an ablation amongst different angular margins has been conducted. In this analysis $m = \{4, 5, 6\}$ has been considered. Empirically, the proposed model with $m = 5$ emerges out to be

Table 2 Comparison of HSADML framework generated embeddings using different machine learning classifiers

Classifier	Accuracy (in %)				F1-score					MCC
	Me	Gl	Pt	Avg	Me	Gl	Pt	Macro avg	Micro avg	
k-NN ($k = 1$)	95.85	99.76	99.28	98.69	0.9719	0.9964	0.9841	0.9841	0.9869	0.9796
Random forest	94.93	99.76	99.28	98.47	0.9671	0.9941	0.9846	0.9817	0.9847	0.9762
SVM (gaussian kernel)	95.85	99.76	99.28	98.69	0.9719	0.9964	0.9841	0.9841	0.9869	0.9796
SVM (polynomial kernel)	96.77	98.81	98.57	98.26	0.9630	0.9940	0.9805	0.9792	0.9826	0.9729

Table 3 Comparison of HSADML framework with various loss functions

Loss	Accuracy (in %)				F1-score					MCC
	Me	Gl	Pt	Avg	Me	Gl	Pt	Macro avg	Micro avg	
SoftMax loss	95.39	99.05	100.00	98.47	0.9695	0.9940	0.9825	0.9820	0.9847	0.9763
Modified softmax loss	94.93	99.76	99.28	98.47	0.9671	0.9941	0.9846	0.9817	0.9847	0.9762
Triplet loss	96.31	99.28	99.28	98.58	0.9698	0.9952	0.9891	0.9830	0.9858	0.9797
Sphere face loss (proposed)	95.85	99.76	99.28	98.69	0.9719	0.9964	0.9841	0.9841	0.9869	0.9796

Table 4 AUC based comparison of HSADML framework with various loss functions

Loss	Me	Gl	Pt	Avg
Softmax loss	0.9748	0.9942	0.9921	0.9870
Modified softmax loss	0.9741	0.9964	0.9878	0.9861
Triplet loss	0.9780	0.9954	0.9909	0.9880
Sphere face loss (proposed)	0.9771	0.9968	0.9909	0.9882

better in balancing the trade-off between hard-training and margin-based separation, thereby achieving the best performance. Quantitative results are presented in Table 5 while Fig. 6 graphically compares the SphereFace Loss variants using t-SNE plot. From the figure, a challenging competition lies between $m = 4$ and $m = 5$ models,

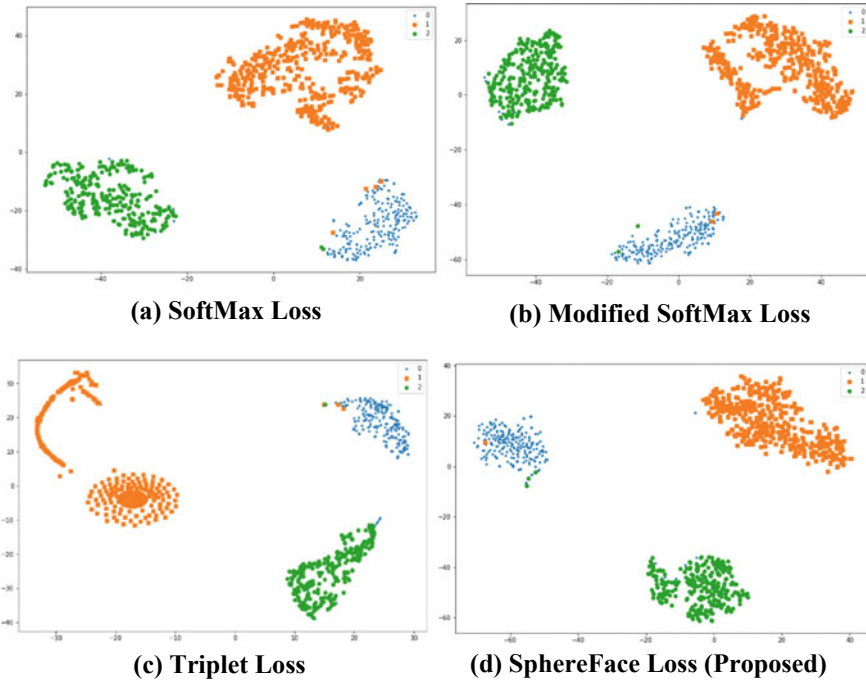


Fig. 5 t-SNE comparison of HSADML framework’s extracted embeddings with that of ones extracted with other loss functions. Blue color represents Meningioma Class, Orange Glioma and Green Pituitary Tumor

though $m = 5$ model attains higher inter-class separation. AUC based comparison of models trained with different angular margins has been shown in Table 6. The attained AUC value of the $m = 5$ curves are higher than the other variants. This trend can be identified specifically for the Gl and Me class, while for the Pt class the model trained with $m = 4$ attained slightly better result. From the following analysis over angular margins, it can be concluded that angular margin makes the training challenging which results in robust performance, but if the margin is not adequately tuned, then the model will converge sub-optimally.

Table 5 Ablation study over angular margin

Margin	Accuracy (in %)				F1-score					MCC
	Me	Gl	Pt	Avg	Me	Gl	Pt	Macro avg	Micro avg	
$m = 4$	95.85	99.05	99.64	98.47	0.9674	0.9917	0.9876	0.9822	0.9847	0.9762
$m = 6$	94.00	99.52	99.64	98.26	0.9622	0.9952	0.9790	0.9788	0.9826	0.9730
$m = 5$	95.85	99.76	99.28	98.69	0.9719	0.9964	0.9841	0.9841	0.9869	0.9796

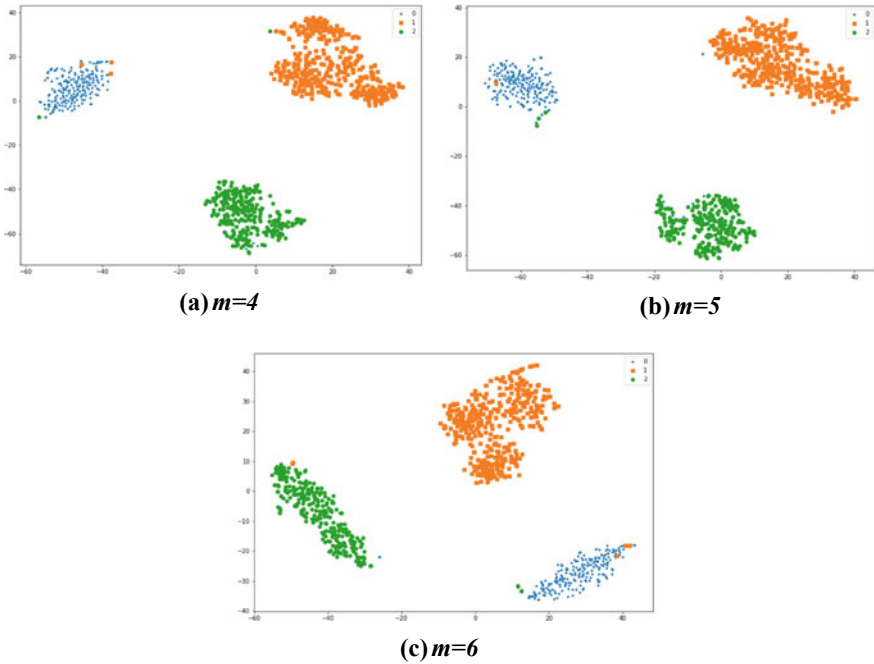


Fig. 6 t-SNE based graphical ablation over different angular margins. HSADML framework uses $m = 5$ as the angular margin. Blue color represents Meningioma Class, Orange Glioma and Green Pituitary Tumor

Table 6 AUC based comparison between models trained under different angular margins

Margin	Me	Gl	Pt	Avg
$m = 4$	0.9757	0.9922	0.9935	0.9871
$m = 6$	0.9679	0.9956	0.9896	0.9843
$m = 5$	0.9771	0.9968	0.9909	0.9881

Next part of the analysis is dedicated to the comparison of the HSADML framework with the other state-of-the-art approaches. A bar graph depicting a comparative analysis of average accuracy between approaches [4, 5, 7, 22–25] has been plotted in Fig. 7. Since, most of the previous works [4, 7] consider evaluation on 70–30%, training–testing data-split, we compare our results using the same protocol. It is evident from the same that the proposed HSADML framework outperforms the other approaches while obtaining encouraging results. It is to be pointed out that the HSADML framework attains better performance for each class, this makes it highly generalized. Thus, basing learning on angular distances in a hyper-spherical manifold can help in optimizing intra-class compactness while increasing inter-class distances. This brings significant improvement in the performance.

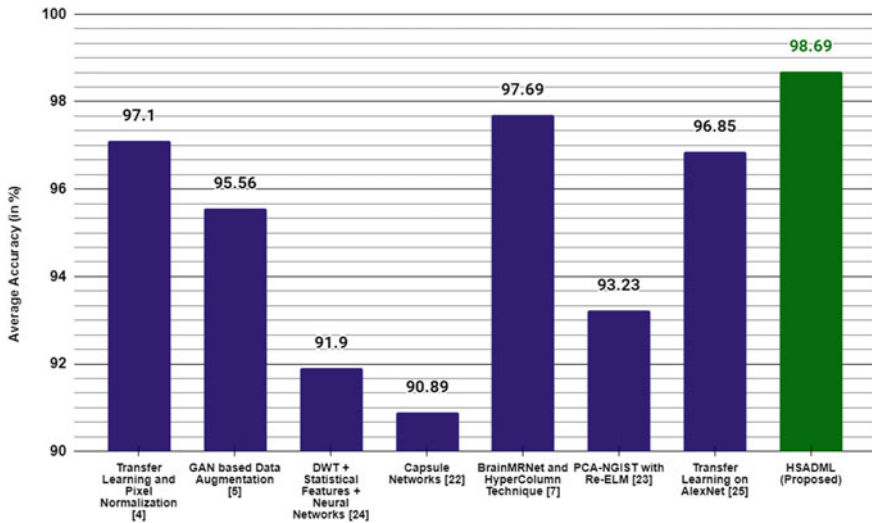


Fig. 7 Comparison of HSADML framework with state-of-the-art methods

4 Conclusion and Future Aspect

This paper proposed the HSADML framework for brain tumor classification. The proposed approach introduced the use of deep angular metric learning using SphereFace Loss for facilitating generalization and robustness in identification. The loss was instrumental in increasing intra-class separability and reducing intra-class variability; this resulted in achieving significant performance gains. This research didn't emphasize much over the backbone network thus in future aspects of the work there is a gap for the introduction of some attention-based domain-specific network. Another aspect of the research shall explore on in-depth validation of approaches which has been hampered due to a single-dataset. Nevertheless, the high-end performance and lightweight backbone of the HSADML framework motivates its scalability and extends application for other modalities as well.

References

1. Paul JS, Plassard AJ, Landman BA, Fabbri D (2017) Deep learning for brain tumor classification. In medical imaging 2017: biomedical applications in molecular, structural, and functional imaging. Int Soc Opt Photonics 10137: 1013710. <https://doi.org/10.1171/2.2254195>
2. Sajjad M, Khan S, Muhammad K, Wu W, Ullah A, Baik SW (2019) Multi-grade brain tumor classification using deep CNN with extensive data augmentation. J Comput Sci30:174-182. <https://doi.org/10.1016/j.jocs.2018.12.003>
3. Deng J, Dong W, Socher R, Li LJ, Li K, Fei LF (2019) A large-scale hierarchical image database. In: Proceedings of the IEEE conference on computer vision and pattern recognition .

- IEEE. pp 248–255 <https://doi.org/10.1109/CVPR.2009.5206848>
4. Deepak S, Ameer PM (2019) Brain tumor classification using deep CNN features via transfer learning. *Comput Biol Med* 111:103345. <https://doi.org/10.1016/j.compbio.2019.103345>
 5. Ghassemi N, Shoeibi A, Rouhani M (2020) Deep neural network with generative adversarial networks pre-training for brain tumor classification based on MR images. *Biomed Signal Process Control* 57:101678. <https://doi.org/10.1016/j.bspc.2019.101678>
 6. Afshar P, Plataniotis KN, Mohammadi A (2020) Boost caps: a boosted capsule network for brain tumor classification. In: *Proceedings of the 42nd annual international conference of the IEEE engineering in medicine & biology society (EMBC) IEEE*. pp. 1075–1079. <https://doi.org/10.1109/EMBC44109.2020.9175922>
 7. Toğaçar M, Ergen B, Cömert Z (2021) Tumor type detection in brain MR images of the deep model developed using hyper column technique, attention modules, and residual blocks. *Med Biol Eng Comput* 59(1):57–70. <https://doi.org/10.1007/s11517-020-02290-x>
 8. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: *Proceedings of the advances in neural information processing systems*. pp 5998–6008. <https://doi.org/10.1145/1218913.1218915>
 9. Gupta K, Thapar D, Bhavsar A, Sao AK (2019) Deep metric learning for identification of mitotic patterns of HEP-2 cell images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops* pp. 0–0. <https://doi.org/10.1109/CVPRW.2019.00141>
 10. Yi D, Lei Z, Liao S, Li SZ (2014) Deep metric learning for person re-identification. In: *Proceedings of the 22nd International Conference on Pattern Recognition IEEE*. pp. 34–39. <https://doi.org/10.1109/ICPR.2014.16>
 11. Chu R, Sun Y, Li Y, Liu Z, Zhang C, Wei Y (2019) Vehicle re-identification with viewpoint-aware metric learning. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision* pp 8282–8291. <https://doi.org/10.1109/ICCV.2019.00837>
 12. Schroff F, Kalenichenko D, Philbin J (2015) Facenet: a unified embedding for face recognition and clustering. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 815–823. <https://doi.org/10.1109/CVPR.2015.7298682.D>
 13. Thapar D, Jaswal G, Nigam A, Kanhangad V (2019) PVSNet: Palm vein authentication siamese network trained using triplet loss and adaptive hard mining by learning enforced domain specific features. In: *Proceedings of the IEEE 5th international conference on identity, security, and behavior analysis (ISBA) IEEE*. pp. 1–8. <https://doi.org/10.1109/ISBA.2019.8778623.D>
 14. Wen Y, Zhang K, Li Z, Qiao Y (2016) A discriminative feature learning approach for deep face recognition. In: *Proceedings of the European conference on computer vision*. Springer, Cham. pp 499–515. https://doi.org/10.1007/978-3-319-46478-7_31
 15. Chen W, Chen X, Zhang J, Huang K (2017) Beyond triplet loss: a deep quadruplet network for person re-identification. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 403–412. <https://doi.org/10.1109/CVPR.2017.145.D>
 16. Wang J, Zhou F, Wen S, Liu X, Lin Y (2017) Deep metric learning with angular loss. In: *Proceedings of the IEEE International conference on computer vision* pp. 2593–2601. <https://doi.org/10.1109/ICCV.2017.283>
 17. Liu W, Wen Y, Yu Z, Li M, Raj B, Song L (2017). Sphreface: deep hypersphere embedding for face recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 212–220. <https://doi.org/10.1109/CVPR.2017.713>
 18. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*. retrieved from <https://arxiv.org/pdf/1704.04861>
 19. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE international conference on computer vision* pp. 1026–1034. <https://doi.org/10.1109/ICCV.2015.123>
 20. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: *Proceedings of the international conference on machine learning*. PMLR. pp. 448–456
 21. Cheng J (2017). Brain tumor dataset figshare. <https://doi.org/10.6084/m9.figshare.1512427.v5>

22. Afshar P, Plataniotis KN, Mohammadi A (2019) Capsule networks for brain tumor classification based on MRI images and coarse tumor boundaries. In: Proceedings of the ICASSP 2019–2019 IEEE international conference on acoustics, speech and signal processing (ICASSP) IEEE. pp. 1368–1372 <https://doi.org/10.1109/ICASSP.2019.8683759>.
23. Gumaei A, Hassan AM, Hassan MR, Alelaiwi A, Fortino G (2019) A hybrid feature extraction method with regularized extreme learning machine for brain tumor classification. IEEE Access 7:36266–36273. <https://doi.org/10.1109/ACCESS.2019.2904145>.
24. Ismael MR, Ikhlas AQ (2018) Brain tumor classification via statistical features and back-propagation neural network. In: Proceedings of the IEEE international conference on electro/information technology (EIT), IEEE. pp. 0252–0257. <https://doi.org/10.1109/EIT.2018.8500308>
25. Kaur T, Gandhi TK (2020) Deep convolutional neural networks with transfer learning for automated brain image classification. Mach Vis Appl 31(3):1–16. <https://doi.org/10.1007/s00138-020-01069-2>

Document Analysis and Recognition (DAR)

Model Compression Based Lightweight Online Signature Verification Framework



Chandra Sekhar Vorugunti, S. Balasubramanian, Pulabaigari Viswanath,
and Avinash Gautam

1 Introduction

An online signature is considered as a legitimate means of verifying one's identity. The rapid advances in digital technologies resulted in the wide usage of electronic gadgets like Graphic Tablets, Stylus Pens, etc., which are equipped with a pressure-sensitive screen to capture the online signature. These devices apprehend both the geometric (x, y co-ordinates) and dynamic properties (pressure, inclination of pen, inclination of device etc.) of the writer signing trace by sensing the movements of the pen-tip.

The recent advances in Deep Learning technologies resulted in Convolutional neural networks (CNNs), which demonstrated extraordinary outcomes in challenging computer vision problems like Image Segmentation, Object Detection etc. [1–4]. To develop CNN-based OSV frameworks to achieve improved classification accuracies, the main impediment is the increasing size of CNN models. This prevents OSV frameworks from being widely used in devices with minimal computational resources, such as mobile/embedded devices. Real-time user authentication is in high demand since light weight devices are commonly employed for M-Commerce and banking applications. Hence, the deployment of OSV frameworks in these lightweight devices is a critical requirement.

C. S. Vorugunti (✉) · P. Viswanath
Indian Institute of Information Technology, Sri City, India
e-mail: chandrasedkhar.v@iiits.in

S. Balasubramanian
Sri Sathya Sai Institute of Higher Learning, Anantapur, India

A. Gautam
Birla Institute of Technology & Science, Pilani, India

To address this issue, we are using an emerging topic of model pruning, which removes the filters and neurons which are less contributive and less important in learning. The model pruning results in fine-tuning the CNN-based frameworks to be deployable in mobile/embedded devices. In this work, we propose a light weight CNN-based online signature verification system and improve the classification outcomes with CNN pruning. In line with this, we preview various CNN-based OSV frameworks proposed in the literature.

2 Literature Survey

In [5], the authors suggested a CNN-centric OSV framework that consists of a collection of depth-wise separable (DWS) convolution layers. The OSV framework based on DWS convolution results in a light weight model with fewer training parameters and allows for few shot learning. In the Skilled-01 category of the MCYT-100 dataset, the model earned a state-of-the-art EER of 13.42%. Following that, [6] introduced a deep convolutional Siamese network-based OSV framework (DCSN). Based on contrastive loss, the Siamese networks extract robust writer-specific feature descriptions, allowing the model to learn intra-writer variability (Genuine-Genuine) and inter-individual variability quickly (Genuine-Forgery). Another intriguing paper is [2], which creates a hybrid feature set by combining handcrafted features with Convolution Autoencoder high-level feature representations (CAE). The Depth-wise Separable Convolutional Neural Network receives the hybrid feature set as an input (DWSCNN). In the skilled-01 category of the MCYT-100 dataset, the model earned a state-of-the-art EER of 13.38%. In [1], the authors have suggested a hybrid feature fusion-based OSV system in which the CAE's deep representational features are fused with handcrafted features that reflect the cluster heads of writer-specific feature vectors. The model achieved an EER of 13.26% in the Skilled-01 category of the MCYT-100 dataset. To increase the speed and accuracy of online signature verification, Okawa et al. [7] suggested a unique single-template technique that employs mean templates based on Euclidean barycenter-based DTW barycenter averaging (EB-DBA) and local stability weighted dynamic time warping (LS-DTW). The framework has an EER of 0.72% in the MCYT dataset's Skilled-5 category and 2.08% in the SVC dataset's Skilled-5 category. For training a 1D convolutional neural network model, Lai et al. [8] designed a novel learning-by-synthesis approach. In the Skilled-5 category of the SVC dataset, the model had an EER of 3.88%.

Even though various OSV frameworks are proposed as discussed above, no work is proposed in the OSV landscape which discusses on analyzing the contribution of each filter and each neuron towards classification accuracy. The improved performance of CNN frameworks is often due to the improved model sizes with hundreds of layers and millions of trainable parameters. Hence, in this work, we are proposing a CNN model pruning-based on removing less important filters and neurons, which is memory efficient and facilitates real-time signature classification.

The primary contributions of this paper are as follows:

- We have presented a new OSV framework based on Depth-wise Separable CNN (DWSCNN), which reduces the amount of parameters and operations required by the framework significantly.
- To prune the non-contributive filters of convolution layers and neurons of dense layers of CNN architecture, we suggested a unique CNN pruning technique.
- Experiments on three commonly used datasets, namely MCYT-100, SVC, and SUSIG, were employed to thoroughly evaluate the proposed model.

The rest of the paper is structured into four sections. Several phases of our suggested OSV model are presented in Sect. 3. The experimentation analysis, as well as the model’s outcome, are discussed in Sect. 4. The proposed model is compared to other recent state-of-the-art models in this paper. Conclusion is presented in Sect. 5.

3 Proposed Online Signature Verification Model

The proposed OSV framework is a collection of two modules, i.e., (a) a pruning technique, and (b) Depth-wise Separable Convolution based OSV framework which is depicted in Fig. 1. Each module of the framework is discussed in detail in the following sub-sections.

3.1 The Proposed Pruning Technique

Deep learning has gone a long way in recent years and has established itself as a key computer technology in the digital age. Image processing and computer vision are two representative application areas [1, 2, 9]. ResNet [10] and VGGNet [11] are two well-known image categorization models. Although these new models attain high accuracy and quick inference times, they require a large number of parameters to be learned, which consumes a lot of memory and requires a lot of processing power [12, 13]. The fundamental disadvantage of CNN frameworks is that they are heavy weight. Redundancy between multiple filters and neurons is common in CNNs [12,

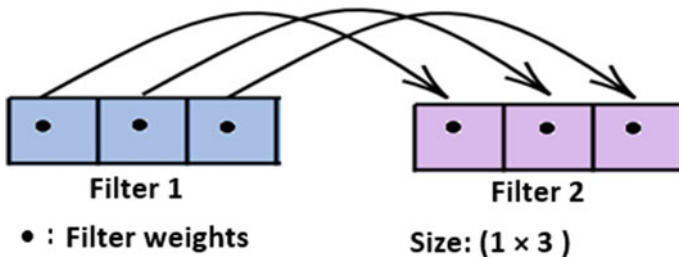


Fig. 1 Pictorial interpretation of step 6 and step 7 of algorithm-1

14]. This paper proposes a novel CNN pruning strategy to remove redundant CNN filters and dense layer neurons that do not contribute to model performance.

The recent works on network pruning [4, 15], confirm that the trend is to prune redundant, non-informative weights in pre-trained CNN models, based on the magnitude of the weights themselves. Inspired by this, we perform pruning as shown in Algorithm–1. In steps 6 and 7 of the pruning algorithm, and as illustrated in Fig. 1, we first check if every element of every filter (of size $1*5$) of each layer is greater than a predefined threshold τ_e . Even if one element is lesser than the predefined threshold, the filter is pruned. We eliminate similar filters by pruning the filters having a lesser element-wise distance to reduce the number of filters. In the case of dense layers, the weights that are lesser than the average weights are set to zero for each dense layer. Subsequent to the pruning of convolutional and dense layers, an inference is performed on the validation data (denoted by X_Val). If the accuracy drop is $\leq \tau$, again the same series of steps are executed. Finally, the pruned model is used for inference on the test data. As an ablation study, which is illustrated in Tables 1, 2 and 3, we have experimented with nine types of pruned models including the baseline model.

Algorithm-1: Neural Network Pruning

1. **Input:** Basenet, reference data X_Val , training data X_Train , tolerable accuracy drop τ , element wise threshold τ_e , distance threshold τ_d
 2. **while** Accuracy_Drop \leq Tolerable Range τ not reached **do**
 3. **for each** ConvLayer **in** Basenet **do**
 4. **for each** j^{th} filter F_{ij} in Layer i
 5. Obtain the array_of_weights $conv_{ij}$ which has all the weights of the filter F_{ij}
 6. **if** each element of $conv_{ij} > \tau_e$
 7. $d_{jk} = \text{EuclideanDistanceBetweenFilters}(F_{ij}, F_{ik})$
 8. **if** $d_{jk} < \tau_d$
 9. prune filter F_{ik} // **Computational Complexity = $O(L \cdot N^2)$**
 10. **else**
 11. prune F_{ij}
 12. **end for**
 13. **end for**
 14. **for each** Dense_Layer **in** Basenet **do**
 15. Obtain the array_of_weights_dense $_p$ representing the weights of the p^{th} dense layer
 16. Replace the weights $< \text{avg}(\text{array_of_weights_dense}_p)$ with 0
 17. **end for**
 18. **end while**
 19. // return the pruned
-

Table 1 The EER outcome by various proposed OSV models: MCYT-100 dataset

	S_01	S_05	S_10	S_15	S_20	R_01	R_05	R_10	R_15	R_20
Model1: baseline model	10.77	6.75	2.87	0.75	0.10	2.53	0.48	0.16	0.04	0.01
Model2: pruned model	7.98	2.15	0.36	0.30	0.00	2.28	0.19	0.05	0.01	0.00
Model3: 3 times pruned	9.92	3.73	0.76	0.34	0.00	2.32	0.21	0.07	0.03	0.00
Model4: 10 times pruned	10.31	5.81	1.13	0.65	0.03	2.41	0.27	0.1	0.05	0.00
Model5: pruning: dense1	10.45	6.12	1.56	0.76	0.06	2.57	0.31	0.18	0.05	0.01
Model6: pruning: dense2	10.57	6.18	1.61	0.82	0.11	2.61	0.28	0.19	0.1	0.02
Model7: pruning: convolution layers	11.13	7.14	1.89	1.13	0.14	2.69	0.34	0.19	0.12	0.02
Model8: pruning: dense1 + convolution layer	9.87	6.87	1.9	1.2	0.13	2.57	0.29	0.21	0.15	0.03
Model9: pruning: dense2 + convolution layer	10.12	6.91	1.93	1.4	0.27	2.79	0.41	0.28	0.13	0.05

Table 2 The EER outcome by various proposed OSV models: SVC dataset

	S_01	S_05	S_10	S_15	R_01	R_05	R_10	R_15
Model1: baseline model	6.55	2.57	0.65	0.00	7.47	1.94	0.04	0.07
Model2: pruned model	3.65	1.77	0.45	0.00	3.72	0.24	0.00	0.00
Model3: 3 times pruned	4.92	3.12	2.14	2.78	5.45	2.25	2.00	1.98
Model4: 10 times pruned	6.39	5.48	4.28	2.13	5.13	2.12	2.16	2.10
Model5: pruning: dense1	6.65	5.97	4.81	3.13	5.60	2.65	2.18	2.15
Model6: pruning: dense2	6.78	6.12	5.13	3.56	5.64	2.70	2.43	2.31
Model7: pruning: convolution layers	6.10	6.58	6.23	3.23	6.33	3.15	3.11	2.87
Model8: pruning: dense1 + convolution layer	6.82	7.1	6.46	3.76	6.14	3.12	3.00	3.01
Model9: pruning: dense2 + convolution layer	6.19	6.92	6.11	3.33	6.38	3.16	3.14	3.12

Table 3 The EER outcome by various proposed OSV models: SUSIG dataset

	S_01	S_05	S_10	S_15	R_01	R_05	R_10	R_15
Model1: baseline model	17.93	13.65	8.07	6.6	11.13	2.37	1.44	0.85
Model2: pruned model	12.39	4.45	1.47	1.1	9.13	2.84	1.38	0.12
Model3: 3 times pruned	16.14	7.45	4.76	4.13	10.17	2.91	1.43	0.15
Model4: 10 times pruned	17.67	8.34	5.34	5.67	10.87	2.72	1.41	0.14
Model5: pruning: dense1	15.75	7.01	5.11	5.24	10.73	2.64	1.37	0.16
Model6: pruning: dense2	15.64	6.70	4.87	4.29	10.11	2.75	1.43	0.19
Model7: pruning: convolution layers	15.64	6.70	4.87	4.29	10.11	2.75	1.43	0.19
Model8: pruning: dense1 + convolution layer	14.39	7.34	5.14	5.25	9.98	3.21	1.54	0.21
Model9: pruning: dense2 + convolution layer	14.76	7.23	5.65	5.97	10.16	3.13	1.51	0.20

4 Ablation Study of Various Pruned Models

4.1 Experimentation Setup

To appraise the proposed pruning technique and OSV framework, we have evaluated the baseline and various pruned models by conducting experiments on the widely accepted MCYT-100, SVC and SUSIG datasets [1, 2, 9, 16]. We have mainly focused on few shot learning, i.e., S_01, S_05, S_10, S_15, S_20, R_01, R_05, R_10, R_15 and R_20 categories, where ‘‘S’’ and ‘‘R’’ represents the Skilled and Random forgery experimentation categories.

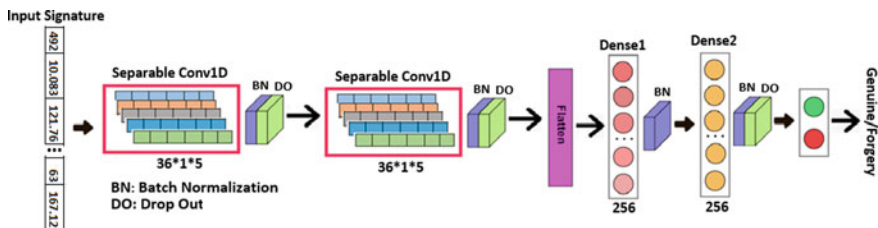


Fig. 2 Block diagram of the proposed online signature verification model

4.2 Ablation Study

We investigated eight types of pruning models in addition to the baseline model to provide a more in-depth analysis of the pruned model’s performance. As depicted in Tables 1, 2 and 3, we have considered nine models for experimentation. Model1 representing the baseline model is an unpruned model, whose architecture is depicted in Fig. 2. Model2 is the pruned version of the baseline model using the proposed pruning algorithm. Model3 represents a pruned model in which the filters and weights are reduced or made zero by 1/3rd of the total weights of the baseline model. Model4 represents the pruned model in which 1/10th of the filters and the weights of the baseline model are set to zero. Model5–Model9 represents the pruned models in which various combinations of convolution and dense layers are inactivated. The specifics are available in the tables. Tables 1, 2 and 3 summarize that a balance needs to be maintained while pruning the filters and weights of neurons. Random pruning of filters and weights of neurons does not result in optimal EER. The baseline model pruned with our proposed algorithm results in least EER in all categories of experimentation. The pruned model results in the state-of-the-art EER in Skilled-1 category of MCYT-100 and SVC datasets by yielding an EER of 7.98 and 3.65%. It is clearly depicted in Tables 1, 2 and 3 that the other pruned models are not efficient in delivering lesser EER. The point to be noted is that, for optimal EER, the combination of convolution and dense layers are equally important. The pruning of either of the layers is not efficient to yield lesser EER.

5 Comparative Study

We compare and contrast the suggested framework with contemporary and state-of-the-art frameworks in this part. The frameworks with the lowest EER values are highlighted with (*), while those with the next lowest EER are marked with (♦). In the context of MCYT-100, the suggested framework achieved state-of-the-art results in the S_01, S_10, S_15, S_20, R_01, R_05, R_10, R_15, R_20 categories. In the SVC dataset categories S_01, S_05, S_10, S_15, R_10, and R_15, the framework achieved exceptional EER. In the case of SUSIG, the framework achieved the highest

EER in the categories S_15 and R_15. Tables 4, 5 and 6 show a comparison of EER values with state-of-the-art OSV frameworks, which are assessed on similar datasets to those used to evaluate our proposed framework. The (*) denotes that the model produced the best EER value, while the (♦) denotes that the model produced the second best EER value. In the MCYT-100 (DB1) category, our proposed framework achieves state-of-the-art performance in S_01 (one shot learning), S_05, S_10, S_15, S_20, R_10, R_15, and R_20.

With reference to SVC, the framework yields the outstanding EER in all Skilled categories of experimentation and R_10 and R_15 categories. In regard to SUSIG, the framework outcomes the best EER in the S_15 and R_15 categories. Even while alternative frameworks proposed in [2, 5, 7, 8, 10, 11, 17, 26, 27] yield lower EER values than the proposed framework, these models are not comprehensively examined with all possible categories of experimentation, i.e., Skilled 1,5,10,15,20 and Random 1, 5, 10, 15, 20, as shown in Tables 4, 5, 6. The ability of OSV frameworks to deploy in real-time contexts is measured by greater classification accuracies with a small number of training samples and for all potential categories of experiments. As a result, we have thoroughly tested our proposed model in both skilled and random forgery categories using all feasible training situations (1, 5, 10, 15, 20), and the results are summarized in Tables 4, 5 and 6. Hence, we can summarize that the pruning mechanism should aim at compressing and accelerating the models without sacrificing much of predictive performance.

6 Conclusion and Future Work

In this paper, we offer an online signature verification method based on depth-wise separable convolutional neural networks. Similar filters and neurons with weights less than the average weights are removed using the pruning procedure. In the CNN and dense layers, we presented a pruning strategy to remove noncontributing filters and neurons. As a result, the model is simplified. In addition to the pruned model, we looked at models with weights that were 1/3rd and 1/10th of the baseline model's weights, as well as the baseline model pruned with various combinations of convolution layers and dense layers. To the best of our knowledge, this is the first time a pruning technique has been applied to the OSV framework. We thoroughly tested the model using all types of experimentation, including skilled 1, 2, 3, 4, 5, 10, 15, 20, and random 1, 2, 3, 4, 5, 10, 15, 20. Furthermore, on most standard signature datasets, the suggested approach surpassed the state-of-the-art result, indicating that more study in this area is warranted.

Table 4 Comparative analysis of the proposed model against SOTA models on MCYT-100 dataset. (Symbol * indicates first best EER and \blacklozenge second best EER)

Method	S_01	S_05	S_10	S_15	S_20	R_01	R_05	R_10	R_15	R_20
Proposed: baseline model	10.77	6.75	2.87	0.75 \blacklozenge	0.10	2.53	0.48	0.16	0.04 \blacklozenge	0.01 \blacklozenge
Proposed: pruned model	7.98 \blacklozenge	2.15	0.36*	0.30*	0.00*	2.28	0.19	0.05*	0.01*	0.00*
Clustering + feature fusion [1]	13.26	2.66	2.58	3.01	1.2	6.01	2.91	0.07	0.05	0.04
Feature fusion + few shot learning [2]	13.38	3.02	1.83 \blacklozenge	1.25	1.2	4.03	0.42	0.1	0.09	0.08
RPWC classifier [9]	-	9.7	-	-	-	-	3.4	-	-	-
Target-wise [17]	13.56	-	-	-	-	4.04	-	-	-	-
Deep learning + DTW (Tested with only 50 users) [18]	-	2.40	-	-	-	-	-	-	-	-
Angular robotic features [19]	-	3.44	-	-	-	-	0.75	-	-	-
Stroke based RNN [20]	10.46	-	-	-	-	-	-	-	-	-
Fewshot learning [5]	13.42	7.03	5.70	3.95	2.20	2.00 \blacklozenge	0.05	0.06 \blacklozenge	0.01*	0.00*
Stroke-wise [17]	13.72	-	-	-	-	5.04	-	-	-	-
SynSig2Vec[8]	3.84*	2.38	-	-	-	0.55*	-	-	-	-

Table 5 Comparative analysis of the proposed model against SOTA models on SVC dataset

Method	S_01	S_05	S_10	S_15	R_01	R_05	R_10	R_15
Proposed: baseline model	4.37♦	2.57	0.65	0.00	7.47	1.94	0.04	0.07
Proposed: pruned model	3.65*	1.77♦	0.45♦	0.00*	3.72	0.24	0.00*	0.00*
Feature clustering + few shot learning [1]	7.71	3.43	2.75	0.45	3.09	0.41	0.12	0.22
Feature fusion + few shot learning [2]	5.95	3.93	2.98	0.45	3.61	0.39	0.13	0.19
Relief-1 [21]	–	–	8.10	–	–	–	–	–
Stroke point warping [22]	–	–	1.00	–	–	–	–	–
Fewshot learning [5]	5.83	0.87*	0.35*	0.2♦	9.08	1.4	0.15	0.02♦
Target-wise [17]	18.63	–	–	–	0.50*	–	–	–
Stroke-wise [17]	18.25	–	–	–	1.90♦	–	–	–
SynSig2Vec [8]		3.88				0.17		
Machine learning [23]		5.76						
Classifiers + machine learning [3]		2.62						
Time-series averaging [7]		2.08	1.53			0.11	0.03♦	

Table 6 Comparative analysis of the proposed model against SOTA models on SUSIG dataset

Method	S_01	S_05	S_10	S_15	R_01	R_05	R_10	R_15
Proposed: baseline model	17.93	13.65	8.07	6.6♦	11.13	2.37	1.44	0.85♦
Proposed: pruned model	12.39	4.45	1.47	1.1*	9.13	2.84	1.38	0.12*
Clustering + few shot learning [1]	15.84	4.95	1.68	–	1.70*	1.37	0.21*	–
Feature fusion + few shot learning [2]	17.96	5.17	2.07	–	1.87♦	1.53	0.30♦	–
Pole-zero models [16]	–	2.09	–	–	–	–	–	–
With all domain [24]	–	–	3.88	–	–	–	–	–
Kinematic Theory based [25]	7.87	–	–	–	3.61	–	–	–
VSAr–DTW [26]	–	3.09	–	–	–	0.78*	–	–
Target-wise [17]	6.67*	–	–	–	1.55	–	–	–
Fewshot learning [5]	10.41	0.8*	0.63♦	–	8.7	2.5	1.26	–
Stroke-wise [17]	7.74♦	–	–	–	2.23	–	–	–
Information divergence [27]	–	1.6♦	2.13	–	–	–	–	–
Curvature + Hausdorff distance[28]	–	7.05	–	–	–	1.02	–	–
VSA–DTW [26]	–	3.83	–	–	–	0.78*	–	–

References

1. Vorugunti C, Pulabaigari V, Mukherjee P, Sharma A (2020) DeepFuseOSV: online signature verification using hybrid feature fusion and depthwise separable convolution neural network architecture. *IET Biometr* 9:259–268
2. Vorugunti C, Pulabaigari V, Gorthi R, Mukherjee P (2020) OSVFuseNet: online signature verification by feature fusion and depth-wise separable convolution based deep learning. *Neurocomputing* 409:157–172
3. Chandra S, Singh K, Kumar S, Ganesh K, Sravya L, Kumar B (2021) A novel approach to validate online signature using machine learning based on dynamic features. *Neural Comput Appl* 33:12347–12366
4. Han S, Mao H, Dally WJ (2015) Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint [arXiv:1510.00149](https://arxiv.org/abs/1510.00149)*
5. Vorugunti C, Gorthi R, Pulabaigari V (2019) Online signature verification by few-shot separable convolution based deep learning. In: 2019 International Conference on Document Analysis and Recognition (ICDAR)
6. Vorugunti C, Devanur SG, Mukherjee P, Pulabaigari V (2021) OSVNet: convolutional siamese network for writer independent online signature verification. In: 2019 International conference on document analysis and recognition (ICDAR). Sydney, Australia, pp 1470–1475
7. Okawa M (2021) Time-series averaging and local stability-weighted dynamic time warping for online signature verification. *Pattern Recogn* 112:107699
8. Lai S, Jin L, Zhu Y, Li Z, Lin L (2021) SynSig2Vec: forgery-free learning of dynamic signature representations by sigma lognormal-based synthesis. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. pp 1–1
9. Nanni L, Lumini A (2006) Advanced methods for two-class problem formulation for on-line signature verification. *Neurocomputing* 69:854–857
10. Al-Hmouz R, Pedrycz W, Daqrouq K, Morfeq A, Al-Hmouz A (2017) Quantifying dynamic time warping distance using probabilistic model in verification of dynamic signatures. *Soft Comput* 23:407–418
11. Guru D, Manjunatha K, Manjunath S, Somashekara M (2017) Interval valued symbolic representation of writer dependent features for online signature verification. *Expert Syst Appl* 80:232–243
12. Doroz R, Kudlacik P, Porwik P (2018) Online signature verification modeled by stability-oriented reference signatures. *Inf Sci* 460–461:151–171
13. Parziale A, Diaz M, Ferrer M, Marcelli A (2019) SM-DTW: stability modulated dynamic time warping for signature verification. *Pattern Recogn Lett* 121:113–122
14. Manjunatha KS, Manjunath S, Guru DS, Somashekara MT (2016) Online signature verification based on writer dependent features and classifiers. *Pattern Recogn Lett* 80:129–136
15. Yeom S, Seegerer P, Lapuschkin S, Binder A, Wiedemann S, Müller K, Samek W (2021) Pruning by explaining: a novel criterion for deep neural network pruning. *Pattern Recogn* 115:107899
16. Rashidi S, Fallah A, Towhidkhal F (2013) Authentication based on pole-zero models of signature velocity. *J Med Signals Sensors* 3:195
17. Diaz M, Fischer A, Ferrer M, Plamondon R (2018) Dynamic signature verification system based on one real signature. *IEEE Trans Cybernet* 48:228–239
18. Guru D, Manjunatha K, Manjunath S (2013) Online signature verification based on recursive subset training. *mining intelligence and knowledge exploration*. 350–361
19. Diaz M, Ferrer M, Quintana J (2018) Robotic Arm Motion for Verifying Signatures. In: 2018 16th International conference on frontiers in handwriting recognition (ICFHR)
20. Li C, Zhang X, Lin F, Wang Z, Liu J, Zhang R, Wang H (2019) A stroke-based RNN for writer-independent online signature verification. In: 2019 International Conference on Document Analysis and Recognition (ICDAR)
21. Yang L, Cheng Y, Wang X, Liu Q (2018) Online handwritten signature verification using feature weighting algorithm relief. *Soft Comput* 22:7811–7823

22. Kar B, Mukherjee A, Dutta P (2018) Stroke point warping-based reference selection and verification of online signature. *IEEE Trans Instrum Meas* 67:2–11
23. Chandra S (2020) Verification of dynamic signature using machine learning approach. *Neural Comput Appl* 32:11875–11895
24. Pirlo G, Cuccovillo V, Diaz-Cabrera M, Impedovo D, Mignone P (2015) Multidomain verification of dynamic signatures using local stability analysis. *IEEE Trans Hum Mach Syst* 45:805–810
25. Fischer A, Plamondon R (2017) Signature verification based on the kinematic theory of rapid human movements. *IEEE Trans Hum Mach Syst* 47:169–180
26. Diaz M, Ferrer M, Quintana J (2019) Anthropomorphic features for on-line signatures. *IEEE Trans Pattern Anal Mach Intell* 41:2807–2819
27. Tang L, Kang W, Fang Y (2018) Information divergence-based matching strategy for online signature verification. *IEEE Trans Inf Forensics Secur* 13:861–873
28. He L, Tan H, Huang Z (2019) Online handwritten signature verification based on association of curvature and torsion feature with Hausdorff distance. *Multimedia Tools Appl* 78:19253–19278

End-to-End Transformer-Based Architecture for Text Recognition from Document Images



Dipankar Ganguly, Akkshita Trivedi, Bhupendra Kumar, Tushar Patnaik, and Santanu Chaudhury

1 Introduction

Towards development of robust Optical Character Recognition System (OCR) with high tolerance to image degradation as well as capacity to handle deprecated characters, several challenges are faced. One of the reasons attributed to this is non-availability of such noisy datasets to fine tune systems. Furthermore, the standard evaluation methods are comprised of high quality datasets. Thus, end-to-end realization of OCRs with such capacity is seldom encountered even within the research community.

Motivation of our work derives from the end goal of having an End-to-End book conversion pipeline, agile enough to suit the peculiarities of several books still possess strong recognition accuracies. Recurrent Neural Networks (RNNs) [7] or specifically the Bi-Directional Long Short-Term Memory Sequences (Bi-LSTMs) [18] in document images have fairly established the state of the art in terms of End-to-End Modelling. Nevertheless, the models are powerful and comprehend the learning of diverse scripts and language peculiarities. Considerable efforts have been made in sequence-to-sequence modelling, particularly in language models and Machine Translation [4, 6, 38], thus re-shaping the encoder-decoder architectures [26, 41]. More recently, the Transformer-based approaches have gained momentum [39], though the very limited application of the same is seen in text recognition.

Supported by MeitY-Ministry of Electronics & Information Technology, Government of India.

D. Ganguly (✉) · B. Kumar · T. Patnaik
Centre for Development of Advanced Computing (C-DAC), Noida, Uttar Pradesh, India
e-mail: dipankargangulycdac@gmail.com

A. Trivedi · S. Chaudhury
Indian Institute of Technology (IIT), Jodhpur, Rajasthan, India

The standard Recurrent networks, particularly LSTMs perform computation or tuning alongside the symbol positions from the input and output. This kind of sequential nature inhibits any parallelization of training, this has also been discussed by others [26, 39]. Thus, training longer sequences impose extremely long training times. In terms of computational efficiency, various approaches have been proposed and significant work has been carried out [1]. However, inherent sequential nature of the problem still remains with all the approaches.

On the other hand, Attention-based mechanisms have consistently secured their place in sequence-to-sequence modelling tasks [12]. The Attention mechanism essentially enables the global dependencies between the input and output. However, Attention-based mechanisms are mostly combined with RNN architectures. Thus, the shortcoming of the existing architecture motivated design and experiment with our existing architecture particularly for historical document recognition.

In this paper, we propose a Transformer-based OCR architecture fused with Masked BERT Language Model with attention layers for document image recognition in addition to GAN and DBPN-based denoising and Super-Resolution (SR) [32] for state-of-the-art document image recognition. We shall share our experiences in designing the architecture particularly in relation to convergence of Transformer Network and the insights of our experiment. We empirically show that the architecture outperforms existing state of the art for Odiya document images from Odiya Virtual Academy [29].

2 Background

The standard process of text recognition for document images begins with denoising and Super-Resolution (SR) of document image [8, 14, 44]. Denoising becomes an essential component as high noise is observed in historical documents. Subsequently, SR the module intends to derive a high resolution (HR) output from a low resolution (LR) document image. The challenges like low resolution and the presence of skew are being handled within this layer.

This is followed by Image Document segmentation which is intended to extract text segments comprising either a character or word or even a line. Essentially, text-segmentation works by labelling a set of spatially adjusted features comprising of group of pixels with visual highlights. Broadly, segmentation is classified as line-based, word based or character-based [2, 35].

The latest developments for recognition are centred around end-to-end memory networks, which are usually based on a recurrent attention mechanism, which are proven to be better in comparison with sequence-aligned recurrence across various tasks. This module finally takes the segmented sequences and generates corresponding characters or words. In our proposed architecture, the segmented words are fed to our Transformer-based model architecture fused with a Masked BERT Language Model to recognize text. This has been coupled with Global and Normalized Attention Mechanisms, enabling transformers to support more parallelism with faster convergence and state-of-the-art accuracies.

3 Related Work

There are a handful of works on Transformer architecture-based Text Detection or Object Detection, such by Lyu et al. [27]. They proposed an architecture for Scene Text Detection using Attention and Transformer-based units to recognize texts. In another work [5] Carion et al. have used Transformers in Object Detection and have presented a novel framework.

In direction of sophisticated SR, due to breakthrough advancement in the field of the deep neural networks, many new SR methods have been proposed [11, 20, 21]. Similarly, Zhang et al. [43] presented a CNN-based Super resolution algorithm specific to the demands of OCR. However, the proposed methods are mostly feed-forward in nature. Thereby having inability of mapping the relationship between LR and HR. Hence, in this paper, we try to integrate SR method proposed by Haris et al. [15], which proposes a projection network, feeding back the error predictions at each layer, collecting the self-correcting features for up-sampling at every stage to improve SR resolution.

In terms of Segmentation, Long et al. [24] revolutionized the idea of semantic segmentation by using ImageNet database [9]. They have up-sampled the images with deconvolutional layers and subsequently appending lower layers to improve predictions. In other domains such as Scene text detection, in design of image segmentation, many deep learning-based algorithms have been developed with promising results [23, 25, 42]. In document image segmentation, Ronneberger et al. [33] introduced U-Net by proposing a U-shaped symmetric architecture between the contracting path to capture setting and expanding path that empowers exact restrictions, with a lower layer for each level. Also, Santos et al. [34] developed a multi-step handwritten text-segmentation framework, utilizing Y and X histogram projections to eliminate false lines and words, respectively.

Hochreiter and Schmidhuber [19] introduced Long Short-Term Memory in 1997, which has become de-facto standard in the Text Recognition. Graves et al. [13] proposed the use of Bi-Directional LSTM(Bi-LSTM) architecture which allowed bi-directional (forward and backward layers) longer range context. In this paper, we propose integrating the architecture introduced by Ray et al. [31] which combines Connectionist Temporal Classification(CTC) to learn the labelling unsegmented and unaligned sequence of data and a Bi-LSTM model for the advantages mentioned above.

Developments of complete text recognition solutions, such as [28] proposed Arabic handwritten document segmentation framework, utilizing a variant of U-net with residual blocks(RU-net) for text line segmentation; for word segmentation, BLSTM-CTC (Bidirectional Long Short-Term Memory followed by a Connectionist Temporal Classification). In the similar line, CRAFT [3] proposed a text detector without character annotations by generating a pseudo character-level ground truths from an interim word-level datasets. Also, EAST [45] was designed for fast and accurate text detection with a single neural network and with appropriate loss function, rotated

rectangular or quadrilateral text regions are generated. In another work by Liao et al. [22], a single layer neural network word-based text detector called TextBoxes, combined with a text recognition algorithm named Convolutional Recurrent Neural Network(CRNN) [36] produced significant results in terms of prediction.

Significant accuracy in text-based detection is also discussed in [40] by introducing a deep learning-based super-resolution framework without additional computing cost.

4 Proposed Framework

Our proposed architecture comprises cascaded document image enhancement and recognition as depicted in Fig. 1. It broadly comprises of three modules, i.e. denoising and super-resolution module, U-net-Based Segmentation, and Transformer-based Recognition Combined with BERT Language Model as decoder which is jointly optimized.

Initially, the document image is fed into the DBPN [15] and GAN module which achieves denoising and SR task by an iterative, multi-stage up (extracting features to upgrade to HR) and down (resizing the image as per LR configuration)-sampling operators, connected mutually to extract the non-linear relation between LR and HR, as shown in the Fig. 1.

The U-net [33] architecture has been modified, which consists of 2 paths, the encoder or the contracting path catches the settings in the image. The encoder consists of convolutional and max-pooling layers stacked together. The decoder or the expanding path empowers exact limitation utilizing transposed convolutions. Thus, an end-to-end FCN is generated, which can accept an input image of any size. Figure 1 shows the U-Net architecture.

Finally coming to recognition, the proposed transformer architecture consists of a Residual Network [16] (Res-Net) layer at first, particularly the pre-trained Res-Net 18 is used with Transfer Learning approach [30]. The transfer learned Res-Net 18 is followed by a fully connected layer that acts as a bridge to connect the encoder transformer units, which is a modified Multi-Head Attention [39] and is a multi-layer bi-directional transformer encoder. The layer is responsible to capture global dependencies between the feature map and output, this works by aggregating information from the parts of the input. This layer is followed by a Normalized Attention Layer that acts as a separator to disconnect output with attention computation, thus enabling parallel and faster optimization and convergence. Finally a word-based decoder is fused with BERT Masked LM to decode corresponding words.

Our approach is mostly script independent and would work for a wide variety of historical document images. We have performed our experiments on Odia Historical books from Odia Virtual Academy as such datasets are not popular and very difficult to curate.

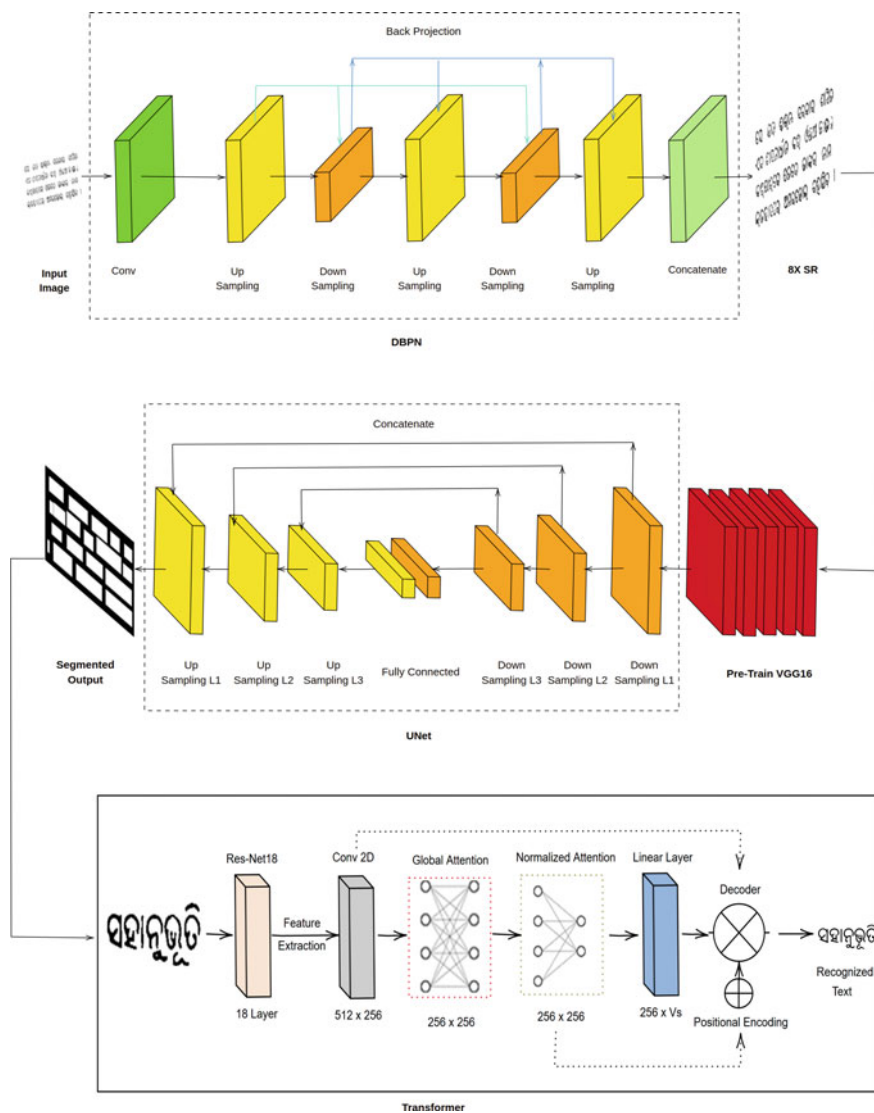


Fig. 1 The proposed Transformer Architecture with Masked BERT LM decoder

5 Methodology

In this section, we describe finer details about our proposed framework.

5.1 Super Resolution and Segmentation

Document image size of 64×64 is fed to the DBPN architecture at first, the image size is up-scaled to 512×512 and then, back-projected to 64×64 , the residual between the reconstructed and the observed LR/HR maps is up/down-scaled accordingly and added with the observed maps to enhance the extraction. The result is a SR image of output 512×512 . The SR image picks up pre-trained weights from VGG16 model, such that we eliminate the possibility of weights being assigned randomly, thus increasing our chances of accurate prediction. The pre-trained weights and SR image is now passed to U-Net. During encoding, the SR image undergoes 2 un-padded convolutions recurrently, with each convolution followed by a leaky Rectified Linear Unit (ReLU) and max-pooling for down-sampling with a stride of 2. The number of feature channels doubles at every step of down-sampling ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512$). During decoding, the feature maps are up-sampled, followed by halving the feature channels and adding with the corresponding down-sampled features at the time of encoding, and 2 un-padded convolutions and leaky ReLU. The output from U-Net, i.e. the extracted words are then fed to the proposed transformer. The segmented words are then fed to the transformer.

5.2 Global Attention

In order to capture global dependencies, we have used a Global Attention mechanism which is built with Multi-Head Attention units [39] which works to aggregate information from input. Instead of going linearly performing single attention, we perform this in parallel by projecting the queries, keys, and values of the transformer to different dimensions across each value. We apply global attention module to the flattened feature map generated by Res-Net followed by a convolutional layer, the feature sequence I with shape of $k \times c$, which precisely in our case is 512×256 . Corresponding to each feature vector we use $I_i (i \in [1, k])$, the position vector E_i is embedded with position index i . Subsequently, a feature vector F with position embedded information is available. We have incorporated many transformer layers into series to aggregate information from F . Within each transformer unit, the corresponding query, keys and values are obtained as [27]:

$$Q_l^i = \begin{cases} F_i & l = 1, \\ O_{l-1}^i & l > 1 \end{cases} \quad (1)$$

$$K_l^i = \begin{cases} F_i & l = 1, \\ O_{l-1}^i & l > 1 \end{cases} \quad (2)$$

$$V_l^i = \begin{cases} F_i & l = 1, \\ O_{l-1}^i & l > 1 \end{cases} \quad (3)$$

In the Eq. 1, Q_l^i represents query vector for the i -th transformer unit in the l -th transformer layer. In same equation O_{l-1}^i represents the output from the previous transformer layer. Alongside Q_l^i , in Eq. 2 K_l^i & 3 V_l^i are corresponding key and value vectors for the same l -th layer of the transformer.

With all necessary query, key and value vectors computed within each transformer unit, the overall transformer output is weighted summation of all the values. The weights sum of each layer is calculated with the formula :

$$\alpha_l^{ij} = \frac{\exp(W_l^q Q_l^i \cdot W_l^k Q_l^{ij})}{\sum_{j'=1}^k \exp(W_l^q Q_l^i \cdot W_l^k Q_l^{ij})} \quad (4)$$

In the Eq. 4, W_l^q denotes the trainable weights across the layers. Finally, the output of each transformer is represented as

$$O_l^i = \text{act_func} \left(\sum_{j=1}^k \alpha_l^{ij} W_l^v \cdot V_l^{ij} \right) \quad (5)$$

In the above Eq. 5, W_l^v is the learned weight and act_func is a non-linear activation which is can be referred from [10, 39]. The outputs of the last transformer is taken as the global attention layer output.

5.3 Normalized Attention

Normalized Attention is a sparse layers of transformer encoders to aid parallelism in attention and provide separation of Global Attention to Outputs. Residual connections are built to promote information exchange between the layers.

The basic attention, as described in [37], is designed to work serially and are usually integrated with Recurrent Networks as

$$\alpha_t = \text{Attention}(h_{t-1}, \alpha_{t-1}, \mathbf{I}) \quad (6)$$

In the above Eq. 6, h_{t-1} and α_{t-1} represents the weights of the hidden state and attention of the previous steps of RNN decoder, I is the encoded feature sequence. Therefore, computation of the step t is limited by previous steps, which is a bottleneck.

To overcome this problem, we have devised a Normalized Attention, by modifying the architecture proposed in [27]. In this layer, the dependency relationships are not fully dependent and can be optimized simultaneously. The normalization is dependent on the following function:

$$\alpha_t = \text{softmax}(W_2 \tanh(W_1 O^T)) \tag{7}$$

In the above equation W_1 W_2 represents learnable parameters on the global attention layer.

Once the computation is completed only the normalized weights coefficients α , encoded within the feature sequence are obtained as output from the layer, which is given by the Eq. 8, where indexes i & j represent the outputs node and feature vector index:

$$G_i = \sum_{j=1}^k \alpha_{ij} I_j \tag{8}$$

This layer prevents the range of values in the other layers into changing too much, thus the model trains faster and has a better ability to generalize.

All the available OCRs are compared with our proposed architecture and compared to empirically show the best with Critical distance diagram. In Fig. 2 character recognition accuracies are compared whereas Fig. 3 showcases comparison with word recognition accuracies. The abbreviations are OCR EG—Classical Odia OCR based on cascaded rule-based architectures; DNN—Deep Bi-LSTM with CTC Recognition Engine; Tesseract OCR for Odia; DNN LM—Deep Bi-LSTM with CTC Recognition with Fraternal dropout-based Language Model; TOCR—Proposed Transformer-based OCR

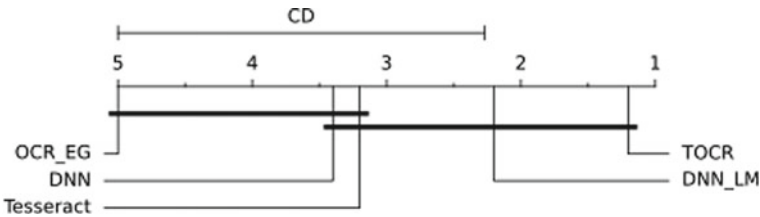


Fig. 2 Character Level Critical Distance

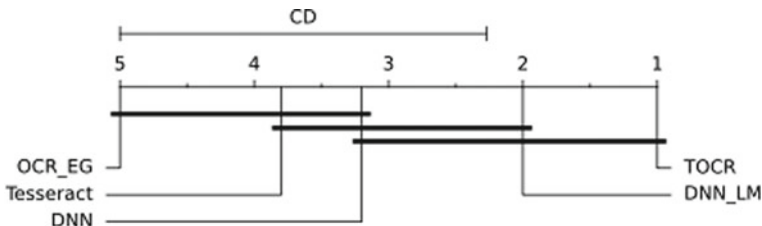


Fig. 3 Word Level Critical Distance

5.4 Decoder

The transformer decoder is fused with a pre-trained Language model on Odia Corpus. We have 1 Million Odia sentences from Odia Virtual Academy Books and Web Crawl. Thereby, we have trained a Masked BERT LM [10]. The decoder of the transformer unit is combined with the Masked LM decoder and predicts output words with probability given by

$$P_i = \text{softmax}(WG_i + b) \tag{9}$$

Here, W and b are weights and biases learned during training.

This amalgamation of Masked LM is essential as the Normalized Attention removes certain dependencies which are pivotal to the decoder and is compensated by the BERT LM. The entire transformer network is jointly optimized (Table 1).

6 Experimental Results

6.1 Dataset

We have tested our approach on Odia script. We have 26 books from Odia Virtual Academy scanned at 300 dpi, of which 17 books are very old; published between 1920 and 1980, and 9 books are relatively new (published after 1980). In total our dataset comprises 1700 pages with total 423 K words and 54 K unique words. Thus, the dataset has a good blend of very old, degraded, noisy documents along with newer printed documents. These together combine to put forth a plethora of document recognition challenges. Text graphics separation as well as segmentation is performed before the document is fed into the recognition pipeline.

Our datasets are divided into Training (80%), Test (10%), and Validation (10%). The DBPN along with SR with DBPN and U-net is trained on 15K segmented noise resilient images along with corresponding original noise present in the image.

Table 1 The Table compares the accuracy of our proposed framework with existing Odia OCR

Validation data	Proposed OCR		DNN OCR + LM		DNN OCR		OCR engine(Consortia)		Tesseract	
	Word (%)	Char (%)	Word (%)	Char(%)	Word (%)	Char (%)	Word (%)	Char (%)	Word (%)	Char (%)
Test Set 1	92.12	94.03	90.60	93.80	89.00	93.50	68.84	86.78	85.70	93.52
Pages: 89										
Words: 23457										
Test Set 2	90.15	93.34	86.83	91.23	85.00	90.91	7.64	12.30	20.03	53.47
Pages: 34										
Words: 8302										
Test Set 3	88.05	94.79	84.74	91.79	83.00	91.03	3.47	7.40	15.93	47.24
Pages: 26										
Words: 6584										
Test Set 4	87.12	93.81	83.05	92.61	82.00	92.30	5.40	16.66	17.24	52.50
Pages: 103										
Words: 23893										
Test Set 5	89.78	92.05	83.47	91.32	82.35	91.18	70.63	87.13	83.22	93.65
Pages: 94										
Words: 25690										

6.2 Results

We have performed intensive testing across different available OCRs for Odia and also carried out hypothesis testing using Autorank [17] for character and word-level accuracies as shown in Fig. 2.

References

1. Anwar S, Hwang K, Sung W (2015) Fixed point optimization of deep convolutional neural networks for object recognition. In: 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 1131–1135. IEEE
2. Arica N, Yarman-Vural FT (2001) An overview of character recognition focused on off-line handwriting. IEEE Trans Syst Man Cybern Part C (Appl Rev) 31(2):216–233. <https://doi.org/10.1109/5326.941845>
3. Baek Y, Lee B, Han D, Yun S, Lee H (2019) Character region awareness for text detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9365–9374
4. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint [arXiv:1409.0473](https://arxiv.org/abs/1409.0473)

5. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-end object detection with transformers. In: European conference on computer vision, pp 213–229. Springer
6. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint [arXiv:1406.1078](https://arxiv.org/abs/1406.1078)
7. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint [arXiv:1412.3555](https://arxiv.org/abs/1412.3555)
8. Dai D, Wang Y, Chen Y, Gool LV (2015) How useful is image super-resolution to other vision tasks? CoRR [arXiv:1509.07009](https://arxiv.org/abs/1509.07009)
9. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp 248–255. IEEE
10. Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805)
11. Dong C, Loy CC, Tang X (2016) Accelerating the super-resolution convolutional neural network. In: European conference on computer vision, pp 391–407. Springer
12. Dutil F, Gulcehre C, Trischler A, Bengio Y (2017) Plan, attend, generate: planning for sequence-to-sequence models. arXiv preprint [arXiv:1711.10462](https://arxiv.org/abs/1711.10462)
13. Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2009) A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans Pattern Anal Mach Intell* 31(5):855–868. <https://doi.org/10.1109/TPAMI.2008.137>
14. Greenspan H (2009) Super-resolution in medical imaging. *Comput. J.* 52(1):43–63. <https://doi.org/10.1093/comjnl/bxm075>
15. Haris M, Shakhnarovich G, Ukita N (2018) Deep back-projection networks for super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1664–1673
16. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
17. Herbold S (2020) Autorank: a python package for automated ranking of classifiers. *J Open Source Softw* 5(48):2173 (2020). <https://doi.org/10.21105/joss.02173>
18. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780
19. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
20. Kim J, Lee JK, Lee KM (2016) Deeply-recursive convolutional network for image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1637–1645
21. Lai WS, Huang JB, Ahuja N, Yang MH (2017) Deep laplacian pyramid networks for fast and accurate super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 624–632
22. Liao M, Shi B, Bai X, Wang X, Liu W (2017) Textboxes: a fast text detector with a single deep neural network. Proceedings of the AAAI conference on artificial intelligence, vol 31(1). <https://ojs.aaai.org/index.php/AAAI/article/view/11196>
23. Liu X, Meng G, Pan C (2019) Scene text detection and recognition with advances in deep learning: a survey. *Int J Document Anal Recogn (IJ DAR)* 22(2):143–162
24. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3431–3440
25. Long S, He X, Yao C (2020) Scene text detection and recognition: the deep learning era. *Int J Comput Vis*, 1–24
26. Luong MT, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. arXiv preprint [arXiv:1508.04025](https://arxiv.org/abs/1508.04025)
27. Lyu P, Yang Z, Leng X, Wu X, Li R, Shen X (2019) 2d attentional irregular scene text recognizer. arXiv preprint [arXiv:1906.05708](https://arxiv.org/abs/1906.05708)

28. Neche C, Belaid A, Kacem-Echi A (2019) Arabic handwritten documents segmentation into text-lines and words using deep learning. In: 2019 international conference on document analysis and recognition workshops (ICDARW), vol 6, pp 19–24. <https://doi.org/10.1109/ICDARW.2019.501110>
29. Parida S, Bojar O, Dash SR (2020) Odiencorp: odia–english and odia-only corpus for machine translation. In: Smart intelligent computing and applications, pp 495–504. Springer
30. Pratt LY (1993) Discriminability-based transfer between neural networks. *Advances in neural information processing systems*, pp 204–204
31. Ray A, Rajeswar S, Chaudhury S (2015) Text recognition using deep blstm networks. In: 2015 Eighth international conference on advances in pattern recognition (ICAPR), pp 1–6. <https://doi.org/10.1109/ICAPR.2015.7050699>
32. Ray A, Sharma M, Upadhyay A, Makwana M, Chaudhury S, Trivedi A, Singh A, Saini A (2019) An end-to-end trainable framework for joint optimization of document enhancement and recognition. In: 2019 international conference on document analysis and recognition (ICDAR), pp. 59–64. IEEE
33. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, pp 234–241. Springer
34. Santos RP dos, Clemente GS, Ren TI, Cavalcanti GD (2009) Text line segmentation based on morphology and histogram projection. In: 2009 10th international conference on document analysis and recognition, pp 651–655. IEEE
35. Santos RPD, Clemente GS, Ren TI, Cavalcanti GDC (2009) Text line segmentation based on morphology and histogram projection. In: Proceedings of the 2009 10th international conference on document analysis and recognition, pp 651–655. ICDAR '09, IEEE Computer Society, USA. <https://doi.org/10.1109/ICDAR.2009.183>
36. Shi B, Bai X, Yao C (2016) An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans Pattern Anal Mach Intell* 39(11):2298–2304
37. Shi B, Wang X, Lyu P, Yao C, Bai X (2016) Robust scene text recognition with automatic rectification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4168–4176
38. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. arXiv preprint [arXiv:1409.3215](https://arxiv.org/abs/1409.3215)
39. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. arXiv preprint [arXiv:1706.03762](https://arxiv.org/abs/1706.03762)
40. Wang L, Li D, Zhu Y, Tian L, Shan Y (2020) Dual super-resolution learning for semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
41. Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, Krikun M, Cao Y, Gao Q, Macherey K et al (2016) Google’s neural machine translation system: bridging the gap between human and machine translation. arXiv preprint [arXiv:1609.08144](https://arxiv.org/abs/1609.08144)
42. Xu Y, Wang Y, Zhou W, Wang Y, Yang Z, Bai X (2019) Textfield: learning a deep direction field for irregular scene text detection. *IEEE Trans Image Process* 28(11):5566–5579. <https://doi.org/10.1109/TIP.2019.2900589>
43. Zhang H, Liu D, Xiong Z (2017) Cnn-based text image super-resolution tailored for ocr. In: 2017 IEEE visual communications and image processing (VCIP), pp 1–4. <https://doi.org/10.1109/VCIP.2017.8305127>
44. Zhang L, Zhang H, Shen H, Li P (2010) A super-resolution reconstruction algorithm for surveillance images. *Signal Process* 90(3):848–859
45. Zhou X, Yao C, Wen H, Wang Y, Zhou S, He W, Liang J (2017) East: an efficient and accurate scene text detector. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5551–5560

A Hybrid Approach for Table Detection in Document Images



Sunil Kumar Vengalil, Kevin Xavier, Konda Amith Sai, Sree Harsha, Ganesh Barma, and Neelam Sinha

1 Introduction

Detection of structural elements like tables, paragraphs, and headers in images of digital documents is one of the core problems in automatic document processing. The problem of table detection is almost as complex as object detection in natural images and hence traditional image processing and machine learning approaches do not perform well. Using image processing approaches, coming up with a perfect algorithm that works in all possible scenarios is infeasible as it requires tuning a number of parameters and thresholds [1]. Further these parameters highly depend on the type and quality of documents. The challenge in using Machine Learning (ML) approaches is coming up with the right set of features. Evidence from object detection [2, 3] tells us that deep learning approaches should perform better than algorithmic and ML approaches. However, the challenge here is the non-availability of a huge number of annotated samples for table detection similar to ImageNet [4] for object detection. Most of the publicly available datasets [5, 6] for table detection tasks are either (1) small in size typically of the order of a few thousands or even less or (2) restricted to certain types of documents [6]. However, one of the recent dataset, TableBank [7], has 417K images of word and latex documents taken from the Internet.

One of the feature-based approaches [1] uses morphological operations in order to segment structures like text blocks and lines. Another popular image-processing-

Supported by Mphasis CSR grant

S. K. Vengalil (✉) · K. A. Sai · S. Harsha · G. Barma · N. Sinha
International Institute of Information Technology, Bangalore, India
e-mail: sunilkumar.vengalil@iiitb.org

K. Xavier
Razorthink Technologies, Bangalore, India
e-mail: kevin.xavier@razorthink.com

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022
U. Mudenagudi et al. (eds.), *Proceedings of the Satellite Workshops of ICVGIP 2021*,
Lecture Notes in Electrical Engineering 924,
https://doi.org/10.1007/978-981-19-4136-8_11

based approach uses the fact that spacing between columns will be larger than spacing between words [8]. Like any other image-processing-based algorithm, these methods also suffered from the drawback that the algorithm is difficult to generalize across document types. Due to non-generalizability of algorithmic approaches, Machine-Learning-based techniques were introduced. Kasar et al. [9] utilizes SVM classifier with features like horizontal and vertical lines to segment tables in document images. Embley et al. [10] provide an excellent survey of various table detection approaches using image processing and Machine Learning. Anh et al. use graph neural networks [11] to segment tables in images of invoice documents. Azka Gilani et al. use faster-RCNN [2] for segmenting tables in document images after separating out text and non-text regions [12].

Even though Deep-Learning-based approaches give better performance, they require large annotated data and suffer from lack of explainability.

To circumvent the above issues, in this article, we propose a hybrid approach where we augment a recent deep learning model, TableNet [13], by adding additional input channels with word-level features. In particular, our major contributions in this study are as follows:

1. We introduce a set of features based on alignment and spacing between words in order to detect tabular structures in document images.
2. We further illustrate the significance of these features using multiple ML classifiers Logistic Regression and SVM.
3. We introduce a hybrid model where a recent deep learning model, TableNet [13], is augmented by providing an additional input image with words segmented.

2 Proposed Method

We perform table segmentation using the following three approaches:

1. Classify each word as table word versus non-table word using binary classifiers, Logistic Regression, and SVM, with word alignment features mentioned in Sect. 2.2 as the input to the classifier.
2. Using a deep learning model TableNet [13] to segment the table boundaries.
3. A hybrid approach by augmenting the TableNet with an additional input channel. A binary image with all words masked, as shown in Fig. 5, is fed at this input channel.

In the case of SVM and logistic regression, we use 12 word-level features derived from the word patch segmentation output. SVM and logistic regression directly predict whether each word belongs to a table or not. Even though the word-level binary classifier does not give exact table boundaries, it is beneficial in two ways: (1) It gives a quantitative measure of how significant the features are for table detection. (2) It can be used to refine the prediction from other two approaches. We leave this work as a future work and in this work we just report the results of word-level binary

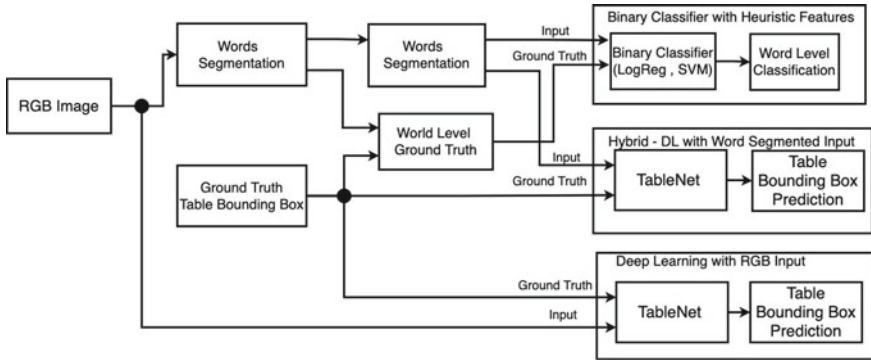


Fig. 1 Pipeline of the proposed approaches

classifier using the proposed features. For the DL-based approach, the images are directly provided as input. For the proposed hybrid approach, we use binary images with segmented words, see Fig. 5 as input. The DL and hybrid approaches generate pixel-level segmentation for table regions from which word-level classification is computed for comparing the approaches. The overall pipeline of the proposed method is illustrated in Fig. 1.

2.1 Dataset

We performed experiments on the marmot dataset. The dataset consists of images of 2000 pages extracted from research article documents of which 1000 are for Chinese and 1000 are for English language. For each language, 500 documents are with, possibly multiple, tables and 500 documents are without any tables. For generating the training set, we used only 500 English document images which had at least one table present in it. A sample image from Marmot dataset is shown in Fig. 2a.

2.2 Preprocessing and Feature Detection

The preprocessing and feature detection pipeline is shown in Fig. 3. The document image is first converted to binary image by Otsu’s thresholding. After this preprocessing stage, text lines in the image are segmented using the spacing between each line. The text lines are further segmented into words, again using the character spacing, and the top, bottom, left, and right coordinates of each word are obtained.

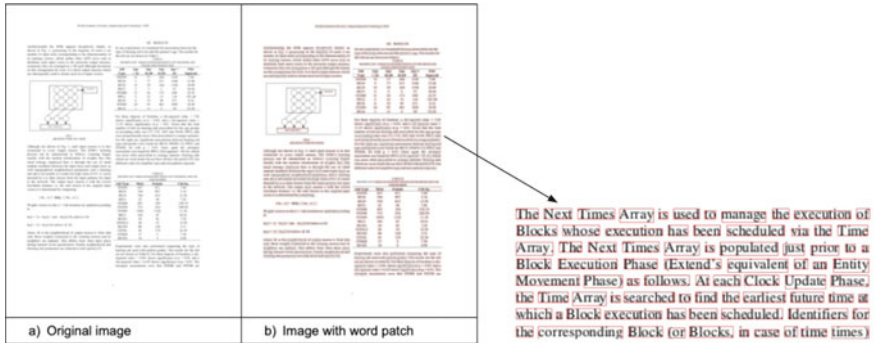


Fig. 2 Sample image from Marmot dataset and corresponding output of word segmentation

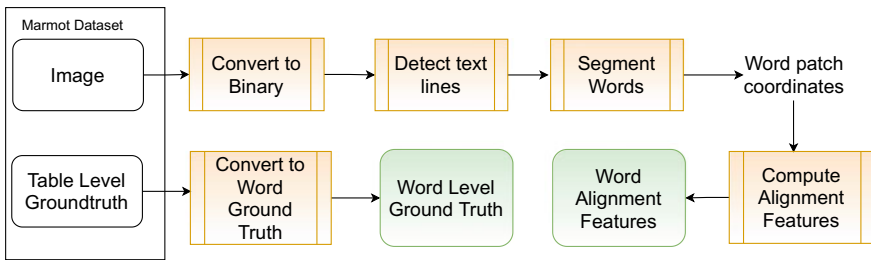


Fig. 3 Feature detection pipeline

2.2.1 Segmenting Text Lines and Word Patches

Text and non-text regions in the document were separated out first using a DL model that was trained using synthetically generated images and labels. Each text block in the document was further broken down into individual text lines. This is done by counting the number of background pixels along each row. The count is normalized with respect to the width of the text block and if the normalized value is less than a threshold the entire row is labeled as blank row. All the consecutive blank rows are merged into a single region which corresponds to the spacing between two rows. Rectangular regions between line spacing are taken as the bounding box coordinates of text lines.

Once each text line is segmented, a similar algorithm is used to segment the word patches in each text line. This time the number of background pixels along each column of a text line is found and normalized with respect to the height of the text line. If the normalized value is less than a threshold the column is marked as blank column and all consecutive blank columns are merged to get the spacing between two word patches.

Since the abovementioned algorithm is based on thresholds which can vary from document to document, we obtained additional evidences for word patches using two more third-party tools:

1. Tesseract—An Open-Source OCR library from Google that also gives bounding box coordinates of word patches.
2. Google Cloud Vision—A cloud-based API that parses a document image and provides information like word patch bounding box and text inside each word patch image.

We combined the word patch bounding box information from all three sources (our algorithm, Tesseract, and Google Cloud Vision) using a max voting scheme and generated the final word patch output.

The entire document is represented by a collection of words W , where each $w \in W$ is a 4-tuple (t, l, b, r) representing the top, left, bottom, and right coordinates of the word patch image. See Fig. 2b for a sample image with boundaries marked.

2.2.2 Word Alignment Features

Using the word patch coordinates, we computed 12 derived features that capture left and right alignment between words and spacing between words. Features computed based on left alignment are given in Table 1. Similar features are computed for right and center alignment also. Each word is represented by a 12-dimensional feature vector.

Finally, table ground-truth coordinates in the original Marmot dataset [5] were used to give a binary label, table versus non-table, to each word in the document. Table 2 summarizes the number of table and non-table words used in training and validation sets.

Table 1 Sample features for left alignment. Similar features are added for right and center alignment also

Feature name	Description
Top_left_aligned	Number of words left aligned to the word lying above the word
Continuous_top_left_aligned	Number of words left aligned to the word lying immediately above the word contiguously
Bottom_left_aligned	Number of words left aligned to the word lying below the word
Continuous_bottom_left_aligned	Number of words left aligned to the word lying immediately below the word contiguously

Table 2 Dataset used for training ML models. Features detected on word patches extracted from Marmot dataset are used for training and validation

	Table words	Non-table words	Total
Training	39478	39478	78956
Validation	16919	159625	176544
Total	56397	199103	255500

2.3 Classification of Words

Figure 1 shows the architecture of the overall training and prediction system.

Following three approaches were followed to train a model for classifying each word as table or non-table word:

1. Binary Machine Learning classifiers trained using 12-dimensional feature vectors corresponding to each word.
2. Using Deep learning model TableNet [13] to find the bounding box of each table. Each word lying inside the bounding box predicted by TableNet is classified as table words and all other words are marked as non-table words.
3. A hybrid approach where the TableNet model is augmented with an additional input channel carrying information about boundaries of word segments.

2.4 Machine Learning Approach

Figure 4 shows the block diagram of ML training pipeline. The word-level features mentioned in Table 1 and the label, table versus non-table, are used to train binary classifiers Logistic Regression and SVM.

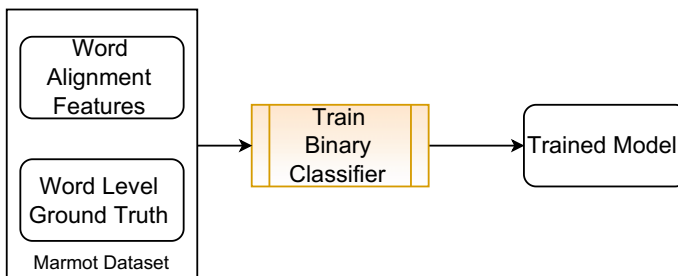


Fig. 4 Block diagram of the machine learning training pipeline. We used Logistic Regression and SVM

2.5 Deep Learning Approach

We trained a TableNet [13] model using raw images normalized and resized to 256×256 . TableNet is an end-to-end deep learning model that leverages the inherent interdependencies between table discovery and table structure identification. This model uses an initialized base network with pre-trained VGG19 functionality. Two decoder branches follow for (1) table area segmentation and (2) column segmentation within table area. Next, rule-based row extraction is used to extract data into individual table cells. The model is trained using a multi-task loss function which is a combination of table loss and column loss. The model takes a single input image and produces two output images labeled with different semantics for tables and columns. The models share the VGG19 coding layer for the table and column detectors, while the decoders for the two tasks are separate. The shared common level is trained repeatedly from the gradient received from the table and column detectors while the decoder is trained independently. Semantic information about the underlying data type is then used to further improve model performance. Using VGG19 as the core network, pre-trained over one million images from the ImageNet database. ImageNet datasets enable the exploitation of prior knowledge in the form of low-level features learned through training on ImageNet.

2.6 Hybrid Approach

We followed the same approach as in the DL approach, but instead of training the image using original images we used images with word patches masked as shown in Fig. 5.

3 Results and Discussions

In this section, we discuss the experimental results for table segmentation on Marmot dataset using the three approaches discussed in Sect. 2.3. Table 3 provides a comparison of precision, recall, and F1-score at word-level classification using all three approaches.

Among the ML classifiers, SVM with RBF kernel gave the best word-wise F1-score of 83.1% on test images. Linear Regression resulted in lesser performance since they do not add any non-linear transformation which might be important for table detection task. This should also explain why SVM with linear kernel gave lesser performance compared to SVM with RBF kernel.

Table segmentation using TableNet gave a precision, recall, and pixel-wise F1-score of 99.9%, 90.08%, and 94.78%, respectively.



Fig. 5 Sample image from Marmot dataset with segmented word patches used for hybrid approach

Table 3 Comparison of different binary classifiers' performance on classifying each word as table versus non-table

	Precision	Recall	F1-score
Logistic Regression	85.2	67.8	75.5
SVM Linear	84.2	66.9	74.6
SVM RBF	91.4	76.2	83.1
Hybrid	99.98	94.33	97.07

The original TableNet model was trained with all three RGB channels of the image. However, document images are binary in nature and further the details of character shapes are immaterial for table task. We anticipate better results in binary images as the number of possible states for each pixel is reduced to two, whether the pixel is part of word patch or not. This prompted us to modify the input image with a binary mask image with word patches segmented resulting in a hybrid approach. Using this approach the pixel-wise recall increased by 0.7% which resulted in an increased F1-score of 95.2%.

Figure 6 shows the sample prediction results using SVM. The words predicted as table words are marked in green and non-table words are marked in blue. Figure 7 shows the table mask predicted by the hybrid deep learning model for a sample test image. Figure 8 shows the table masks predicted by the TableNet model. It is evident from Figs. 7 and 8 that the boundaries of the predicted table mask using hybrid model are more sharper and rectangular as compared to the prediction of TableNet model.

It is observed that the proposed hybrid approach outperformed the vanilla TableNet model. The assumption of hybrid approach is that the gray values of pixels in a document image are distributed bimodal. This hybrid approach can further be improved with the addition of additional word-level features. Apart from this CRF can also be experimented and is expected to outperform SVM-based classifiers as CRF models the correlation between labels of neighboring words. Though we have used Marmot data, further research can be done with TableBank [7] dataset which contains much larger number of document images with tables.

4 Conclusion

In this study, we propose new word-level features that can be used for detecting tabular structures in document images. We illustrate the significance of the proposed features by training binary classifier, Logistic Regression, and SVM, to predict whether each word belongs to a table or not. We get an F1-score of 83.1% for SVM with RBF kernel on Marmot dataset. We further propose a mechanism to augment a popular deep learning model, TableNet, by providing word segmentation information at the input. The proposed augmented model outperforms the original TableNet model with

among the first n report weeks times is rise generally found among the sites with the earliest reports Table 11 also shows the median number of weeks until the cumulative number of DD cases exceeded 5

Table 11: The number of weeks from week 40 onwards to the first laboratory diagnosis influenza cases in the regions was sorted with respect to the median week of the first case. The median number of weeks until the cumulative number of DD cases exceeded 5 is shown in the first column

	0-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45	45-50	50-55	55-60	60-65	65-70	70-75	75-80	80-85	85-90	90-95	95-100
Goteborg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
KS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Umea	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Malmo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boras	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Skovde	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lund	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Uppsala	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Halmstad	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Orrebro	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Helsingborg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Karlstad	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lulea	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Salou	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Jonkopings	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Svedhemstad	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Uddevalla	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sundsvall	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Linkoping	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Eksholmen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Vasteras	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kalmar	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Varmland	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Uppsala	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Vaxjo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Uppsala	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Since the settlement areas of the laboratories differ the reason that the larger sites reach a larger cumulative sum than the smaller sites could be either that the outbreak occurs earlier in the larger sites or that the probability of a large number is greater for a large population, or a combination of the two (this question will be further studied below)

Spatial analysis with numerous clusters However regional data on influenza in Sweden are available only for 25 large regions which was found unsuitable for standard cluster analysis Thus we studied the possible spread of neighbouring areas by analysing how the geographical position indicated by latitude and longitude is associated with the time of the outbreak Table 12 shows the correlations between the coordinates and the number of weeks until the number of DD cases exceeded 5 None of these correlations differed significantly from zero

Fig. 6 Prediction results using SVM with RBF kernel. Table words are marked in green and non-table words are marked in blue

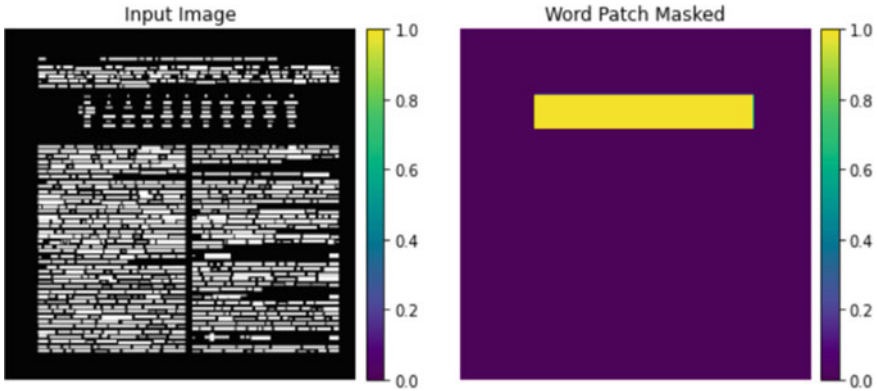


Fig. 7 Pixel-level prediction of table mask using hybrid model

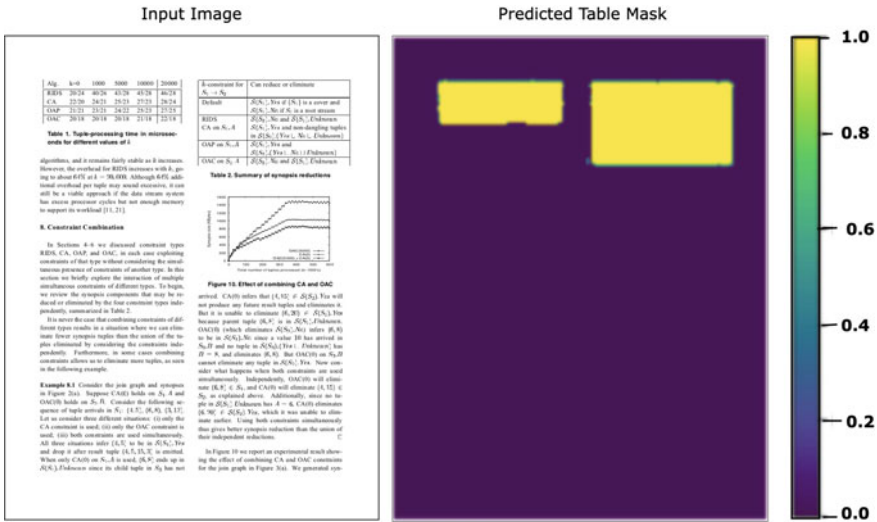


Fig. 8 Pixel-level prediction of table mask using TableNet model

an increase in recall by 0.7%. This resulted in an increase in F1-score from 94.8% to 95.2%. The hybrid model can further be improved by adding more word-level features.

Acknowledgements This work is supported by Mphasis CSR grant.

References

1. Tran DN et al (2015) Table detection from document image using vertical arrangement of text blocks. *Int J Contents* 11(4):77–85
2. Ren S et al (2015) Faster r-cnn: towards real-time object detection with region proposal networks. *Adv Neural Inf Process Syst* 28:91–99
3. Redmon J et al (2016) You only look once: unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*
4. Deng J et al (2009) Imagenet: a large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*. IEEE
5. Fang J et al (2012) Dataset, ground-truth and performance metrics for table detection evaluation. *2012 10th IAPR international workshop on document analysis systems*. IEEE
6. Zhong X, Tang J, Jimeno Yepes A (2019) Publaynet: largest dataset ever for document layout analysis. *2019 international conference on document analysis and recognition (ICDAR)*. IEEE
7. Li M et al (2019) Tablebank: a benchmark dataset for table detection and recognition. *arXiv preprint [arXiv:1903.01949](https://arxiv.org/abs/1903.01949)*
8. Mandal S et al (2006) A simple and effective table detection system from document images. *Int J Document Anal Recogn (IJ DAR)* 8(2):172–182
9. Kasar T et al (2013) Learning to detect tables in scanned document images using line information. *2013 12th international conference on document analysis and recognition*. IEEE
10. Embley DW et al (2006) Table-processing paradigms: a research survey. *Int J Document Anal Recogn (IJ DAR)* 8(2): 66–86
11. Anh TT, In-Seop N, Soo-Hyung K (2015) A hybrid method for table detection from document image. *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*. IEEE
12. Gilani A et al (2017) Table detection using deep learning. *2017 14th IAPR international conference on document analysis and recognition (ICDAR), Vol 1*. IEEE
13. Paliwal SS et al (2019) Tablenet: deep learning model for end-to-end table detection and tabular data extraction from scanned document images. *2019 international conference on document analysis and recognition (ICDAR)*. IEEE

Workshop on Computer Vision Applications (WCVA)

The Ikshana Hypothesis of Human Scene Understanding



Venkata Satya Sai Ajay Daliparthi 

1 Introduction

The human brain can seamlessly perceive diverse perceptual and semantic information regarding the natural scene/image during a glance [21, 30, 58, 62]. The visual scene information perceived during/after a glance refers to the gist (a summary) of the scene/image. The gist includes all the visual information from the low-level (e.g., colors and contours) to the high-level (e.g., shapes and activation). Due to this reason, [55] suggested that the gist can be investigated at both the perceptual and conceptual levels. The structural representation of the image refers to the perceptual gist, and the semantic information of the image refers to the conceptual gist. However, the conceptual gist is more refined and modified than the perceptual gist [55]. Several works [4, 19, 20, 26, 57, 58, 65] in neuroscience have addressed the fundamental question, i.e., “how does the human brain performs several visual tasks?” by investigating through conceptual and perceptual gist. They conducted several experiments and proposed various theories to explain how modeling of the scene occurs in the human brain. However, there was no general principle that explains the functioning of the human brain. Even though there is a general principle, we expect that to be different from human-to-human. Depending on the situation and the environment, the human brain can seamlessly grasp the information by recognizing the objects and observing their structure. On the other hand, for a computer to do the same is the fundamental goal of the computer vision field.

In recent years, deep learning methods have shown a significant improvement over traditional handcrafted techniques on several computer vision tasks. Though these deep neural networks (DNNs) achieved state-of-the-art performance in many cases, the one major drawback is the requirement of massive labeled data. The collection of a huge amount of labeled data is an expensive and time taking process. Even though

V. S. S. A. Daliparthi (✉)

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

e-mail: veda18@student.bth.se

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022
U. Mudenagudi et al. (eds.), *Proceedings of the Satellite Workshops of ICVGIP 2021*,
Lecture Notes in Electrical Engineering 924,
https://doi.org/10.1007/978-981-19-4136-8_12

these DNNs are said to be inspired by the functioning of the human brain, is this how the human brain learns to perform any visual task? **NO**. Because the human brain does not require massive labeled data to perform any visual task, and it can perform with few data samples. However, we cannot observe a similar phenomenon in the case of many DNNs.

Semantic segmentation is the task of assigning a class label to every pixel in the given image, which has applications in various fields such as medical, autonomous driving, robotic navigation, localization, and scene understanding. The prominent work FCN [48] adopted the image-classification networks [35, 75, 81] for semantic segmentation. Later on, several works [2, 12, 67, 78, 83, 98, 100, 107] improved the FCN [48] architecture, and proven to be successful in diverse semantic segmentation benchmarks [5, 15, 108]. However, these methods mainly focus on achieving state-of-the-art performance by using the entire and additional datasets [16] (for pre-training). Due to this reason, even though various methods [12, 78] outperformed U-Net [67] in terms of accuracy and computational complexity, the U-Net [67] architecture is still exploited in several medical image segmentation methods due to its ability to perform with few data samples [74]. Although several few-shot semantic segmentation (FSS) methods are introduced to address this problem, they often use techniques such as meta-learning [18, 59, 64, 85, 93] and metric learning [73, 89, 89, 90, 95, 101, 102, 106] on top of the existing architectures.

Unlike FSS methods, we tackle the formerly mentioned drawback of the DNNs, i.e., the requirement of massive labeled data, from a neuroscience perspective. In this work, we propose a hypothesis of human scene understanding mechanism named Ikshana. The idea is that, “to understand the conceptual gist of a given image; humans look at the image multiple times recurrently at different scales”. Following the Ikshana hypothesis, we propose a novel neural-inspired CNN architecture named IkshanaNet, a multi-scale architecture that learns representations at full image resolution. In contrast to the existing CNN architectures that pass the input image only to the initial layer (stem module), our method feeds the input image to every module in the network and to the best of our knowledge, this is the first work to propose the same.

To evaluate the performance of IkshanaNet, we conduct extensive experiments on the entire and subsets of the Cityscapes and Camvid benchmarks. Moreover, we conduct multiple ablation studies to verify the effect of image scales in IkshanaNet. The empirical results illustrate that our method outperforms several baselines on the entire and few data samples. Furthermore, the ablation studies shows the importance of multi-scale information in achieving considerable performance. We hope that our hypothesis sparks future research in neural network architectures for vision tasks.

2 Related Work

In **Neurological** terms, all the low-level and high-level computer vision tasks come under a single term called human scene understanding. A scene is a view of a real-world environment that contains multiple surfaces and objects organized in a mean-

ingful way. In neuroscience, the perceptual gist is more investigated compared to the conceptual gist. The early works on the conceptual gist [30, 61] explained that a typical scene fixation of 275 to 300 ms is often sufficient to understand the gist of the image. Several works on the perceptual gist [4, 19, 20, 26, 56–58, 65, 72] provided insight into how the modelling of the scene occurs in the human brain through perceiving boundaries, blobs, scales, texture, contours, openness, depth, and so on. The information perceived through the perceptual gist is refined and extracted into the conceptual gist (the semantic meaning) during the cognitive process. Thus, the conceptual gist is highly dependent upon the perceptual gist. In many cases [15, 16, 108], we do not explicitly encode the perceptual process in DNNs, and the CNN learns various representations regarding the image during the training process. Thus, our hypothesis focuses on the conceptual gist rather than the perceptual gist.

Neural networks exist from a long time [50, 68, 70] and some prominent works [14, 16, 25, 35, 43, 75, 81, 82] made them popular during recent years. In our work, we use the convolutional neural network (CNN) architecture [36, 88] to learn representations from the images, which itself is inspired by [23, 29]. The architecture of IkshanaNet is inspired by [28, 75] and related to [27, 39].

The first seminal work on **Semantic segmentation (SS)** using deep learning is the fully convolutional networks (FCN) [48]. Later on, many semantic segmentation networks followed the FCN [48] architecture. The total prominent works on deep learning-based semantic segmentation methods can be roughly classified into five categories. They are (i) Encoder-decoder based methods (DeconvNet [54], SegNet [2], U-Net [67], RefineNet [41, 42], FC-DenseNet [33], and GFR-Net [1]), (ii) Regional proposal methods (MaskRCNN [24], FPN [44], and PANet [46]), (iii) Increased resolution of feature map methods (DeepLab series [8–10, 12], PSPNet [107], DenseASPP [96], and HRNet [78]), (iv) Context information methods (ParseNet [47], ATS [11], DANet [22], OCNet [99], OCR [98], EncNet [104], Non-local [91], ZigZagNet [40], ACFNet [103], CoCurNet [105], GLAD [38], and HANet [13]) (v) Boundary refinement methods ([3, 7, 17, 49], Gated-SCNN [83], and SegFix [100]). The IkshanaNet uses the dilated convolutions, interpolation of feature maps, and skip connections from different layers in the network. Therefore, our work is related to the formerly mentioned encoder-decoder and increased resolution of feature map methods.

Few-shot segmentation (FSS) methods [6, 18, 45, 59, 64, 73, 85, 86, 89, 90, 93, 95, 101, 102, 106] are introduced to handle limited training data. They use meta-learning (knowledge distillation), metric-learning (similarity learning), and a combination of both the techniques on top of FCN [48] based architectures, which often involve multistage training. The metric-learning techniques can be further classified into the prototypical feature learning [18, 37, 87, 90, 102, 106] and the affinity learning [89, 97, 101] techniques. Unlike general SS methods, FSS methods are evaluated on different benchmarks and handle novel class categories during testing. Since the IkshanaNet does not use any of the formerly mentioned FSS techniques and only handles the classes seen in the training data, our method is more closely related to the general SS methods than the FSS methods.

3 Method

3.1 Ikshana (the Eye) Hypothesis

In her prominent work [61], professor Mary C. Potter found that an average human can understand the gist of the image between the time interval of 125 to 300 ms. Furthermore, through several works [19, 20, 26, 30, 57, 58, 65, 72] in neuroscience, it is evident that humans understand the gist of the image in a certain time interval. During that time interval, the Ikshana hypothesis approximates the functioning of the human brain. The Ikshana hypothesis states that **“To understand the conceptual gist of a given image, humans look at the image multiple times recurrently, at different scales.”** The word Ikshana is derived from the Sanskrit language, which has many synonyms such as the eye, sight, look, and so on.

We present an example to explain the Ikshana hypothesis in Fig. 1, where there is an image (x) on the left side and the human brain mechanism on the right side. According to the Ikshana hypothesis, for a human to understand the conceptual gist of the given image, the following process occurs in the human brain:

At a time step (t), during the first glance (Φ_1), the brain learns the first representation ($f(x)$) from the image (x) and stores that representation in the memory (M), as shown in the Eq. 1.

$$f(x) = \Phi_1(x); \quad M = f(x) \quad (1)$$

At a time step ($t + 1$), during the second glance (Φ_2), the brain holds the first representation ($f(x)$) in the memory and learns the second representation ($g(x)$) from the image and the first representation ($x, f(x)$). Then the brain stores the representation ($g(x)$) along with ($f(x)$) in the memory (M), as shown in the Eq. 2.

$$g(x) = \Phi_2(x, f(x)); \quad M = f(x), g(x) \quad (2)$$

At a time step ($t + 2$), during the third glance (Φ_3), the brain holds the first and the second representations ($f(x), g(x)$) in the memory and learns the third representation ($h(x)$) from the image and the previous representations ($x, f(x), g(x)$). Then the brain stores the representation ($h(x)$) along with ($f(x), g(x)$) in the memory (M), as shown in the Eq. 3.

$$h(x) = \Phi_3(x, f(x), g(x)); \quad M = f(x), g(x), h(x) \quad (3)$$

From Eqs. 1, 2, and 3, this kind of recurrent process occurs at ($t + n$) times at a single image scale. Depending upon the given task (T), by combing all the information stored in the memory until the ($t + n$)th time step, the brain understands the conceptual gist (Y_1) of the image at a single scale, as shown in the Eq. 4.

$$Y_1 = T(f(x), g(x), h(x), \dots, n(x)) \quad (4)$$

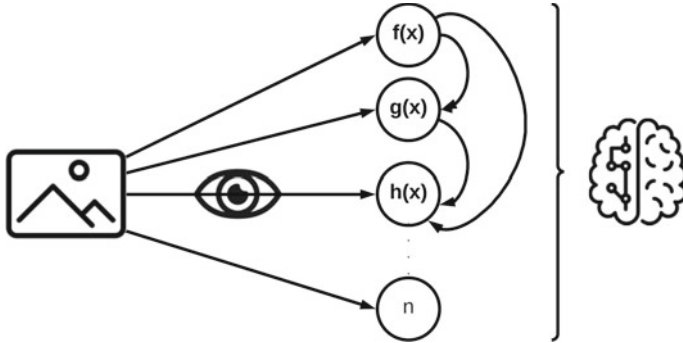


Fig. 1 The Ikshana Hypothesis at single scale

This process occurs at N different scales and generates N different outputs ($Y_1, Y_2, Y_3, \dots, Y_n$). By considering all the outputs, the brain selects some of those representations and forgets the remaining representations. In this way, the brain learns (Δ) the final output (Y) of the given visual task (T), as shown in the Eq. 5.

$$Y = \Delta(Y_1, Y_2, Y_3, \dots, Y_N) \tag{5}$$

From the Eqs. 1, 2, 3, 4, and 5, this is how Ikshana hypothesis approximates the functioning of the human brain, while human understands the conceptual gist of the image. The time required (or the number of glances required) by an average human to understand the gist of the image may depend upon several factors such as the given task, age, intelligence, memory, and so on.

The existing CNN architectures such as VGG [75], Resnet [25], DenseNet [28], and so on learns a representation (say $f(x)$) with 32/64 filters from the input image and learns further representations on top of the $f(x)$ until the network achieves adequate performance. In contrast, the network designed by following the Ikshana hypothesis learns representations from the input image and previous outputs at each glance/layer.

3.2 *IkshanaNet Architecture*

In this section, we introduce a novel neural-inspired encoder-decoder CNN architecture named IkshanaNet, designed by following the Ikshana hypothesis. Humans can look at the image and seamlessly learn various useful representations regarding it [21, 30, 58, 62]. On the other hand, for a computer to do the same, we use the convolutional neural network [23, 29, 36] architecture to learn representations. The IkshanaNet architecture uses three image scales and consists of 4M parameters. The entire architecture is made of three building blocks, and they are: (1) the glance

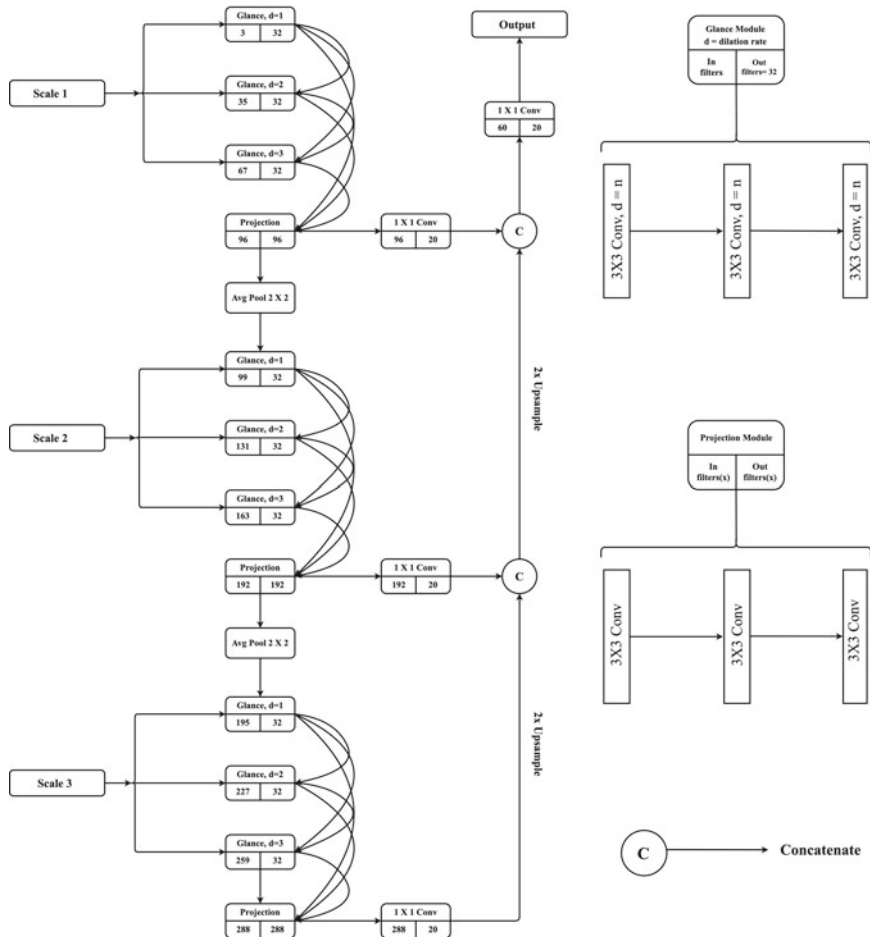


Fig. 2 IkshanaNet-main architecture

module, (2) the projection module, and (3) a 1×1 convolutional layer, as illustrated in Fig. 2.

The **glance module** consists of three 3×3 convolutional layers (with the same dilation rates), and we use it to learn representations from the given image (or a feature map). The number of input filters passed into the glance module varies several times in the architecture; however, it always returns a feature map with 32 filters. The **projection module** consists of three 3×3 convolutional layers, and we use it to refine the representations learned from the glance modules. The input and output filters are always the same for the projection module. We use the **1×1 convolution layers** to reduce the number of filters in a given future map. Except for the last 1×1 convolutional layer that returns the final output, every convolutional layer in the

architecture is followed by a batch normalization [31] and a ReLU [52] activation layer.

In the **encoder** part, the IkshanaNet learns representations at three image scales. At scale 1, we pass the input image through a glance module with a dilation rate ($d = 1$), which returns a feature map with 32 filters. Then we concatenate the input image with the previously learned feature map ($32 + 3 = 35$). The concatenation of the input image with the feature map is essential to ensure that we are learning representations from the input image. Then we pass the feature map through another glance module with a dilation rate ($d = 2$) and concatenate the resulting feature map with the feature maps from the preceding layers ($32 + 32 + 3 = 67$). We pass the resulting feature map through another glance module with a dilation rate ($d = 3$), which takes in 67 filters and returns 32 filters. Again, we concatenate the resulting feature map with feature maps from the preceding layers ($32 + 32 + 32 + 3 = 99$). At this point, we remove the input image from the feature map through tensor slicing ($99 - 3 = 96$), and the resulting feature map consists of ($32 + 32 + 32 = 96$) filters learned from three glances modules. In this way, the network followed the Ikshana hypothesis had three glances recurrently at the full resolution. Then we pass the feature map through a projection module to refine the representations ($96 = 96$). Here, we pass the refined feature map through a 1×1 convolutional layer that reduces 96 filters into 20 filters and name it the side one output (Y_1). Simultaneously, we pass the feature map through an average pooling layer, which reduces the size of the feature map by a factor of two.

At scale 2, we down-sample the input image by a factor of two and concatenate with the pooled feature map from the scale 1 ($96 + 3 = 99$). We pass the resulting feature map with 99 filters through three glance modules with different dilation rates ($d = 1, 2, 3$) and concatenate all the outputs as follows ($99 + 32 + 32 + 32 = 195$). Then we remove the image from the feature map ($195 - 3 = 192$) and pass it through a projection module to refine the representations ($192 = 192$). Then we pass the refined feature map through a 1×1 convolutional layer that reduces 192 filters into 20 filters and name it the side two output (Y_2). Then, we pass the refined feature map through an average pooling layer that reduces the size by a factor of two.

At scale 3, we down-sample the input image by a factor of four and concatenate with the pooled feature map from the scale 2 ($192 + 3 = 195$). Here, we follow the same process ($195 + 32 + 32 + 32 = 291$); ($291 - 3 = 288$); ($288 = 288$) as the scale 2 part, which returns a feature map with 20 filters, and name it the side three output (Y_3).

In the **decoder** part, we bi-linearly interpolate the outputs from two scales (Y_2 and Y_3) to match with the output of scale 1 Y_1 , i.e., the input image size. Then we concatenate all the three outputs ($20 + 20 + 20 = 60$) and pass it through a 1×1 convolutional layer, which returns a feature map with 20 filters, that is the final output of the network [$Y = \Delta(Y_1, Y_2, Y_3)$].

Depth Architectures: Here, we introduce three variants of the IkshanaNet named IkshanaNet-3G, IkshanaNet-6G, and IkshanaNet-12G. If we remove the projection layers in IkshanaNet-main, then it will remain with three scales and three glances at each scale; it is IkshanaNet-3G (which consists of 514 K parameters). If we increase

the number of glances per scale, from three to six, then it is IkshanaNet-6G (which consists of 1.8M parameters), and from three to twelve, then it is IkshanaNet-12G (which consists of 6.5M parameters).

Multi-scale Architectures: Here, we introduce three variants of IkshanaNet named IkshanaNet 1S-6G, 2S-3G, and 3S-2G. In IkshanaNet 1S-6G, there are no pooling layers and contain six glances at full-scale resolution (which consists of 257K parameters). In IkshanaNet 2S-3G, there are two scales and three glances at each scale (which consists of 259K parameters). In IkshanaNet 3S-2G, there are three scales and two glances at each scale (which consists of 260 K parameters).

4 Experiments

4.1 Experimental Setup

GPU: 1 X NVIDIA Tesla T-4 (16 GB VRAM)

Framework: PyTorch 1.8 [60]

Epochs: 180 ; **Batch Size:** 2

Criterion: Pixel-wise cross-entropy loss

Learning Rate Scheduler: ReduceLROnPlateau (decrease factor = 0.5 and patience = 20 epochs) with an initial learning rate of $1e - 06$.

Optimizer: Stochastic gradient descent [66] with Nesterov momentum [53]¹

Random Seed: To ensure that data splits are reproducible, we set the random seed 42 in the function `torch.utils.data.random-split`.

Pre-Processing: We normalize all the images with mean and standard deviation values of ImageNet [16] dataset. We did not use any data augmentation techniques.

Baselines: We use the open-source implementations for networks DeepLabV3+ (ResNet-101) [32], DeepLabV3 (DenseNet-161) [77], HRNet-V2 [79], and U-Net [51]. We import DeeplabV3+ with encoder networks such as ResNet [25], MobileNet-V2 [71], ResNext [92], EfficientNet [84], and RegNet [63] from the segmentation models library [94].

4.2 Experiments on Cityscapes

The Cityscapes [15] semantic segmentation dataset consists of 5, 000 finely annotated high-quality images, which are further divided into 2, 975/500/1, 525 images for training, validation, and testing. During the evaluation, only 19 classes are considered

¹ For all the baselines, we use the Nesterov momentum of 0.9 for the SGD [66] optimizer by following [12, 25, 28, 63, 71, 84]. For the IkshanaNet and its variants, we use the Nesterov momentum of 0.7 for the SGD [66] optimizer by tuning with several values such as 0.5, 0.6, 0.7, 0.8, and 0.9, i.e., the only hyper-parameter tuning step in this work. In our preliminary experiments, we observe that the training of IkshanaNet is unstable with 0.9 momentum. We hypothesize that this phenomenon is due to the small size of IkshanaNet compared to baseline networks.

out of the 35 classes. Therefore, by using the cityscapes-scripts, we convert the 35 classes into 20 classes (including background). We resize all the images from the resolution of 1024×2048 to 512×1024 .

4.2.1 Baseline Experiments

Here, we use the networks DeeplabV3+ (ResNet-101 [25]), DeeplabV3 (DenseNet-161 [28]), HRNet-V2 [80], and U-Net [67] as the baselines² to compare with IkshanaNet-main.

We train all the networks on the entire dataset T_{2975} and provide the mean class IoU results evaluated on the validation-set in Table 1, where we observe the following:

- (i) U-Net [67] (49.3) shown top performance within the baseline networks followed by HRNet-V2 [80] (48.0).
- (ii) IkshanaNet outperformed U-Net by 5.2 % and HRNet-V2 [80] by 6.5 %.
- (iii) IkshanaNet outperformed baselines by a huge margin in classes such as fence, pole, traffic light, traffic sign, rider, bus, motorcycle, and bicycle.
- (iv) Even though U-Net [67] and IkshanaNet learn representations at full-scale resolution before reducing the spatial resolution, the IkshanaNet still outperforms U-Net [67] in the formerly mentioned classes.

4.2.2 Data Ablation Study

While trained on few data samples, the network size might strongly influence the performance. The networks ResNet-101 [25] (59.3 M), DenseNet-161 [28] (43.2 M), HRNet-V2 [80] (65.9), and U-Net [67] (31.0) consists more number of parameters compared to IkshanaNet-main (4 M). To make it a fair comparison, we include DeeplabV3+ [12] with several light-weight encoder networks (such as ResNet-18 [25], MobileNet-V2 [71], EfficientNet-b1 [84], and RegNetY-08 [63]) along with the networks from the baseline experiments.

Here, we conduct a data ablation study on five different subsets of the training data, T_{1487} , T_{743} , T_{371} , T_{185} , and T_{92} (suffix number represents the number of training samples in the subset) by using the same validation set (500 images).

In Table 2, we provide the mean class IoU results evaluated on the validation set, the average M.IoU score, the number of parameters (in million), and the GFLOPs [76] (calculated with an input resolution of $1 \times 512 \times 1024 \times 3$).

From Table 2, we observe the following:

² For the baselines, ResNet-101 [25], DenseNet-161 [28], and HRNet-V2 [80], we use the ImageNet [16] pre-trained weights. Because in the existing literature, the architectures [12, 80, 98, 107] used an ImageNet pertained network as a feature extractor and reported the results by using pre-trained weights only. However, in the case of IkshanaNet and U-Net [67] no pre-training is done. Since this work addresses the requirement of massive data, this provides strong motivation against pre-training.

Table 1 Class-wise IoU results of the Cityscapes baseline experiments

Method	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorecycle	Bicycle	Average
ResNet101 [25]	95.0	66.1	81.9	15.0	13.5	26.7	20.7	29.5	86.7	55.4	89.3	48.5	6.3	85.5	6.8	26.1	19.0	9.8	32.0	42.8
DenseNet161 [28]	94.8	64.5	81.3	20.1	13.0	15.8	15.6	28.7	84.6	58.7	86.1	44.1	0.6	84.7	17.0	19.7	23.1	4.3	31.4	41.5
HRNet-V2 [80]	94.9	68.6	84.2	24.0	24.5	39.0	23.2	42.3	86.9	51.5	90.2	55.6	15.3	86.1	19.9	36.1	21.2	2.2	46.1	48.0
U-Net [67]	94.9	69.4	85.3	27.3	28.7	41.0	32.2	49.0	88.6	46.3	90.4	59.1	14.5	86.5	12.4	28.4	15.5	10.9	55.6	49.3
IkshanaNet-Main	95.6	72.8	85.9	22.6	35.3	49.6	47.0	60.7	89.2	48.9	91.6	63.3	28.8	87.1	18.4	40.3	21.8	16.5	60.8	54.5

Table 2 Cityscapes data ablation experiments evaluated on the validation set

Backbone	T_{1487}	T_{743}	T_{371}	T_{185}	T_{92}	T_{avg}	Param(M)	GFLOPs
ResNet-18 [25]	42.6	35.6	27.9	22.4	21.0	29.9	12.3	36.8
MobileNet-V2 [71]	38.5	32.2	30.6	22.5	19.2	28.6	4.4	12.3
EfficientNet-b1 [84]	37.8	32.5	26.9	24.6	19.8	28.3	7.4	4.6
RegNetY-08 [63]	28.5	31.9	29.4	27.4	22.1	27.9	7.0	17.2
ResNet-101 [25]	29.3	28.8	28.6	21.6	19.4	25.5	59.3	177.8
DenseNet-161 [28]	33.3	30.1	26.0	24.9	20.8	27.0	43.2	129.4
HRNet-V2 [80]	27.8	18.8	23.3	18.3	15.4	20.7	65.9	187.8
U-Net[67]	42.8	34.2	30.2	27.8	25.0	32.0	31.0	387.1
IkshanaNet-Main	43.4	40.2	31.7	29.9	25.8	34.2	4.0	413.3

- (i) U-Net [67] (T_{avg} –32.2) achieves top average performance within the baselines.
- (ii) Even though U-Net [67] consists of 31M parameters, it still managed to outperform its lightweight counterparts.
- (iii) IkshanaNet outperformed all other baselines in the M.IoU score and the average M.IoU score in all five subsets.
- (iv) IkshanaNet consists of fewer parameters, and EfficientNet-b1 [84] consists of fewer GFLOPs than other networks.

4.2.3 Multi-scale Ablation Study

In Sect. 3.1, the Ikshana hypothesis stated that “humans often require multi-scale information to understand the gist of an image”. Therefore, to verify the requirement of multi-scale information, we conduct a multi-scale ablation study.

Here, we train three different variants of IkshanaNet, such as the 1S-6G, 2S-3G, and 3S-2G (explained in Sect. 3.2) on the five different subsets of the training data (same as Sect. 4.2.2). In Table 3, we provide the results of the multi-scale ablation study evaluated on the validation set.

From Table 3, we observe that:

- (i) IkshanaNet-3S-2G network outperforms other networks in the M.IoU score, the average M.IoU score, and requires fewer GFLOPs, while requiring the same number of parameters.

Table 3 Cityscapes multi-scale ablation experiments results

Backbone	T_{1487}	T_{743}	T_{371}	T_{185}	T_{92}	T_{avg}	Param(M)	GFLOPs
1S-6Glances	29.2	24.9	23.3	20.2	18.1	23.1	0.26	136.0
2S-3Glances	37.3	34.9	33.2	25.7	24.0	31.0	0.26	70.0
3S-2Glances	43.5	36.9	34.4	27.5	26.5	33.8	0.26	42.4

- (ii) The multi-scale information improved the performance and decreased the computational complexity (GFLOPs) of the network and vice-versa.
- (iii) From Tables 2 and 3, we observe that IkshanaNet 3S-2G network (with only 260K parameters) outperforms all the baselines in the data ablation study by occupying approximately 10x few GFLOPs and 15x few parameters than IkshanaNet-main.

The above observations suggest that, the multi-scale architectures can achieve superior performance than an isometric architecture.

4.3 Experiments on Camvid

The Cambridge-driving labeled video dataset [5] for semantic segmentation consists of 700 images, which are further divided into 367 training, 101 validation, and 233 testing sets. We convert the 32 classes to 12 classes (including background) by following [2, 34] and resize the images from the resolution of 720×960 to 368×480 .

4.3.1 Baseline Experiments

Here, according to the size of the networks, we classify the total networks into three different sets.

Set-1 consists of DeeplabV3+ [12] with the encoder networks such as Resnet-18 [25], EfficientNet-b1 [84], RegNetY-08 [63], MobileNet-V2 [71], and IkshanaNet-3G (see Sect. 3.2).

Set-2 consists of DeeplabV3+ [12] with the encoder networks such as Resnet-50 [25], EfficientNet-b4 [84], RegNetY-40 [63], and ResNext-50 [92], and IkshanaNet-6G (see Sect. 3.2).

Table 4 Camvid baseline experiments results

Backbone	T_{367}		T_{183}		T_{91}		T_{avg}		Param(M)	GFLOPs
	Val	Test	Val	Test	Val	Test	Val	Test		
ResNet-18 [25]	83.3	64.9	79.7	63.7	70.0	56.6	77.7	61.7	12.3	12.4
EfficientNet-b1 [84]	84.4	68.4	75.0	61.3	77.0	58.8	78.8	62.8	7.4	1.5
RegNetY-08 [63]	80.4	64.3	77.7	61.4	70.9	57.8	76.3	61.2	7.0	5.8
MobileNet-V2 [71]	80.8	63.9	77.3	56.1	66.1	54.6	74.7	58.2	4.4	4.1
IkshanaNet-3G	81.6	65.7	80.0	62.5	78.0	61.2	79.9	63.1	0.5	26.0
ResNet-50 [25]	78.6	61.6	80.0	60.3	78.3	55.9	80.0	59.3	26.7	25.0
EfficientNet-b4 [84]	82.7	64.1	77.7	62.2	75.6	60.5	78.7	62.3	18.6	1.7
RegNetY-40 [63]	80.8	62.0	76.4	61.0	74.9	59.2	77.4	60.7	21.5	18.8
ResNext-50 [92]	80.1	62.6	77.3	56.1	66.1	54.6	74.5	57.8	26.2	25.0
IkshanaNet-6G	83.3	67.8	81.4	65.9	76.0	60.0	80.2	64.6	1.8	82.0
ResNet-101 [25]	81.6	63.8	75.6	56.4	70.1	55.7	75.8	58.6	59.3	59.9
EfficientNet-b6 [84]	80.6	65.0	80.3	57.8	77.4	60.4	79.4	61.0	42.0	1.9
RegNetY-80 [63]	78.5	62.0	78.2	63.8	66.2	53.8	74.3	59.9	40.3	34.4
DenseNet-161 [28]	77.8	58.6	75.7	57.8	73.0	53.8	75.5	56.7	43.2	43.6
HRNet-V2 [80]	81.1	63.6	79.1	62.9	72.9	55.0	77.7	60.5	65.9	63.5
U-Net [67]	83.0	69.5	78.0	62.8	76.8	61.6	79.3	64.6	31.0	130.0
IkshanaNet-12G	83.9	70.0	83.3	67.1	76.5	60.6	81.2	65.9	6.5	285.0
IkshanaNet-M	83.2	68.5	79.9	62.9	72.2	58.8	78.4	63.4	4.0	139.0

Set-3 consists of DeeplabV3+ [12] with the encoder networks such as Resnet-101 [25], EfficientNet-b6 [84], RegNetY-80 [63], DeepLabV3 (DenseNet-161 [28]), HRNet-V2 [80], U-Net [67], and IkshanaNet-12G (see Sect. 3.2) ³.

Additionally, we include IkshanaNet-main and did not compare it with other networks. By using the same validation, we train each network on three different subsets of the training data, T_{367} , T_{183} , and T_{91} .

In Table 4, we provide the mean IoU results evaluated on the validation set, the test set, the average M.IoU score of all the variants, the number of parameters (in Million), and the GFLOPs [76] (calculated the GFLOPs with an input resolution of $1 \times 368 \times 480 \times 3$). From Table 4, we observe the following:

In **Set-1**: (i) IkshanaNet-3G outperforms all other networks in the subsets T_{91} , T_{avg} , and requires fewer parameters. (ii) EfficientNet-b1 [84] outperforms other networks in the T_{367} and requires fewer GFLOPs.

In **Set-2**: (i) IkshanaNet-6G outperforms all other networks in the subsets T_{367} , T_{183} , T_{avg} , and requires fewer parameters.

³ Same as Sect. 4.2.1, except for U-Net [67] and IkshanaNet-12G, we use the ImageNet [16] pre-trained weights for all the networks in the Set-3.

Table 5 Camvid multi-scale ablation experiments results

Backbone	T_{367}		T_{183}		T_{91}		T_{avg}		Param(M)	GFLOPs
	Val	Test	Val	Test	Val	Test	Val	Test		
1S-6Glances	79.2	60.0	77.8	58.8	66.7	50.9	74.6	56.6	0.26	45.6
2S-3Glances	80.1	65.6	79.5	60.1	77.2	59.5	78.9	61.7	0.26	23.1
3S-2Glances	82.9	66.5	80.9	62.8	77.5	60.8	80.4	63.4	0.26	14.0

(ii) EfficientNet-b4 [84] outperforms all other networks in the subset T_{91} and requires fewer GFLOPs.

In **Set-3**: (i) IkshanaNet-12G outperforms all other networks in the subsets T_{367} , T_{183} , T_{avg} , and requires fewer parameters.

(ii) U-Net [67] outperformed other networks in the subset T_{91} and EfficientNet-b6 [84] requires fewer GFLOPs than other networks.

4.3.2 Multi-scale Ablation Study

Same as Sect. 4.2.3, by using the same validation set, we train three different variants of IkshanaNet such as 1S-6G, 2S-3G, and 3S-2G (explained in Sect. 3.2) on three subsets of the training data (T_{367} , T_{183} , and T_{91}).

In Table 5, we provide the mean IoU results evaluated on the validation set, the test set, the average score of all variants, the parameters, and the GFLOPs. We calculate the GFLOPs with an input resolution of 1x368x480x3.

From Table 5, we observe that, the IkshanaNet-3S-2G network outperforms all other networks in all the subsets (T_{367} , T_{183} , T_{91} , T_{avg}), and requires fewer GFLOPs. The results are similar to the Sect. 4.2.3 (Table 3), demonstrating the importance of multi-scale information.

5 Validity Threats

- (i) Most of the existing works [12, 98, 107] used a mini-batch size of 8 and SyncBN [69, 104] for training. However, due to the limited availability of the computing resources, we train all the networks with a mini-batch size of 2. Due to this reason, we cannot directly compare the performance of our method with the state-of-the-art methods.
- (ii) In this work, even though the training data splits are reproducible, the performance of the networks trained on subsets of the training data might depend upon the fact that “how well the subset represents the whole dataset?”. If we use a

different random seed to generate the splits, then the exact behavior may or may not be expected.

- (iii) In this work, through multi-scale ablation experiments, we observe that multi-scale information is often necessary to improve the performance of the networks. By observing the images in Cityscapes, and CamVid datasets, it is evident that the images consist of multi-scale objects. However, this phenomenon might not be valid to other datasets, where there exist no multi-scale objects.

6 Conclusion

In this work, we attempt to bridge the gap between the current vision DNNs and the human visual system by proposing a novel hypothesis of human scene understanding and a neural-inspired CNN architecture that learns representations at full-scale resolution.

The empirical results illustrate the effectiveness of our method on entire and few data samples compared to the baselines. Also, through multi-scale ablation studies, we observe that using multi-scale information improves the performance of IkshanaNet by reducing the computational complexity.

Moreover, we observe that our method is just an improvement over the baselines, and it is still dependent on the data. Hence, it is nowhere close to the human visual system. Therefore, a better-performing and computationally efficient architectures based on the Ikshana hypothesis will be studied in the future work.

Furthermore, we hope that our hypothesis inspires future generation of neural inspired vision architectures.

6.1 Code

<https://github.com/dvssajay/The-Ikshana-Hypothesis-of-Human-Scene-Understanding>.

Acknowledgements I would like to thank all the anonymous reviewers and area chairs for their constructive feedback. I would like to thank Shri Daliparathi Sathi Raju Garu for providing computational resources to conduct the main experiments of this work. Also, I would like to thank Mr. Florian Westphal for providing GPUs to conduct the preliminary experiments of this work.

References

1. Amirul Islam M, Rochan M, Bruce NDB, Wang Y (2017) Gated feedback refinement network for dense image labeling. In: Computer vision and pattern recognition (CVPR). <https://doi.org/10.1109/CVPR.2017.518>
2. Badrinarayanan V, Kendall A, Cipolla R (2017) Segnet: a deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans Pattern Anal Mach Intell* 39(12):2481–2495. <https://doi.org/10.1109/TPAMI.2016.2644615>
3. Bertasius G, Shi J, Torresani L (2015) High-for-low and low-for-high: efficient boundary detection from deep object features and its applications to high-level vision. In: Proceedings of the IEEE international conference on computer vision (ICCV). <https://doi.org/10.1109/ICCV.2015.65>
4. Biederman I (1987) Recognition-by-components: a theory of human image understanding. *Psychol Rev* 94(2):115. <https://doi.org/10.1037/0033-295X.94.2.115>
5. Brostow GJ, Shotton J, Fauqueur J, Cipolla R (2008) Segmentation and recognition using structure from motion point clouds. In: *ECCV* (1), pp 44–57. https://doi.org/10.1007/978-3-540-88682-2_5
6. Cao Z, Zhang T, Diao W, Zhang Y, Lyu X, Fu K, Sun X (2019) Meta-seg: a generalized meta-learning framework for multi-class few-shot semantic segmentation. *IEEE Access* 7:166109–166121. <https://doi.org/10.1109/ACCESS.2019.2953465>
7. Chen LC, Barron JT, Papandreou G, Murphy K, Yuille AL (2016) Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In: *CVPR*. <https://doi.org/10.1109/CVPR.2016.492>
8. Chen L, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2015) Semantic image segmentation with deep convolutional nets and fully connected crfs. In: 3rd international conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, conference track proceedings. [arXiv:1412.7062](https://arxiv.org/abs/1412.7062)
9. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2018) Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans Pattern Anal Mach Intell* 40(4):834–848. <https://doi.org/10.1109/TPAMI.2017.2699184>
10. Chen L, Papandreou G, Schroff F, Adam H (2017) Rethinking atrous convolution for semantic image segmentation. [arXiv:1706.05587](https://arxiv.org/abs/1706.05587)
11. Chen LC, Yang Y, Wang J, Xu W, Yuille AL (2016) Attention to scale: Scale-aware semantic image segmentation. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 3640–3649. <https://doi.org/10.1109/CVPR.2016.396>
12. Chen LC, Zhu Y, Papandreou G, Schroff F, Adam H (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp 801–818 (2018). https://doi.org/10.1007/978-3-030-01234-2_49
13. Choi S, Kim JT, Choo J (2020) Cars can't fly up in the sky: improving urban-scene segmentation via height-driven attention networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9373–9383. <https://doi.org/10.1109/CVPR42600.2020.00939>
14. Chollet F (2017) Xception: seep learning with depthwise separable convolutions. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 1800–1807. <https://doi.org/10.1109/CVPR.2017.195>
15. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3213–3223. <https://doi.org/10.1109/CVPR.2016.350>
16. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp 248–255. <https://doi.org/10.1109/cvprw.2009.5206848>

17. Ding H, Jiang X, Liu AQ, Thalmann NM, Wang G (2019) Boundary-aware feature propagation for scene segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV). <https://doi.org/10.1109/ICCV.2019.00692>
18. Dong N, Xing EP (2018) Few-shot semantic segmentation with prototype learning. In: BMVC, vol 3
19. Evans KK, Treisman A (2005) Perception of objects in natural scenes: is it really attention free? *J Exp Psychol Human Percept Perform* 31(6):1476. <https://doi.org/10.1037/0096-1523.31.6.1476>
20. Fei-Fei L, Iyer A, Koch C, Perona P (2007) What do we perceive in a glance of a real-world scene? *J Vision* 7(1):10–10. <https://doi.org/10.1167/7.1.10>
21. Friedman A (1979) Framing pictures: The role of knowledge in automatized encoding and memory for gist. *J Exp Psychol Gen* 108(3):316. <https://doi.org/10.1037//0096-3445.108.3.316>
22. Fu J, Liu J, Tian H, Li Y, Bao Y, Fang Z, Lu H (2019) Dual attention network for scene segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3146–3154. <https://doi.org/10.1109/CVPR.2019.00326>
23. Fukushima K, Miyake S, Ito T (1983) Neocognitron: a neural network model for a mechanism of visual pattern recognition. *IEEE Trans Syst Man Cybern* 45(5):826–834. <https://doi.org/10.1109/TSMC.1983.6313076>
24. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969. <https://doi.org/10.1109/ICCV.2017.322>
25. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778. <https://doi.org/10.1109/CVPR.2016.90>
26. Henderson JM (2003) Human gaze control during real-world scene perception. *Trends Cognit Sci* 7(11):498–504. <https://doi.org/10.1016/j.tics.2003.09.006>
27. Huang G, Chen D, Li T, Wu F, van der Maaten L, Weinberger K (2018) Multi-scale dense networks for resource efficient image classification. In: International conference on learning representations. <https://openreview.net/forum?id=Hk2aImxAb>
28. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708. <https://doi.org/10.1109/CVPR.2017.243>
29. Hubel DH, Wiesel TN (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol* 160(1):106. <https://doi.org/10.1113/jphysiol.1962.sp006837>
30. Iraub H (1981) Rapid conceptual identification of sequentially presented pictures. *J Exp Psychol Human Percept Perform* 7(3):604. <https://doi.org/10.1037/0096-1523.7.3.604>
31. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pp 448–456. PMLR
32. jfzhang95: pytorch-deeplab-xception (deeplabv3+ with resnet-101 backbone). <https://github.com/jfzhang95/pytorch-deeplab-xception>
33. Jégou S, Drozdal M, Vazquez D, Romero A, Bengio Y (2017) The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: 2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW), pp 1175–1183. <https://doi.org/10.1109/CVPRW.2017.156>
34. Kendall A: Segnet-tutorial. <https://github.com/alexgkendall/SegNet-Tutorial>
35. Krizhevsky A, Sutskever I, E. hinton, geoffrey (2012) imagenet classification with deep convolutional neural networks. *Neural Inf Process Syst* 25(10.1145), 3065386. <https://doi.org/10.5555/2999134.2999257>
36. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324. <https://doi.org/10.1109/5.726791>

37. Li G, Jampani V, Sevilla-Lara L, Sun D, Kim J, Kim J (2021) Adaptive prototype learning and allocation for few-shot segmentation. CoRR [arXiv:2104.01893](https://arxiv.org/abs/2104.01893)
38. Li X, Zhang L, You A, Yang M, Yang K, Tong Y (2019) Global aggregation then local distribution in fully convolutional networks. In: BMVC
39. Liao Q, Poggio T (2016) Bridging the gaps between residual learning, recurrent neural networks and visual cortex. arXiv preprint [arXiv:1604.03640](https://arxiv.org/abs/1604.03640)
40. Lin D, Shen D, Shen S, Ji Y, Lischinski D, Cohen-Or D, Huang H (2019) ZigzagNet: fusing top-down and bottom-up context for object segmentation. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 7482–7491. <https://doi.org/10.1109/CVPR.2019.00767>
41. Lin G, Milan A, Shen C, Reid I (2017) RefineNet: multi-path refinement networks for high-resolution semantic segmentation. In: CVPR. <https://doi.org/10.1109/CVPR.2017.549>
42. Lin G, Liu F, Milan A, Shen C, Reid I (2019) Refinenet: multi-path refinement networks for dense prediction. IEEE Trans Pattern Anal Mach Intell. <https://doi.org/10.1109/TPAMI.2019.2893630>
43. Lin M, Chen Q, Yan S (2014) Network in network. In: 2nd international conference on learning representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, conference track proceedings. [arXiv:1312.4400](https://arxiv.org/abs/1312.4400)
44. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125. <https://doi.org/10.1109/CVPR.2017.106>
45. Liu L, Cao J, Liu M, Guo Y, Chen Q, Tan M (2020) Dynamic extension nets for few-shot semantic segmentation. In: Proceedings of the 28th ACM international conference on multimedia, pp 1441–1449. MM '20, Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/3394171.3413915>
46. Liu S, Qi L, Qin H, Shi J, Jia J (2018) Path aggregation network for instance segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8759–8768. <https://doi.org/10.1109/CVPR.2018.00913>
47. Liu W, Rabinovich A, Berg AC (2015) Parsenet: looking wider to see better. CoRR [arXiv:1506.04579](https://arxiv.org/abs/1506.04579)
48. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440. <https://doi.org/10.1109/TPAMI.2016.2572683>
49. Marin D, He Z, Vajda P, Chatterjee P, Tsai SS, Yang F, Boykov Y (2019) Efficient segmentation: learning downsampling near semantic boundaries. In: 2019 IEEE/CVF international conference on computer vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019, pp 2131–2141. IEEE. <https://doi.org/10.1109/ICCV.2019.00222>
50. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. *Bullet Math Biophys* 5(4):115–133. <https://doi.org/10.1007/BF02478259>
51. milesial: Pytorch-unet. <https://github.com/milesial/Pytorch-UNet>
52. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: *Icml*
53. Nesterov YE (1983) A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In: *Dokl. akad. nauk Sssr*. vol 269, pp 543–547
54. Noh H, Hong S, Han B (2015) Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE international conference on computer vision, pp 1520–1528. <https://doi.org/10.1109/ICCV.2015.178>
55. Oliva A (2005) Gist of the scene. In: *Neurobiology of attention*, pp 251–256. Elsevier. <https://doi.org/10.1016/B978-012375731-9/50045-8>
56. Oliva A, Schyns PG (1997) Coarse blobs or fine edges? evidence that information diagnosticity changes the perception of complex visual stimuli. *Cognit Psychol* 34(1):72–107. <https://doi.org/10.1006/cogp.1997.0667>
57. Oliva A, Schyns PG (2000) Diagnostic colors mediate scene recognition. *Cognit Psychol* 41(2):176–210. <https://doi.org/10.1006/cogp.1999.0728>

58. Oliva A, Torralba A (2001) Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int J Comput Vis* 42(3):145–175. <https://doi.org/10.1023/A:1011139631724>
59. Pambala AK, Dutta T, Biswas S (2020) SML: semantic meta-learning for few-shot semantic segmentation. *CoRR* [arXiv:2009.06680](https://arxiv.org/abs/2009.06680)
60. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: an imperative style, high-performance deep learning library. In: *Advances in neural information processing systems* 32, pp 8024–8035. Curran Associates, Inc. (2019), <http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
61. Potter MC (1975) Meaning in visual search. *Science* 187(4180):965–966. <https://doi.org/10.1126/science.1145183>
62. Potter MC (1976) Short-term conceptual memory for pictures. *J Expe Psychol Human Learn Memory* 2(5):509. <https://doi.org/10.1037/0278-7393.2.5.509>
63. Radosavovic I, Kosaraju RP, Girshick R, He K, Dollár P (2020) Designing network design spaces. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 10428–10436. <https://doi.org/10.1109/cvpr42600.2020.01044>
64. Rakelly K, Shelhamer E, Darrell T, Efros AA, Levine S (2018) Conditional networks for few-shot semantic segmentation. In: *6th International Conference on Learning Representations, ICLR, Vancouver, BC, Canada, Workshop Track Proceedings*. <https://openreview.net/forum?id=SkMjFKJwG>
65. Rayner K (1998) Eye movements in reading and information processing: 20 years of research. *Psychol Bulletin* 124(3):372. <https://doi.org/10.1037/0033-2909.124.3.372>
66. Robbins H, Monro S (1951) A stochastic approximation method. *The annals of mathematical statistics*, pp 400–407. <https://doi.org/10.1214/aoms/1177729586>
67. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention*, pp 234–241. Springer. https://doi.org/10.1007/978-3-319-24574-4_28
68. Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev* 65(6):386. <https://doi.org/10.1037/H0042519>
69. Rota Bulò S, Porzi L, Kotschieder P (2018) In-place activated batchnorm for memory-optimized training of dnn. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. <https://doi.org/10.1109/CVPR.2018.00591>
70. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *Nature* 323(6088):533–536. <https://doi.org/10.1038/323533a0>
71. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) Mobilenetv2: inverted residuals and linear bottlenecks. In: *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp 4510–4520. <https://doi.org/10.1109/CVPR.2018.00474>
72. Schyns PG, Oliva A (1994) From blobs to boundary edges: evidence for time- and spatial-scale-dependent scene recognition. *Psychol Sci* 5(4):195–200. <https://doi.org/10.1111/j.1467-9280.1994.tb00500.x>
73. Shaban A, Bansal S, Liu Z, Essa I, Boots B (2017) One-shot learning for semantic segmentation. *Proceedings of the British machine vision conference (BMVC)*, pp 167.1–167.13. <https://doi.org/10.5244/C.31.167>
74. Siddique N, Paheding S, Elkin CP, Devabhaktuni V (2021) U-net and its variants for medical image segmentation: a review of theory and applications. *IEEE Access* 9:82031–82057. <https://doi.org/10.1109/ACCESS.2021.3086020>
75. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: *3rd international conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*. [arXiv:1409.1556](https://arxiv.org/abs/1409.1556)
76. Sovrasov V: Fops-counter.pytorch. <https://github.com/sovrasov/flops-counter.pytorch>
77. stigma0617: Vovnet-deeplabv3 (deeplabv3 with densenet161 backbone). <https://github.com/stigma0617/VoVNet-DeepLabV3>

78. Sun K, Xiao B, Liu D, Wang J (2019) Deep high-resolution representation learning for human pose estimation. In: CVPR. <https://doi.org/10.1109/CVPR.2019.00584>
79. Sun K, Zhao Y, Jiang B, Cheng T, Xiao B, Liu D, Mu Y, Wang X, Liu W, Wang J, Hrnet-semantic-segmentation. <https://github.com/HRNet/HRNet-Semantic-Segmentation>
80. Sun K, Zhao Y, Jiang B, Cheng T, Xiao B, Liu D, Mu Y, Wang X, Liu W, Wang J (2019) High-resolution representations for labeling pixels and regions. CoRR [arXiv:1904.04514](https://arxiv.org/abs/1904.04514)
81. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), pp 1–9. <https://doi.org/10.1109/CVPR.2015.7298594>
82. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of IEEE conference on computer vision and pattern recognition. <https://doi.org/10.1109/CVPR.2016.308>
83. Takikawa T, Acuna D, Jampani V, Fidler S (2019) Gated-scnn: Gated shape cnns for semantic segmentation. ICCV. <https://doi.org/10.1109/ICCV.2019.00533>
84. Tan M, Le QV (2019) Efficientnet: rethinking model scaling for convolutional neural networks. In: Proceedings of the 36th international conference on machine learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol 97, pp 6105–6114. PMLR. <http://proceedings.mlr.press/v97/tan19a.html>
85. Tian P, Wu Z, Qi L, Wang L, Shi Y, Gao Y (2020) Differentiable meta-learning model for few-shot semantic segmentation. In: The Thirty-Fourth AAAI conference on artificial intelligence, AAAI, pp 12087–12094. AAAI Press. <https://aaai.org/ojs/index.php/AAAI/article/view/6887>
86. Tian Z, Lai X, Jiang L, Shu M, Zhao H, Jia J (2020) Generalized few-shot semantic segmentation. CoRR [arXiv:2010.05210](https://arxiv.org/abs/2010.05210)
87. Tian Z, Zhao H, Shu M, Yang Z, Li R, Jia J (2020) Prior guided feature enrichment network for few-shot segmentation. IEEE transactions on pattern analysis and machine intelligence, pp 1–1. <https://doi.org/10.1109/TPAMI.2020.3013717>
88. Waibel A, Hanazawa T, Hinton G, Shikano K, Lang K (1989) Phoneme recognition using time-delay neural networks. IEEE Trans Acoust Speech Signal Process 37(3):328–339. <https://doi.org/10.1109/29.21701>
89. Wang H, Zhang X, Hu Y, Yang Y, Cao X, Zhen X (2020) Few-shot semantic segmentation with democratic attention networks. In: Computer Vision - ECCV 2020—16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII. Lecture Notes in Computer Science, vol 12358, pp 730–746. Springer. https://doi.org/10.1007/978-3-030-58601-0_43
90. Wang K, Liew JH, Zou Y, Zhou D, Feng J (2019) Panet: few-shot image semantic segmentation with prototype alignment. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV) (2019). <https://doi.org/10.1109/ICCV.2019.00929>
91. Wang X, Girshick R, Gupta A, He K (2018) Non-local neural networks. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 7794–7803. <https://doi.org/10.1109/CVPR.2018.00813>
92. Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 5987–5995. <https://doi.org/10.1109/CVPR.2017.634>
93. Xu N, Price B, Cohen S, Yang J, Huang TS (2016) Deep interactive object selection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 373–381. <https://doi.org/10.1109/CVPR.2016.47>
94. Yakubovskiy P (2020) Segmentation models pytorch. <https://github.com/qubvel/segmentation-models.pytorch>
95. Yang B, Liu C, Li B, Jiao J, Ye Q (2020) Prototype mixture models for few-shot semantic segmentation. In: Computer Vision—ECCV 2020—16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII. Lecture Notes in Computer Science, vol. 12353, pp. 763–778. Springer. https://doi.org/10.1007/978-3-030-58598-3_45

96. Yang M, Yu K, Zhang C, Li Z, Yang K (2018) Denseaspp for semantic segmentation in street scenes. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 3684–3692. <https://doi.org/10.1109/CVPR.2018.00388>
97. Yang X, Wang B, Zhou X, Chen K, Yi S, Ouyang W, Zhou L (2020) Brinet: towards bridging the intra-class and inter-class gaps in one-shot segmentation. In: 31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7–10, 2020. BMVA Press. <https://www.bmvc2020-conference.com/assets/papers/0139.pdf>
98. Yuan Y, Chen X, Wang J (2020) Object-contextual representations for semantic segmentation. In: Vedaldi A, Bischof H, Brox T, Frahm JM (eds) Computer vision—ECCV 2020, pp 173–190. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-58539-6_11
99. Yuan Y, Wang J (2018) Ocnet: object context network for scene parsing. CoRR [arXiv:1809.00916](https://arxiv.org/abs/1809.00916)
100. Yuan Y, Xie J, Chen X, Wang J (2020) Segfix: model-agnostic boundary refinement for segmentation. In: European conference on computer vision, pp 489–506. Springer. https://doi.org/10.1007/978-3-030-58610-2_29
101. Zhang C, Lin G, Liu F, Guo J, Wu Q, Yao R (2019) Pyramid graph networks with connection attentions for region-based one-shot semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV) (October 2019). <https://doi.org/10.1109/ICCV.2019.00968>
102. Zhang C, Lin G, Liu F, Yao R, Shen C (2019) Canet: class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR) (June 2019). <https://doi.org/10.1109/CVPR.2019.00536>
103. Zhang F, Chen Y, Li Z, Hong Z, Liu J, Ma F, Han J, Ding E (2019) Acfnets: attentional class feature network for semantic segmentation. In: 2019 IEEE/CVF international conference on computer vision (ICCV), pp 6797–6806. <https://doi.org/10.1109/ICCV.2019.00690>
104. Zhang H, Dana K, Shi J, Zhang Z, Wang X, Tyagi A, Agrawal A (2018) Context encoding for semantic segmentation. In: The IEEE conference on computer vision and pattern recognition (CVPR) (June 2018). <https://doi.org/10.1109/CVPR.2018.00747>
105. Zhang H, Zhang H, Wang C, Xie J (2019) Co-occurrent features in semantic segmentation. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 548–557. <https://doi.org/10.1109/CVPR.2019.00064>
106. Zhang X, Wei Y, Yang Y, Huang TS (2020) Sg-one: similarity guidance network for one-shot semantic segmentation. *IEEE Trans Cybern* 50(9):3855–3865. <https://doi.org/10.1109/TCYB.2020.2992433>
107. Zhao H, Shi J, Qi X, Wang X, Jia J (2017) Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2881–2890. <https://doi.org/10.1109/CVPR.2017.660>
108. Zhou B, Zhao H, Puig X, Fidler S, Barriuso A, Torralba A (2017) Scene parsing through ade20k dataset. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 5122–5130. <https://doi.org/10.1109/CVPR.2017.544>

Worst-Case Adversarial Perturbation and Effect of Feature Normalization on Max-Margin Multi-label Classifiers



Ritesh Kumar Gupta  and Yashaswi Verma 

1 Introduction

Text-based search and retrieval have gained popularity with modern search engines. Naturally, similar technologies can be applied to image search tasks if an image can be described using text by either assigning a few labels, a short caption, or a paragraph. Since it is not feasible to do manual annotation, automated computational methods have become necessary to perform such tasks.

Associating images with texts started by addressing the task of classifying a single (prominent) object in an image. However, an image generally contains multiple objects, that too in some specific context and with some semantic meaning. This leads to the problem of multi-label image annotation, where an image is assigned multiple labels that describe its semantics. This can also be helpful in other related tasks such as object recognition, image retrieval, and image captioning.

Image annotation can be considered as a multi-label classification problem. In multi-label classification, for a given query sample, we predict a subset of labels from a fixed vocabulary. For example, given an image with Indian actors, identify who of all the Indian actors are present in this picture. Several attempts have been made to address the problem of automatic image annotation. Initially, researchers targeted it by translating an image into a few keywords or by mapping the relevance between features of images and their labels [7, 12–14, 20, 25]. Recently, advancements in

RKG contributed to this work while he was a student at IIT Jodhpur.

R. K. Gupta (✉)

Audio Technologies and Codecs (India) Pvt Ltd., Noida, India

e-mail: ritesh@atc-labs.com; gupta.30@iitj.ac.in

Y. Verma

IIT, Jodhpur, India

e-mail: yashaswi@iitj.ac.in

deep learning have helped in improving the performance on the image annotation task [4–6, 29].

On the other hand, it is well-acknowledged that there is a fundamental difference between the human perception of an image and that of a machine. For a machine, an image is simply a chunk of numbers, which has led to the design of algorithms that can produce numeric illusions for machines called adversarial examples or adversarial samples. In [22], it was shown that many machine learning (specifically deep learning) models make mistakes on adversarial samples, which are produced intentionally to fool such models. An adversarial example is produced by a slight modification in the original sample. A machine learning model's performance can be excellent on original samples but degrades on adversarial samples. Such attacks that are performed using adversarial examples are known as adversarial attacks, and have led to an increase in security concerns for machine learning models. For instance, a self-driving car can be forced to make wrong decisions and cause an accident if given an adversarial sample as input, or illegal content can be shared with minor modifications which can bypass a filter applied to prevent them.

There are primarily two types of adversarial attacks. The first one is a targeted attack, and the second one is an untargeted attack. In targeted attacks, an attack is targeted on a model to make it misclassify an adversarial example to a target class. Here, the target model and target class are known/fixed in advance. In untargeted attacks, there is no targeted class. The only purpose is to make some target model misclassify an adversarial example, which means the output of an adversarial example can be anything other than the actual output. It has been found that targeted attacks are more likely to succeed on an adversarial example than untargeted attacks, but they also take more time [19]. In both of these two broad categories, adversarial attacks are based on one of the three models: black-box models, gray-box models, and white-box models. These models are defined based on the knowledge that is available to the attacker. In black-box models, the attackers do not have knowledge about the model architecture to be attacked, however, they can predict outputs to specific inputs. These attacks are made possible based on the concept of transferability of adversarial examples, which means an adversarial example initially designed to attack some other model can attack the target model as well. In gray-box attacks, the attacker does have access to the model architecture but does not have knowledge of the parameters of the model. Here, the attacker can create a surrogate model of the same architecture as that of the target model to create adversarial samples. Additional knowledge of architecture helps gray-box attacks to be more powerful than black-box attacks. In white-box attacks, the attacker has full knowledge about the target model, which enables attackers to create adversarial examples directly using the target model.

In parallel, the research community has also been working on defense against such attacks. In [9], Goodfellow et al. proposed adversarial training to defend against such attacks. It is a brute force approach which generates many adversarial examples which are then used to train robust models. Mandry et al. [16] used adversarial training to show that models trained on MNIST dataset were robust to white-box adversarial attacks. Tramr et al. [23] proposed ensemble adversarial training which

transfers perturbations from other models to increase the robustness. In [15, 28], randomization techniques are used to defend against adversarial attacks.

In this paper, we first propose an algorithm to generate adversarial perturbations. These perturbations do not require prior knowledge of training data or the model. The presented algorithm takes original features as input and generates worst-case perturbation vector, under a given constraint on its norm. To generate adversarial samples, these worst-case perturbations are added to the original feature vector. While machine learning models have been shown to be robust against random perturbations in the past [15, 28], we show that systematically generated random perturbations can act as severe attackers. We also study, for the first time, a simple feature normalization technique using L_2 -norm as a tool to improve the robustness of algorithms against adversarial attacks. Experiments on two benchmark datasets (ESP Game [26] and IAPRTC-12 [11]) show that this simple feature normalization technique helps in improving the robustness of machine learning algorithms against worst-case adversarial attacks, thus establishing feature normalization as a recommended practice to make machine learning algorithms more robust against adversarial samples. It also opens up a new direction for further analyzing the impact of simple techniques like normalization in developing robust machine learning models. Note that both our attack and defense strategies are data- as well as model-independent, and instead are based on the algebraic and distributional properties of vectors, thus making them generalizable.

To summarize, the contributions of this paper are as follows:

1. We present a data- as well as model-independent algorithm to generate adversarial samples using worst-case perturbations.
2. We demonstrate feature normalization as a simple technique to increase the robustness of machine learning models against adversarial attacks.
3. We provide extensive experimental analyses under multiple setups using two state-of-the-art max-margin multi-label classifiers on two benchmark datasets to validate our ideas.

2 Our Approach

In this section, we first present our approach of generating adversarial samples, and then discuss the feature normalization technique to reduce the impact of such attacks.

2.1 *Generating Adversarial Samples*

We propose a data- and model-independent mechanism to generate adversarial samples by adding an adversarial perturbation vector to the original feature vector. These perturbation vectors correspond to the worst-case perturbations to produce adversar-

ial samples. We also show how to control the amount of perturbations to be generated for samples by constraining the norm of the perturbation vector.

For a given feature vector \mathbf{x} , we consider a perturbation $\tilde{\mathbf{v}}$. The worst-case perturbations correspond to the $\|\cdot\|_\infty$ norm, and it is given by $\|\tilde{\mathbf{v}}\|_\infty = \max_d |\tilde{v}_d|$ with a constraint on its magnitude. Here, it can be noted that when the input \mathbf{x} is changed by small perturbation along each dimension by keeping the $\|\tilde{\mathbf{v}}\|_\infty$ the same, the evaluation of a linear evaluation function changes significantly [2].

Algorithm 1 outlines the steps of this approach. The input to the algorithm is a (test) feature vector along with a constant *maxNorm* which controls the amount (magnitude) of perturbation to be generated. With an increase in the value of *maxNorm*, the perturbations become stronger. To generate the adversarial perturbations, we use uniformly distributed random numbers.

Algorithm 1 Generation of Worst-Case Adversarial Sample

- 1: **Input:** Original d -dimensional feature vector \mathbf{x} , value of *maxNorm*
 - 2: **Output:** Adversarial feature vector $\tilde{\mathbf{x}}$
 - 3:
 - 4: **Steps:**
 - 5: Sample a uniformly distributed random vector \mathbf{v} in $[-1, 1]^d$.
 - 6: $m_1 =$ Largest absolute value in \mathbf{v}
 - 7: $m_2 = \frac{\text{maxNorm}}{m_1}$
 - 8: $\tilde{\mathbf{v}} =$ Vector obtained after multiplying each element of \mathbf{v} by m_2 .
 - 9: Generate adversarial feature vector: $\tilde{\mathbf{x}} = \mathbf{x} + \tilde{\mathbf{v}}$
-

After generating an adversarial sample corresponding to a given feature vector, we examine the impact of these adversarial samples on the accuracy of two state-of-the-art linear max-margin multi-label prediction algorithms as discussed in Sect. 3.3.

2.2 Feature Normalization to Safeguard Against Adversarial Attack

Feature normalization is a well-practiced idea that is used as a de facto data pre-processing technique in machine learning. In this paper, we study feature normalization from a new perspective—as a defense mechanism to safeguard a model trained on clean (non-adversarial) data against adversarial attack. Specifically, we consider the well-known Euclidean norm (also called L_2 -norm) for feature normalization, which is given by

$$\|\mathbf{x}\|_2 = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \left(\sum_{i=1}^d x_i^2 \right)^{1/2} \quad (1)$$

To normalize a feature vector using the Euclidean norm, we perform element-wise division of each element in the feature vector by $\|x\|_2$, which gives a normalized feature vector of unit norm.

3 Experiments and Discussion

In this section, we will discuss the datasets, evaluation metrics and multi-label classification methods used in our experiments, and the results of our analyses.

3.1 Datasets

In our experimentation, we have used two benchmark multi-label image annotation datasets, namely ESP Game [26] and IAPRTC-12 [11] as described below.

ESP Game: This dataset was formed using an online game where two players are randomly given an image, and they need to assign some semantically relevant label(s) to that image. Each player does not know the labels being assigned by the other player until a common label is assigned by both of them, by which they score points.

IAPRTC-12: It is a collection of still natural scene images, including pictures of different sports, actions, animals, landscapes, and many other aspects of contemporary life. A detailed textual description is associated in up to three different languages (English, German, and Spanish) to each image in this dataset. Makadia et al. [17, 18] extracted nouns from the English language descriptions, which are treated as annotations to the images.

The ESP Game dataset consists of 20770 images and 268 labels, while the IAPRTC-12 dataset contains 19627 images and 291 labels. Out of 19627 images in IAPRTC-12, 170 images in the training set and 5 images in the test set are duplicated. After removing these images, we are left with 19452 images in IAPRTC-12.

For each image in both the datasets, we use the feature representations provided by the authors of [27], in which a 4096-dimensional feature vector is extracted for each image using the VGG-F model [21]. After this, feature dimensionality has been reduced using PCA to explain 80% of the information. This yields 597-dimensional feature vectors for images in ESP Game and 536-dimensional feature vectors for images in IAPRTC-12. Statistics of these datasets are given in Table 1.

3.2 Evaluation Metrics

We use semantic versions of per-label precision, recall, and F_1 score for evaluation following earlier papers [24, 27], and use the date provided by the authors of [27].

Table 1 Statistics of the two datasets used in our experiments

Dataset	#Samples	#Training samples	#Test samples	#Labels	Labels/sample
ESP Game	20770	18689	2081	268	4.7
IAPRTC-12	19452	17495	1957	291	5.7

While traditional precision, recall, and F1-score used in image annotation and multi-label learning focus on the evaluation of predicted labels, they tend to ignore their semantics. Semantic metrics make use of semantic hierarchies of labels from WordNet [8] by extracting one or more directed paths from parent label to child label. This can be done for each label and gives information about semantic dependencies among them. For example, a “woman” is a “person”, a “lady” is a “woman”, and a “boy” is a “person”. To compute semantic metrics, we construct semantic paths for the ground-truth label-set and the predicted set of labels for a given image. After this, we assign a score (weight) to each semantic path from the ground-truth semantic paths. For every semantic path in the ground-truth semantic paths, we consider all those labels which are common in that semantic path and the predicted set of labels. Among the weights of such labels, the maximum weight is considered as the weight of that semantic path. After computing the weight corresponding to each semantic path from the ground-truth semantic paths, we add all of them. Let \mathbf{s} represent the cumulative weight. Now, we calculate semantic precision as $P^s = \frac{\mathbf{s}}{|P|}$, semantic recall as $R^s = \frac{\mathbf{s}}{|G|}$, and semantic F_1 score as $F_1^s = \frac{2 \times P^s \times R^s}{P^s + R^s}$, where $|P|$ and $|G|$ represent the number of semantic paths constructed for the predicted and ground-truth label-sets, respectively.

3.3 Benchmark Max-Margin Multi-label Methods

Multi-label classification is a fundamental task in machine learning, which is a generalization of the (single-label) classification or multi-class classification task, and has a variety of applications. In the past, max-margin methods such as [1, 2, 10] have been shown to achieve state-of-the-art results on multi-label learning tasks, particularly when the number of samples and labels is large [3], and have thus been considered as benchmarks in studies focusing on diverse learning-based applications. Motivated by this, in this paper, we study the effect of the proposed adversarial attack and defense mechanisms on the image annotation task by empirically studying the performance of two state-of-the-art max-margin multi-label algorithms, namely ProXML [2] and DiSMEC [1], with various levels of adversarial attacks. Both these algorithms learn one classifier per label in a one-versus-rest manner, with the difference being in the regularization and loss terms used in their respective objective functions. For details on these algorithms, the reader may refer to the respective papers.

3.4 Results and Discussion

Now, we discuss the results obtained by experimenting on the above two datasets using the multi-label learning methods we discussed in Sect. 3.3, by evaluating the performance for the top 5, 7, and 9 predicted labels using semantic precision, semantic recall, and semantic F_1 -score. Specifically, we study the effect of adding the proposed perturbations to test features and evaluating the performance without (“After Attack”) and with normalization (“After Normalization”). The results of our experiments are shown in Tables 2, 3, and 4, which represent the results obtained for predicting 5, 7, and 9 labels respectively on both the datasets. Here, the “Baseline” results denote the results of the evaluation on normalized original test features without having any perturbation added. In our experiments, we have generated three sets of adversarial test features using three different values of $maxNorm$ in Algorithm 1. Our results are divided into three setups depending on the value of $maxNorm$, which we keep as 0.01, 0.1, and 1.

The first setup in our experiments uses the $maxNorm$ value of 0.01. The perturbations generated by $maxNorm$ value of 0.01 are very weak and consequently we see that the results after the attack are almost similar to those with the original test features. This is because in the case of $maxNorm$ value of 0.01, the feature distribution does not change much to significantly affect the results in comparison to the baseline results. For the same reason as above, we do not see much effect of feature normalization on the results in this case.

The second setup in our experiments uses a value of 0.1 for $maxNorm$. Here, we have increased the $maxNorm$ value to $10\times$ in comparison to the previous setup, and the results obtained on the adversarial features thus obtained demonstrate the effect of such an attack. We can observe that now the F_1^s score of ProXML on the ESP Game dataset has reduced by around 44% for 5, 7, and 9 labels. Similarly, the F_1^s score of DiSMEC has reduced by around 40% for 5 labels and around 42% for 7 and 9 labels. On the IAPRTC-12 dataset also, both the algorithms experience the effect of adversarial attack and demonstrate similar degradation in scores. In this case, the perturbations added to the test features change the feature distribution significantly, which results in a significant drop in the performance of both the algorithms. Next, unlike the previous case with $maxNorm = 0.01$, now the effect of normalizing the adversarial features becomes visible and we observe improvements in performance in almost all the cases. For example, on the IAPRTC-12 dataset, ProXML achieves an improvement of 14.36, 17.71, and 22.26% in F_1^s score for 5, 7, and 9 labels respectively compared to the results without normalization.

The third setup in our experiments uses a $maxNorm$ value of 1, which means now we are adding extreme perturbations to the test features. As we can observe, this leads to a substantial reduction in performance in all the cases. For example, the F_1^s score of ProXML on both the datasets drops down to the range of 1.68–2.87%, and that of DiSMEC now ranges from 3.95 to 6.13%. We can also observe that on increasing the magnitude of perturbations, the effect of normalization becomes more prevalent. In this case, normalizing the adversarial features leads to an improvement

Table 2 Results for predicting 5 labels on ESP game and IAPRTC-12 datasets with adversarial and normalized adversarial features

Dataset	maxNorm	Method	ProXML			DISMEC		
			P^s	R^s	F_1^s	P^s	R^s	F_1^s
ESP game	0.01	Baseline	41.61	33.55	34.69	41.22	33.42	34.49
		After attack	41.07	33.24	34.33	40.96	33.25	34.31
		After normalization	41.04	33.15	34.28	40.89	33.15	34.24
		% improvement	-0.07	-0.27	-0.14	-0.17	-0.30	-0.20
	0.1	Baseline	41.61	33.55	34.69	41.22	33.42	34.49
		After attack	22.67	19.59	19.61	23.96	20.04	20.37
		After normalization	26.11	19.97	21.14	24.94	19.24	20.29
		% improvement	15.17	1.94	7.80	4.09	-3.99	-0.39
	1	Baseline	41.61	33.55	34.69	41.22	33.42	34.49
		After attack	2.24	2.31	2.1	4.4	4.08	3.95
After normalization		11.96	8.49	9.28	11.5	8.26	8.98	
% improvement		433.93	267.53	333.64	161.36	102.45	127.34	

(continued)

Table 2 (continued)

Dataset	maxNorm	Method	ProXML			DISMEC		
			P^s	R^s	F_1^s	P^s	R^s	F_1^s
IAPRTC-12	0.01	Baseline	42.90	29.02	32.17	43.74	29.20	32.50
		After attack	42.85	29.04	32.18	43.6	29.13	32.44
		After normalization	40.92	29.05	32.24	43.8	29.18	32.47
		% improvement	-4.50	0.03	0.18	0.46	0.17	0.09
0.1	0.1	Baseline	42.90	29.02	32.17	43.74	29.20	32.50
		After attack	26.14	18.39	20.12	28.79	19.28	21.50
		After normalization	29.16	19.21	23.01	30.22	19.23	24.81
		% improvement	11.55	4.46	14.36	4.97	-0.25	15.4
1	1	Baseline	42.90	29.02	32.17	43.74	29.20	32.50
		After attack	1.98	1.61	1.68	6.75	4.82	5.25
		After normalization	10.73	6.43	8.87	12.71	9.21	11.21
		% improvement	441.92	299.38	427.98	88.30	91.08	113.52

Table 3 Results for predicting 7 labels on ESP game and IAPRTC-12 datasets with adversarial and normalized adversarial features

Dataset	maxNorm	Method	ProXML			DiSMEC		
			P^s	R^s	F_1^s	P^s	R^s	F_1^s
ESP game	0.01	Baseline	38.46	42.57	37.94	37.97	42.47	37.67
		After attack	38.16	42.22	37.63	37.56	42.05	37.25
		After normalization	38.2	42.18	37.65	37.58	42.07	37.27
		% improvement	0.11	-0.09	0.053	0.053	0.04	0.05
	0.1	Baseline	38.46	42.57	37.94	37.97	42.47	37.67
		After attack	20.38	24.66	20.94	21.29	25.03	21.61
		After normalization	25.14	26.89	24.42	23.8	25.82	23.27
		% improvement	23.35	9.04	16.61	11.79	3.16	7.68
	1	Baseline	38.46	42.57	37.94	37.97	42.47	37.67
		After attack	2.27	3.36	2.55	4.13	5.44	4.41
		After normalization	11.63	11.91	11.07	11.25	11.67	10.77
		% improvement	412.33	254.46	334.12	172.40	114.52	144.22

(continued)

Table 3 (continued)

Dataset	maxNorm	Method	ProXML			DISMEC		
			P^s	R^s	F_1^s	P^s	R^s	F_1^s
IAPRTC-12	0.01	Baseline	42.35	37.99	37.52	42.99	38.23	37.83
		After attack	42.15	37.94	37.41	42.64	38.03	37.58
		After normalization	42.27	37.96	37.46	42.69	38.08	37.61
		% improvement	0.28	0.05	0.13	0.12	0.13	0.08
0.1	0.1	Baseline	42.35	37.99	37.52	42.99	38.23	37.83
		After attack	24.92	23.87	22.87	26.91	24.7	24.13
		After normalization	30.18	26.17	26.92	29.63	25.18	26.21
		% improvement	21.10	9.64	17.71	10.11	1.94	8.62
1	1	Baseline	42.35	37.99	37.52	42.99	38.23	37.83
		After attack	2.1	2.46	2.14	6.14	6.19	5.8
		After normalization	10.61	9.24	10.16	12.44	12.48	12.89
		% improvement	405.24	275.61	374.77	102.61	101.62	122.24

Table 4 Results for predicting 9 labels on ESP Game and IAPRTC-12 datasets with adversarial and normalized adversarial features

Dataset	maxNorm	Method	ProXML				DISMEC			
			P^s	R^s	F_1^s	P^s	R^s	F_1^s		
ESP game	0.01	Baseline	34.72	49.33	38.44	34.15	49.25	37.98		
		After attack	34.45	48.98	38.14	33.84	48.73	37.64		
		After normalization	34.49	48.99	38.17	33.9	48.78	37.69		
		% improvement	0.12	0.02	0.08	0.18	0.10	0.13		
	0.1	Baseline	34.72	49.33	38.44	34.15	49.25	37.98		
		After attack	18.53	28.85	21.3	19.14	29.14	21.82		
		After normalization	22.99	32.09	25.31	21.85	30.98	24.2		
		% improvement	24.07	11.23	18.82	14.16	6.31	10.91		
	1	Baseline	34.72	49.33	38.44	34.15	49.25	37.98		
		After attack	2.3	4.43	2.87	3.95	6.76	4.71		
		After normalization	10.95	15.01	11.95	10.48	14.51	11.49		
		% improvement	376.09	238.83	316.38	165.32	114.65	143.94		

(continued)

Table 4 (continued)

Dataset	maxNorm	Method	ProXML			DISMEC		
			P^s	R^s	F_1^s	P^s	R^s	F_1^s
IAPRTC-12	0.01	Baseline	40.47	45.50	40.31	41.03	45.91	40.74
		After attack	40.24	45.32	40.12	40.69	45.51	40.38
		After normalization	40.36	45.37	40.19	40.73	45.59	40.47
		% improvement	0.30	0.11	0.17	0.09	0.18	0.23
		Baseline	40.47	45.50	40.31	41.03	45.91	40.74
		After attack	23.46	28.47	24.21	25.07	29.33	25.42
1	0.1	After normalization	27.12	31.94	29.6	28.14	31.24	28.94
		% improvement	15.60	12.19	22.26	12.25	6.51	13.85
		Baseline	40.47	45.50	40.31	41.03	45.91	40.74
		After attack	2.24	3.36	2.55	5.74	7.48	6.13
		After normalization	9.89	14.21	10.74	11.92	16.08	13.14
		% improvement	341.52	322.92	321.18	107.67	114.98	114.36

in all the metrics for both the methods on both datasets. For example, the F_1^s score of ProXML increases by 333.64% for 5 labels, 334.12% for 7 labels, and 316.38% for 9 labels on ESP Game, and on IAPRTC-12, ProXML achieves 427.98, 374.77, and 321.18% improvements in F_1^s scores for 5, 7, and 9 labels, respectively.

In general, our algorithm for generating perturbations allows us to regulate the magnitude of the worst-case perturbations, and increasing the value of $maxNorm$ increases the intensity of the adversarial attack. Next, as we increase the intensity of the attack by increasing the value of $maxNorm$, we see a clear advantage of normalization against the impact of such attacks. These results suggest that although normalization does not neutralize the effect of adversarial attacks completely, it does help in reducing the severity of such attacks.

4 Summary and Conclusion

In this paper, we have proposed a data- as well as model-independent adversarial perturbation generation mechanism to generate the worst-case perturbation vectors under the constraints on their norm. Adversarial samples are then generated by adding these perturbation vectors to original feature vectors without the need for the availability of training data and model, making the proposed attack mechanism highly generalizable. We have also studied the effect of feature normalization from a novel perspective of defense against adversarial attacks on max-margin multi-label classifiers. Empirical studies on two benchmark datasets showed that feature normalization helps in reducing the adverse effect of adversarial attacks up to some extent. It does not neutralize the adversarial attacks completely, however it does reduce the severity of the impact of such attacks on empirical results. Further, the promise of feature normalization in adversarial defense becomes more evident as the strength/magnitude of perturbations increase. These results demonstrate that feature normalization should be adopted as a standard practice to reduce the impact of adversarial attacks and increase the robustness of machine learning models against adversarial attacks in general.

Acknowledgements YV would like to thank the Department of Science and Technology (India) for the INSPIRE Faculty award 2017.

References

1. Babbar R, Schölkopf B (2017) Dismec: distributed sparse machines for extreme multi-label classification. In: Proceedings of the tenth ACM international conference on web search and data mining. WSDM '17, Association for Computing Machinery, New York, NY, USA, pp. 721–729 (2017). <https://doi.org/10.1145/3018661.3018741>
2. Babbar R, Schölkopf B (2019) Data scarcity, robustness and extreme multi-label classification. Mach Learn 108(8):1329–1351 (2019). <https://doi.org/10.1007/s10994-019-05791-5>

3. Bhatia K, Dahiya K, Jain H, Kar P, Mittal A, Prabhu Y, Varma M (2016) The extreme classification repository: multi-label datasets and code. <http://manikvarma.org/downloads/XC/XMLRepository.html>
4. Chen SF, Chen YC, Yeh CK, Wang YC (2018) Order-free RNN with visual attention for multi-label classification (2018). <https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16114/16253>
5. Chen ZM, Wei XS, Wang P, Guo Y (2019) Multi-label image recognition with graph convolutional networks. In: CVPR, pp 5177–5186
6. Dutta A, Verma Y, Jawahar CV (2020) Recurrent image annotation with explicit inter-label dependencies. In: ECCV, pp 191–207
7. Duygulu P, Barnard K, de Freitas JFG, Forsyth DA (2002) Object recognition as machine translation: learning a lexicon for a fixed image vocabulary. In: Heyden A, Sparr G, Nielsen M, Johansen P (eds) Computer vision—ECCV 2002. Springer, Berlin, Heidelberg, pp 97–112
8. Fellbaum C (1998) WordNet: an electronic lexical database (1998). <https://wordnet.princeton.edu/>
9. Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. CoRR [arXiv:1412.6572](https://arxiv.org/abs/1412.6572)
10. Hariharan B, Zelnik-Manor L, Vishwanathan SVN, Varma M (2010) Large scale max-margin multi-label classification with priors. In: Proceedings of the international conference on machine learning, June 2010
11. ImageClef: ImageClef—IAPR TC-12 benchmark. <https://www.imageclef.org/photodata>
12. Jeon J, Lavrenko V, Manmatha R (2003) Automatic image annotation and retrieval using cross-media relevance models. In: Proceedings of the 26th annual international acm sigir conference on research and development in informaion retrieval. SIGIR '03, Association for Computing Machinery, New York, NY, USA, pp 119–126. <https://doi.org/10.1145/860435.860459>
13. Lavrenko V, Feng SL, Manmatha R (2004) Multiple bernoulli relevance models for image and video annotation. In: Proceedings of the 2004 IEEE computer society conference on computer vision and pattern recognition, vol 2, July 2004. IEEE Computer Society, Los Alamitos, CA, USA, pp 1002–1009. <https://doi.org/10.1109/CVPR.2004.171>, <https://doi.ieeeecomputersociety.org/10.1109/CVPR.2004.171>
14. Lavrenko V, Manmatha R, Jeon J (2004) A model for learning the semantics of pictures. In: Thrun S, Saul LK, Schölkopf B (eds) Advances in neural information processing systems, vol 16. MIT Press, pp 553–560. <http://papers.nips.cc/paper/2474-a-model-for-learning-the-semantics-of-pictures.pdf>
15. Liu X, Cheng M, Zhang H, Hsieh CJ (2018) Towards robust neural networks via random self-ensemble. ArXiv [arXiv:1712.00673](https://arxiv.org/abs/1712.00673)
16. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Towards deep learning models resistant to adversarial attacks. ArXiv [arXiv:1706.06083](https://arxiv.org/abs/1706.06083)
17. Makadia A, Pavlovic V, Kumar S (2008) A new baseline for image annotation. In: Forsyth D, Torr P, Zisserman A (eds) Computer vision—ECCV 2008. Springer, Berlin, Heidelberg, pp 316–329
18. Makadia A, Pavlovic V, Kumar S (2010) Baselines for image annotation. Int J Comput Vis 90(1):88–105 (2010). <https://doi.org/10.1007/s11263-010-0338-6>
19. Mopuri KR, Ojha U, Garg U, Babu RV (2018) NAG: network for adversary generation. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 742–751
20. Mori Y, Takahashi H, Oka R (1999) Image-to-word transformation based on dividing and vector quantizing images with words. In: MISRM'99 first international workshop on multimedia intelligent storage and retrieval management. citeseer.ist.psu.edu/368129.html
21. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR
22. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow IJ, Fergus R (2014) Intriguing properties of neural networks. CoRR [arXiv:1312.6199](https://arxiv.org/abs/1312.6199)
23. Tramèr F, Kurakin A, Papernot N, Boneh D, McDaniel P (2018) Ensemble adversarial training: Attacks and defenses. ArXiv [arXiv:1705.07204](https://arxiv.org/abs/1705.07204)

24. Verma Y (2019) Diverse image annotation with missing labels. *Pattern Recognit.* 93:470–484
25. Verma Y, Jawahar CV (2017) Image annotation by propagating labels from semantic neighbourhoods. *Int. J. Comput. Vis.* 121(1):126–148
26. von Ahn L, Dabbish L (2004) Labeling images with a computer game. In: Proceedings of the SIGCHI conference on human factors in computing systems. CHI '04, Association for Computing Machinery, New York, NY, USA, pp. 319–326 (2004). <https://doi.org/10.1145/985692.985733>
27. Wu B, Jia F, Liu W, Ghanem B (2017) Diverse image annotation. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 6194–6202
28. Xie C, Wang J, Zhang Z, Ren Z, Yuille A (2018) Mitigating adversarial effects through randomization. ArXiv [arXiv:1711.01991](https://arxiv.org/abs/1711.01991)
29. Zhu F, Li H, Ouyang W, Yu N, Wang X (2017) Learning spatial regularization with image-level supervisions for multi-label image classification. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 2027–2036

Catch Me if You Can: A Novel Task for Detection of Covert Geo-Locations (CGL)



Binoy Saha  and Sukhendu Das 

1 Introduction

The primary focus of the computer vision community has been on the understanding of visual scenes. To this end, many datasets and tasks have been proposed over the years to build AI systems that can perform specific scene understanding tasks like image classification [4, 10, 23, 25, 28, 30, 35], object detection [6, 7, 19, 22, 31], image captioning [3, 16, 37, 45], visual question answering [1, 8, 16, 37, 38], scene graph generation [18, 32, 33, 36], and many more. With the advent of deep convolutional networks [14, 15, 28, 30], one of the tasks where AI has evolved remarkably is object detection. Most successful object detection approaches are either based on two-stage region proposal methods [9, 26] or single-shot detection methods [20, 39]. These methods have become exceptionally good at recognizing and detecting objects. Thus, object detection can be used to instill knowledge about the distinguishing features of objects into machines.

Although, most of the visual scene understanding tasks in the field of computer vision involve the identification of the objects present in the scene, non-object image regions like hideouts, corners, bends, turns, and other obscured regions of the scene also contain crucial information for a specific set of surveillance tasks. Thus, the next step for advancement toward the goal of scene understanding would be to detect such covert locations in a scene.

Covert places for hiding behind any occluding objects (pillars, doors, and furniture), are concealed locations that are not usually detectable from the viewpoint (camera). However, an intelligent agent can analyze the scene to foresee such poten-

B. Saha (✉) · S. Das

Visualization and Perception Lab, Department of CSE, IIT, Madras, India
e-mail: binoy_saha@cse.iitm.ac.in

S. Das

e-mail: sdas@iitm.ac.in

tial regions around the obscuring (occluding) objects, from where the threats loom large. These specific parts (image regions) around the boundaries of occluding objects are the targets for detection by the algorithm. These locations may either be: (i) potential zones of threat caused by an adversary hiding behind the object, or (ii) target zones for further investigation to detect any other unidentified concealed object. To advance toward the goal of identification of such regions, we propose a novel task termed Covert Geo-Location (CGL) Detection. CGL detection finds applications in military counter-insurgency operations and intelligent scene surveillance (for identification of target zones) with path planning for a robot. In this work, we provide an intelligent visual aid for the identification of such locations.

We define the problem addressed in this paper as: Given an input RGB image, the goal is to identify all covert places (hideouts) in that image. We pose the CGL detection problem as that involving identification of regions encapsulating specific target boundary sub-segments of an obscuring item/object in a scene, which either poses threat or can act as target zones for further inspection of the scene. Since it is not possible to classify any region of an image as a hideout (CGL) without looking at its surroundings, CGL detection requires context-aware detection, and identification of CGLs in an image would require depth information of regions around the boundaries of obscuring items (like pillars, doors, furniture, wall endings, sofa, cot, etc.) and knowledge about the spatial position (in 3D) of the occluding object with respect to its background and neighboring objects in the scene.

We also highlight the importance of extracting depth information for CGL detection, which the traditional (object) detection approaches often do not consider. Our proposed method successfully extracts relevant depth features from only a single RGB image as input and quantitatively yields significant improvement over existing object detection and segmentation models (when adapted and trained for CGL detection).

In methods used for object detection, the classification of any region of interest (RoI) is solely dependent on the features of the RoI itself. Identifying and localizing covert locations will require additional knowledge about the RoI. For instance, to classify any object as an obstruction, the image region containing just the occluding object is not enough; knowledge about its position on the floor and the objects around it are equally important. As identification of covert locations requires knowledge about neighboring and background objects as well as their relative positions, models should learn to perform context-aware detection capturing the 3D spatial relationships between objects (their boundaries) and their surroundings.

For that purpose, we introduce a novel task termed Covert Geo-Location (CGL) Detection. Given an input RGB image, the goal is to identify and localize parts of boundaries of occluding objects, behind which lie potential hideouts (Covert Geo-Locations) in the scene. The output of the algorithms must hence be bounding box templates overlaying specific sections of boundaries of the occluding objects (as shown in Fig. 1a). This task is challenging because the model needs to infer the spatial relationship of an occluding object with the floor (for support, space requirements, and reachability), characteristics of boundaries on the obscuring items, etc. Figure 1b, c show examples of both cases: CGL (green boxes) as well as locations

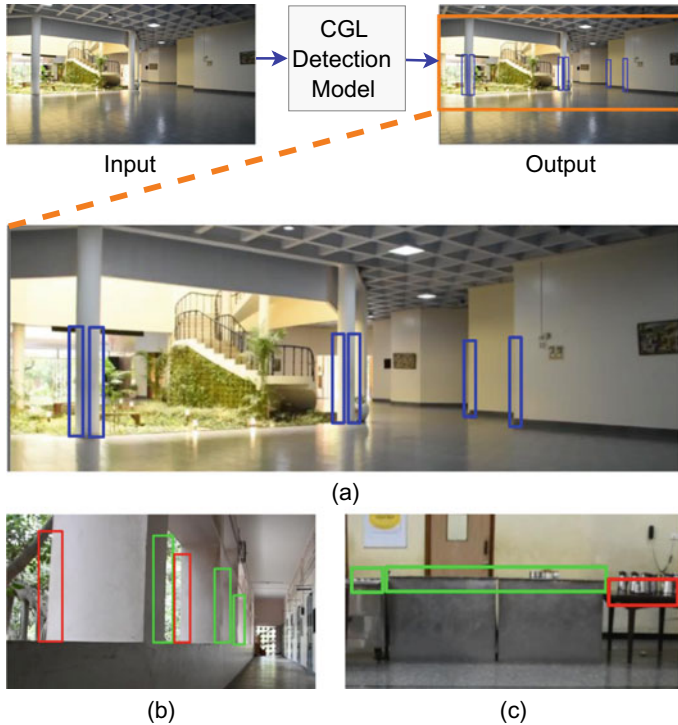


Fig. 1 a shows the input and the expected output for CGL detection. Blue bounding boxes indicate hand-annotated CGL Ground Truth. In b, c, green bounding boxes indicate valid CGL locations; while red ones indicate locations that appear similar to a CGL, but do not fall in that category (hideout or covert locations). In case of (b), no floor or platform exists at the base of the vertical zones (behind the pillar edges), indicated by red bounding boxes, for anything to be placed and obscured. In case of (c), the obscuring item (tabletop) cannot hide anything behind it

that appear similar to a CGL but actually are not (red boxes) so. In case of (b), no floor or platform exists at the base of the vertical zones (behind the pillar edges), indicated by red bounding boxes, for anything to be placed and obscured. In case of (c), the obscuring item (tabletop) cannot hide anything behind it which rests on the floor.

We also present a novel dataset for CGL detection containing real-world images depicting diverse environments captured in normal lighting conditions. Our dataset consists of images captured in indoor environments and building premises. On average, images contain around 8 to 9 CGLs, with the total number of hand-annotated CGLs being ≈ 15 K. We believe that this first-of-a-kind dataset along with its analysis presented in this paper will offer a new domain of work along the line of the aforementioned application.

Humans identify hideouts by searching for areas in the scene where there is enough distance between the foreground and its background. Since a depth map captures the

relative distance information, it can act as a good cue for CGL detection. As depth is independent of illumination, it provides a great complementary modality. However, effectively fusing information from RGB and depth modality is non-trivial, and it often requires additional computational power. Also, most of the existing datasets [5, 8, 13, 21] for vision-related tasks, including our CGL detection dataset, do not contain corresponding depth maps for input images. This leads us to the question, can we train models to jointly generate RGB and depth features, taking only RGB images as input? We address this question in this paper by proposing a novel method for CGL detection.

We observe that object detection models fail to capture the spatial contextual information essential for CGL detection. To perform context-aware detection, we thus map CGL detection to a binary segmentation task and experiment with segmentation models. We propose a novel segmentation-based method that implicitly attempts to train the feature extractor to generate relevant depth features. Our method uses an additional decoder called Depth-aware Feature Learning Block (DFLB), which is trained to classify image regions that have depth patterns similar to CGLs, as one class (say “potential CGL”), and the rest of the image as another class. DFLB serves two purposes, it aids the CGL segmentation block (supervised using hand-annotated ground truth) in attending to image regions having the necessary depth patterns appearing over CGL templates, and helps the common encoder to extract relevant depth features from the input RGB image. Subsequently, we propose two feature-level loss functions for the self-supervision of the common encoder, which further enhances the performance of our model.

Our key contributions are summarized as follows: (1) We propose a novel CGL detection task, which requires context-aware location detection. (2) We present a CGL detection dataset depicting diverse and challenging environments. Images from our dataset contain naturally occurring glares and shadows. (3) We propose a method for jointly learning RGB and depth features taking only a single RGB image as input. We introduce the Depth-aware Feature Learning Block (DFLB), which steers the feature extractor toward the extraction of relevant depth features. (4) We propose two feature-level loss functions, namely Geometric Transformation Equivariance Loss (GTE loss) and Intra-class Variance Reduction loss (IVR loss) for self-supervision of the encoder.

2 Related Work

The ultimate goal of artificial intelligence is to mimic human intelligence. Throughout the history of artificial intelligence, the research community has tried to incrementally move toward this goal by designing tasks and curating datasets for incorporating various aspects of human intelligence into machines. In this work, we propose yet another task that can be used to instill a new type of knowledge into machines. In this section, we briefly describe some of the tasks that have been explored so far for visual scene understanding and contrast each one of them with our proposed task.

Object Detection In object detection [9, 20, 26, 39], the task is to identify and localize all objects present in the scene. For the classification of RoIs (regions that potentially contain an object) in object detection, features of the ROIs alone are sufficient. Global (image-level) context also influences the detection in certain cases but the local spatial context (knowledge about the surroundings of the RoI) is never exploited in object detection. On the other hand, local spatial context with respect to other neighboring regions of the scene is of paramount importance for CGL detection.

Semantic Segmentation The goal in semantic segmentation [29, 34, 46] is to generate a pixel-level classification map having a size of $H \times W$, where each pixel is assigned to one of the K categories. Although CGL detection can be mapped to a binary semantic segmentation task, it is far more challenging to segment CGLs than the categories available in existing segmentation datasets [21, 43, 44].

Scene Recognition: In this task, given an input image, the model needs to classify the entire image into one of the “ K ” scene categories. As pointed out in [40–42], for an intelligent system to understand the environment thoroughly, it should be able to identify the place, and to do that, the system needs to understand what set of objects co-occur at what places and how the arrangement of objects depends upon the place. As CGL detection encourages context-aware location detection, CGL detection models can aid in identifying complex regions like corners, hideouts, bends, turns, and other obscuring regions of the scene, leading to better scene recognition.

Other visual scene understanding tasks: Tasks like visual question answering VQA [1, 8, 38] (which involves answering natural language questions about an input image), scene graph generation SGG [13, 17, 32] (which involves detection of objects in the input image and generation of a graph depicting the relationships between the detected objects), and image captioning [3, 16, 21] (which involves producing natural language captions for an input image) also require complex understanding of the relationships between the objects present in the scene, but RoI classification in all the abovementioned tasks is still context agnostic.

None of the datasets used for existing vision-related tasks has manually annotated regions in images identifying locations of hideout (regions for concealment) to be exploited for the problem addressed in this paper. CGLs can also be considered to be non-object locations (but in the vicinity of objects) around the edges of certain objects, having a distinct depth pattern. In surveillance scenarios, CGL detection can aid in the exploration of zones in an image, which are potential candidates for hazards and threats.

3 Dataset Details

3.1 Images

Existing computer vision datasets do not contain CGL-centric real-world images, so we have curated a novel collection of images for CGL detection. Images of our



Fig. 2 Some of the images from our proposed dataset are overlaid with ground truth CGL bounding boxes. CGL bounding boxes are indicated in blue. Few more image samples are included in the supplementary material. https://drive.google.com/file/d/1_hu31vshulC8P2bTL9xouZQRJ0Btqe0u/view?usp=sharing Link to the gallery and dataset

dataset were captured using a Nikon D7200 DSLR camera. We intentionally avoided any humans in the view, to make the images of our dataset realistic from the point of view of the application (counter-insurgency operations) and to avoid violation of ethical norms. Our dataset consists of 1400 real-world images depicting diverse environments. To build robust models that can handle varied lighting conditions, we have captured some of the images in broad daylight and some of the images during nighttime in normal lighting conditions. As will be the case with images encountered by the model in the real world, images in our dataset contain naturally occurring shadows and glares. Images were captured in corridors, residential houses, offices, labs, classrooms, large dining halls, seminar halls, etc. The majority of our images (sample images are shown in Fig. 2) depict indoor scenes and a few images depict building premises. Our images are of dimension 1080×1920 (H \times W).

We experimented with two train/test splits: split 1 and split 2, each containing non-overlapping sets of image samples in train vs test partitions. In split 1, the images in the test set (offices, lobbies, etc) are chosen from scene conditions and locations different from those in the train set (classrooms, houses, etc). This ensures that the locations are unknown for the model at test time. In split 2, a few images ($\approx 17\%$) in the test set are also from those conditions/locations appearing in the train set too. Split 1 is more challenging as compared to split 2 because the model trained on the training set of split 1 would not have seen test set locations, and thus the model would have to be robust enough not to overfit to any inconspicuous biases present in the dataset (train set), to generalize well on new locations. Both the training sets contain around 80% of the images and the test sets contain the rest of the images ($\approx 20\%$).

3.2 Annotations

Annotation involved drawing bounding boxes around CGLs as in the case of object detection [5, 21]. For annotating our dataset, we used an online interactive annotation tool named Supervisely. Annotations were performed in two phases. In the first phase, bounding boxes were drawn around CGLs and in the second phase, every image was verified by at least three other annotators followed by a master annotator, and minor corrections/adjustments were made to the bounding boxes wherever required.

To give a physical analogy from the real world: imagine the scenario of a hide-and-seek game played indoors by children. As a child (player) enters a room and looks for potential places for hideout, he/she must identify zones around all such objects, each of which has the potential to obscure another object or a friend. All edges of an object are not important to explore. The child is knowledgeable enough with the experience to seek only noteworthy places to explore, based on scene analysis by the human brain. This scenario was described and explained to annotators at the beginning of their work, with examples. So the job of the annotators mainly involved the identification of specific sub-sections of boundaries of obscuring objects, which have either zones for concealment and are adjacent to zones needing further visual exploration for unearthing any occluded object/human.

As discussed earlier, CGL locations in an image are specific boundaries of occluding objects which have the: (i) potential for imminent threats, or (ii) potential to provide scope for further exploration of the scene. Hence, CGLs in an image usually lie around the boundary/edges of items like furniture, pillars, doors, windows, staircase, packing boxes, etc. Image regions around specific edges of these items are treated as CGLs if they have the potential (3D space) to occlude and accommodate an object or person. Anyone (or any item) hiding behind any obscuring item may appear in the view from these CGL regions, making these regions locations of threat as well as of interest for further exploration of the environment.

Although the targets for CGL are specific boundaries of occluding objects, it is hard for both the annotators to delineate them, as well as for detection algorithms to identify such sub-segments of edges and lines. Moreover, threats appearing from behind the occluding objects typically appear from around these boundaries. Hence, we decided to identify CGL as a rectangular template around the boundaries of occluding objects, with partial coverage of both the body of the object and its background layer. This made the task of both the annotator and the detection model easier.

CGL bounding boxes have been annotated such that approximately 40% of the area inside the bounding box contains the obscuring item and the rest (60%) consists of background or neighboring objects. This 60–40 ratio is only notional and was given as advice to annotators. CGLs in our dataset are of two types: horizontal (wherein the height of the bounding box enclosing the CGL is less as compared to its width) and vertical (wherein the width of the bounding box is less as compared to its height).

For the height of vertical CGLs, the average human's height (as qualitatively perceived in the scene by an annotator) was used as a measure of maximum range to

annotate. The width of a vertical CGL and the height of a horizontal CGL depends on the size of obscuring object, but we have tried to follow the convention of keeping these quantities close to 100 pixels. Our dataset contains 15K CGLs, out of which 59% are vertical CGLs and 41% are horizontal CGLs. Annotations are stored in MS COCO JSON format [30]. The size of the anchor box is a crucial hyperparameter for any anchor-based detection model. So we have included distributions for the height and width of horizontal and vertical CGL bounding boxes in the supplementary material. More statistics on the number of horizontal and vertical bounding boxes (per image) are included in the supplementary material.

CGL detection model should be able to detect CGLs irrespective of the obscuring item. So, to support the training of deep models that can learn the underlying characteristics of CGL without overfitting to the obscuring items present in our CGL detection dataset, we have tried to incorporate a diverse set of obscuring items such as door, staircase, pillar, sofa, window, chairs, etc. Although furniture objects, pillars, and obstructions constitute the majority of the obscuring items in our dataset, their appearances do vary significantly across environments.

4 Adaptation of Existing Models for CGL Detection

As CGL detection involves the identification and localization of RoIs, we can either adapt models designed for object detection or we can use binary semantic segmentation by considering pixels inside CGL bounding boxes to be of one class (CGL) and the rest of the image to be of another class (non-CGL/background). In this section, we describe the object detection and segmentation models we have adapted and trained (from scratch) for CGL detection as part of our initial exploration.

4.1 YOLOv3

As YOLOv3 [6] is one of the standard models for most detection tasks, we adapt and train it for CGL detection. However, since CGL detection is much more challenging when compared with object detection, we observe that YOLOv3 (proposed for object detection) fails to capture the complex relationship between CGLs and the neighboring obscuring items.

To classify any region of an input image as a CGL, spatial context is as important as the features of the concerned region. However, existing object detection models do not take into account the spatial context for the classification of RoIs. On the other hand, segmentation models can take into account the spatial context. Thus, we map CGL detection to a segmentation task and explore several existing segmentation models for CGL detection.

4.2 MobileNetv2

CGL detection could find direct application in the field of robotics. Hence, we consider MobileNetv2 [11, 27] for CGL segmentation. MobileNetv2 provides an excellent option if the model is to be deployed on a low-end device, as it uses depthwise separable convolutions to reduce the number of model parameters. It uses an inverted residual block where skip connections are used to connect linear bottleneck layers.

4.3 HRNetv2

Most of the CNN-based feature extractors gradually reduce the resolution of the feature map, but HRNetv2 [29] maintains high-resolution representation throughout the process. It employs parallel high-to-low resolution subnetworks, which exchange information at each layer to produce good high-resolution representation. HRNetv2 has been shown to perform very well on segmentation tasks. Hence, we incorporate it for CGL segmentation.

5 Proposed Method

We propose a segmentation-based model for CGL detection (shown in Fig. 3). Most RGB segmentation models contain a single encoder and a single decoder. For our method, we propose an additional auxiliary decoder called Depth-aware Feature Learning Block (DFLB), which facilitates the joint learning of RGB and depth features by the common RGB encoder and also guides the CGL Segmentation Block (CGLSB) in attending to regions having depth pattern suitable for CGL detection.

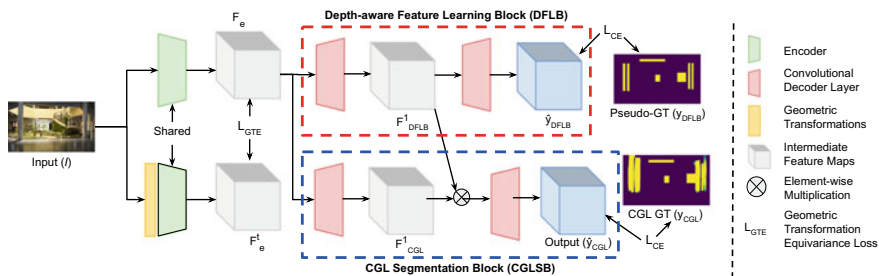


Fig. 3 The proposed architecture for CGL detection using binary semantic segmentation. The dashed red box (top) encloses the proposed Depth-aware Feature Learning Block (DFLB), which aids in learning depth-aware features. The dashed cyan box (bottom) encloses the CGL segmentation block (CGLSB)

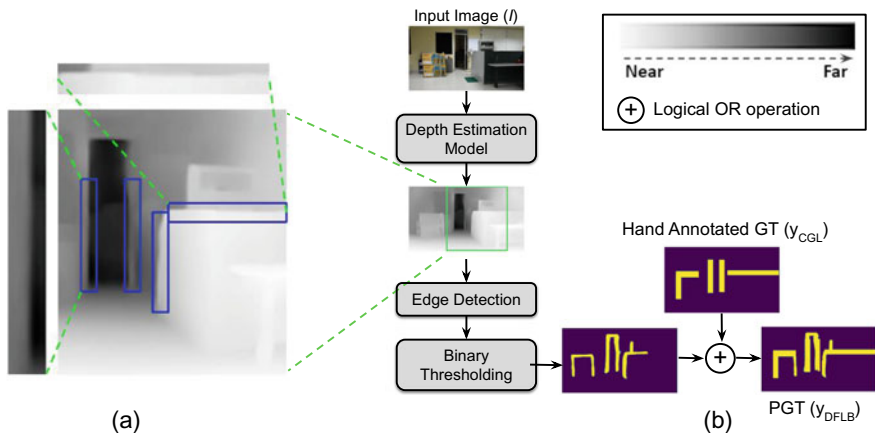


Fig. 4 **a** Shows a part of depth map for the input image shown in **(b)**. GT CGL bounding boxes have been overlaid on the depth map and some of the bounding boxes have been enlarged for better visualization of the depth pattern in a CGL. This depth pattern will henceforth be referred to as the “CGL depth pattern”. **b** shows the flowchart depicting the process of generation of Pseudo-GT (PGT). The green box over the depth map indicates the area that has been enlarged and shown in **(a)**, to clearly exhibit some samples of “CGL depth pattern”. Yellow blobs in PGT represent regions having “CGL depth pattern”

An illustration of CGL depth pattern is shown in Fig. 4a. Although our model resembles multi-task learning networks [2, 24], our main novelty lies in learning relevant depth features using a multi-task learning setting. We also propose to use two novel feature-level loss functions for the self-supervision of the encoder (feature extractor). These loss functions are described in detail in Sects. 5.5 and 5.6.

5.1 Notations

Given an RGB input image $I \in \mathbb{R}^{3 \times H \times W}$, the goal of semantic segmentation is to generate a pixel-level classification map with size $H \times W$, where each pixel is assigned to one of the K categories. Number of categories (K) is 2 (CGL and non-CGL/background) in case of CGL segmentation. Input image is first passed through the common encoder to get encoder feature map denoted by F_e , where $F_e \in \mathbb{R}^{d \times H/4 \times W/4}$. Encoder feature maps are passed to both DFLB and CGLSB that generate label maps \hat{y}_{DFLB} and \hat{y}_{CGL} , respectively, as output, where, $\hat{y}_{DFLB}, \hat{y}_{CGL} \in \mathbb{R}^{2 \times H/4 \times W/4}$. We denote the ground truth segmentation masks for DFLB and CGLSB by y_{DFLB} and y_{CGL} , respectively, where $y_{DFLB}, y_{CGL} \in \mathbb{R}^{H/4 \times W/4}$ (binary maps). Element-wise fusion is performed between the output of first layer of DFLB and that of CGL segmentation block. These intermediate outputs are represented by F_{DFLB}^1 and $F_{CGL}^1 \in \mathbb{R}^{d/4 \times H/4 \times W/4}$, respectively.

5.2 Encoder

To extract rich high-resolution features from the RGB input image, we use HRNetv2 as the encoder. However, in the quantitative results section, we show that our method works well irrespective of the encoder used. Encoder feature maps F_e are passed to both the decoder blocks, DFLB and CGLSB. The loss components from DFLB and the CGLSB together optimize the encoder weights such that the encoder feature maps capture necessary information for satisfying the objectives of both the decoder blocks simultaneously.

5.3 Depth-Aware Feature Learning Block (DFLB)—Auxiliary Decoder

Although the presence of CGL depth pattern alone does not make any region a CGL, we can infer from Fig. 4a that, depth information provides an excellent cue for CGL detection, since there is an abrupt change in depth values across the edges of obscuring items contained in CGLs. Thus, a good encoder should extract features that contain relevant depth information.

To achieve this, we employ an additional auxiliary decoder branch that is trained to perform CGL depth pattern segmentation. As the name suggests, CGL depth pattern segmentation involves pixel-level segmentation of the input image into two classes (One of the classes represents pixels constituting a CGL depth pattern and the other class represents the rest of the pixels). As there is a high correlation between depth information and the desired output for CGL depth pattern segmentation, the segmentation loss for DFLB would compel the common encoder to extract depth-aware features. Hence, the additional decoder has been named Depth-aware Feature Learning Block (DFLB).

The first layer of DFLB is composed of one 2D convolutional layer followed by batch normalization and ReLU. For supervision of DFLB, we generate pseudo ground truth (PGT) as illustrated in Fig. 4b. PGT (y_{DFLB}) is a binary map representing regions having depth pattern similar to that in a CGL by 1s and other regions of the image by 0s. PGT generation involves the following sequence of steps: (1) Pass RGB images to any depth estimation model [12] to get the depth maps (I_{depth}), (2) Smoothen the depth maps, (3) Apply Sobel filter (Canny edge detector does not give superior results) to the smoothed depth map to obtain regions having CGL depth pattern, and (4) Squash the gradient magnitudes using the sigmoid function.

$$g = \phi(G * (K_{avg} * I_{depth})) \quad (1)$$

where $*$ represents the 2D convolutional operation, ϕ is sigmoid function, $G =$

$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ and the smoothing kernel K_{avg} is a 11x11 averaging kernel.

(5) Apply binary thresholding to the output of Step 4.

$$b = \begin{cases} 1, & \text{if } g \geq Th \text{ (where, Th is a gradient threshold)} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

(6) Combine the output obtained in step 5 with the hand-annotated GT (y_{CGL}) using logical OR operation (\oplus), to retain all GT CGL locations in PGT (y_{DFLB}), as

$$y_{DFLB} = b \oplus y_{CGL} \quad (3)$$

The loss function used to train DFLB is given as

$$\mathcal{L}_{DFLB} = \mathcal{L}_{CE}(\hat{y}_{DFLB}, y_{DFLB}) \quad (4)$$

where, \mathcal{L}_{CE} represents cross-entropy loss.

5.4 CGL Segmentation Block (CGLSB)

Part of the proposed architecture (shown in Fig. 3) marked using a dashed cyan box indicates CGLSB. This block is responsible for performing CGL segmentation, with encoder feature maps (F_e) and output of the first layer of DFLB (F_{DFLB}^1) as input. CGLSB comprises the same sub-layers as DFLB but the weights are different for the two blocks, and they are supervised using different ground truth masks. CGLSB is supervised using hand-annotated CGL ground truths encoded as binary segmentation masks (y_{CGL}). In y_{CGL} , pixels inside GT CGL bounding boxes are represented by 1s and other image pixels are represented by 0s. The output of the first layer of DFLB (F_{DFLB}^1) contains depth information about image regions as it is used for final class score estimation by the second layer of DFLB. It (F_{DFLB}^1) helps CGLSB to attend to regions containing CGL depth pattern. The output of this block is generated as given below

$$\hat{y}_{CGL} = f(F_{DFLB}^1 \circ F_{CGL}^1) \quad (5)$$

where f represents the second layer in CGL segmentation block, which computes the final class scores. \circ represents element-wise multiplication.

Loss function used to train CGL segmentation block is given as

$$\mathcal{L}_{CGL} = \mathcal{L}_{CE}(\hat{y}_{CGL}, y_{CGL}) \quad (6)$$

where, \mathcal{L}_{CE} again represents cross-entropy loss.

5.5 Geometric Transformation Equivariance (GTE) Loss

Any geometric transformations performed on the RGB image should get replicated in the feature space as well, i.e., if features for the input image I are F_e and features for the geometrically transformed image are F_e^t , then if we perform the same transformation on F_e , we should obtain F_e^t . In this paper, we have used rotation (by 90°) as the geometric transformation because we want features for horizontally oriented CGLs to be similar to those for vertically oriented CGLs. The loss function is given as follows:

$$\mathcal{L}_{GTE} = \mathcal{L}_{MSE}(F_e^t, F_e^T) \quad (7)$$

where, F_e^T represents the rotated version of F_e . Rotation was performed with the channel dimension as the axis of rotation.

This loss function helps in learning better representations of the input image in a self-supervised manner.

5.6 Intraclass Variance Reduction (IVR) Loss

We want the CGL detection model to focus more on the underlying characteristics of CGL (depth pattern and relationship with respect to neighboring regions of the image) and be robust enough to detect CGLs irrespective of whether the obscuring item was seen during training. So to ensure this, we impose a feature-level constraint using the following loss function:

$$\mathcal{L}_{IVR} = \sum_{i=1}^K \frac{1}{D} \sum_{j=1}^d var(\hat{y}_{CGL}[i] * F_e[j]) \quad (8)$$

where K is the total number of classes (2 for CGL segmentation), and d represents the total number of channels in encoder feature map.

This loss function tries to reduce the intraclass variance in feature space and thus forces the model to focus more on (or extract) features that are absolutely necessary for the task at hand. As a result, the CGL detection model starts to rely more on relevant depth features rather than the features describing the obscuring items. Predicted class probabilities are used to get class information for intraclass feature variance estimation.

Total loss for our model is given as

$$\mathcal{L}_{\text{total}} = \alpha * \mathcal{L}_{CGL} + \beta * \mathcal{L}_{DFLB} + \gamma * \mathcal{L}_{GTE} + \delta * \mathcal{L}_{IVR} \quad (9)$$

Where, $\alpha, \beta, \gamma, \delta$ are hyperparameters.

5.7 Testing

The depth map is not required during testing as it is used just for the supervision of DFLB and the encoder. The output of the model is produced by the CGLSB. DFLB is not used during testing.

6 Results and Experiments

6.1 Evaluation Metric

We use the standard mean Intersection-over-Union (mIoU) metric for the evaluation and comparison of models. In order to compare object detection models with segmentation models, a common evaluation metric is required. So, we convert the output of object detection model into segmentation masks by assigning pixels inside predicted bounding boxes to CGL class (1s) and the rest of the pixels to non-CGL class (0s).

On average, CGLs cover about 10% of the image. So the models usually have a good IoU score for the non-CGL (background) class, which boosts the mIoU score. For this reason, We also report IoU scores for CGL class separately, denoted as CGL IoU.

6.2 Quantitative Results

As evident from Table 1, segmentation models are better at CGL detection. Even though MobileNetv2 has a significantly less number of parameters, it outperforms YOLOv3 on split 2, supporting our hypothesis that segmentation models have an upper hand in CGL detection. mAP_{25} for YOLOv3 is 9.58 on split 1 and 14.5

Table 1 Performance of existing object detection and segmentation models when used for CGL detection. All models were trained from scratch on the proposed CGL detection dataset. Architecture of the decoder (C1) is same as CGLSB

Model	Split 1		Split 2	
	mIoU	CGL IoU	mIoU	CGL IoU
YOLOv3 [6]	52.71	19.28	56.77	26.99
MobileNetv2 + C1 [27]	51.24	15.03	76.72	60.42
HRNetv2 + C1 [29]	55.31	23.36	81.95	69.75

Table 2 Performance of the proposed CGL segmentation model, with different encoders

Encoder	Split 1		Split 2	
	mIoU	CGL IoU	mIoU	CGL IoU
MobileNetv2 [27]	54.19	20.62	78.46	63.40
HRNetv2 [29]	61.95	35.48	83.55	72.38

Table 3 Ablation study using Split 1, showing the effect of each module in the architecture. HRNetv2 was used as the encoder for all these experiments. Table 1 shows the results obtained when all three additional loss components (DFLB, GTE_Loss, IVR_Loss) are absent

DFLB	GTE_Loss	IVR_Loss	mIoU	CGL IoU
✓	✗	✗	57.76	27.64
✗	✓	✗	57.50	26.96
✗	✗	✓	57.97	27.94
✓	✓	✗	58.06	28.05
✓	✗	✓	58.45	28.91
✗	✓	✓	59.04	29.75
✓	✓	✓	61.95	35.48

on split 2. mAP_{50} is 5.81 and 10.6, respectively. YOLOv5 yields similar results as YOLOv3. Base code for segmentation models was taken from [43, 44]. Our proposed method effectively extracts necessary depth information from only an RGB image as input and gives the best results on both data splits (evident from Table 2). However, there is still a lot of scope for performance improvement, providing evidence for the toughness of our dataset and the hardness of the problem we have attempted to solve. The performance is lower on split 1 for all models, as they are subjected to completely unseen environments at test time (in split 1). In split 2, partial overlap exists between train and test set environments.

Table 3 shows how performance varies with the removal of different modules in our architecture. Overall, we observe that the proposed feature-level loss functions complement the supervised segmentation loss, leading to a significant boost in performance.

6.3 Qualitative Results

Figure 5 shows qualitative results for existing as well as the proposed method on split 2. It can be seen that existing models fail to capture the depth pattern in CGLs and rely more on edges in the image to detect CGLs. Our proposed model gives

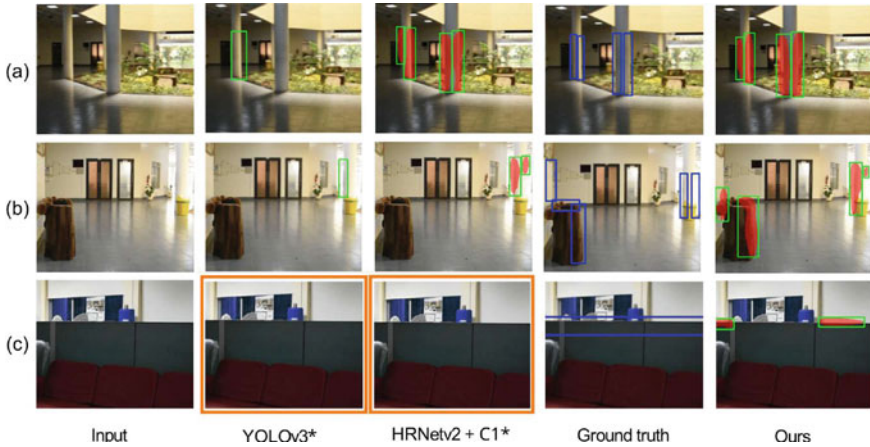


Fig. 5 Qualitative results on split 1. To indicate CGL blobs predicted by segmentation models, we have overlaid those regions of the input image with translucent red-colored masks and enclosed them in green bounding boxes. The last column shows the output of our proposed model with HRNetv2 as the encoder. The orange boxes enclose outputs with no detections. In (c), our model detects only certain portions of the CGL. Qualitative results on Split 2 are included in the supplementary material. [*—existing models adapted and trained for CGL detection]

more importance to depth information and thus it successfully gets rid of the false detections (see example c) made by baseline models. Additionally, YOLOv3 also struggles to perform accurate localization for CGL detection.

7 Conclusion and Discussion

In this paper, we propose a new task termed Covert Geo-Location (CGL) Detection. Given an input image, the goal is to identify and localize Covert Geo-Locations (potential hideouts) in the image. We discuss the importance of this task in visual scene understanding and the AI capabilities required to accomplish this task. We provide a CGL detection dataset containing real-world images captured in diverse environments. We demonstrate the importance of depth information for CGL detection and propose a novel segmentation-based Depth-aware Feature Learning Block (DFLB) that facilitates the extraction of relevant depth features (from only an RGB image as input) required for the proposed task. We also propose two feature-level loss functions which further complement the supervised loss functions. We provide empirical evidence for the superiority of our model over existing object detection and segmentation models (in the absence of any prior work published in this problem domain).

Lately, the effective fusion of RGB and depth information has been an active area of research. Although our task is different from the existing vision-related tasks, and

the ground truth depth information is not available, the existing (or tailored for CGL detection) RGBD segmentation and detection models can also be explored using model-generated pseudo depth map as input along with the RGB image.

References

1. Antol S, Agrawal A, Lu J, Mitchell M, Batra D, Zitnick CL, Parikh D (2015) VQA: visual question answering. In: International conference on computer vision (ICCV) (2015)
2. Caruana R (1997) Multitask learning. *Mach Learn* 28(1):41–75
3. Cornia M, Stefanini M, Baraldi L, Cucchiara R (2020) Meshed-memory transformer for image captioning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10578–10587
4. Cubuk ED, Zoph B, Shlens J, Le QV (2020) Randaugment: practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp 702–703
5. Everingham M, Eslami SMA, Van Gool L, Williams CKI, Winn J, Zisserman A (2015) The pascal visual object classes challenge: a retrospective. *Int J Comput Vision* 111(1):98–136
6. Farhadi A, Redmon J (2018) Yolov3: an incremental improvement. Proceedings of the IEEE conference on computer vision and pattern recognition
7. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
8. Goyal Y, Khot T, Summers-Stay D, Batra D, Parikh D (2017) Making the V in VQA matter: elevating the role of image understanding in Visual Question Answering. In: Proceedings of the IEEE conference on computer vision and pattern recognition
9. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969
10. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
11. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint [arXiv:1704.04861](https://arxiv.org/abs/1704.04861)
12. Hu J, Ozay M, Zhang Y, Okatani T (2019) Revisiting single image depth estimation: toward higher resolution maps with accurate object boundaries. In: IEEE winter conference on applications of computer vision (WACV), pp 1043–1051. IEEE
13. Krishna R, Zhu Y, Groth O, Johnson J, Hata K, Kravitz J, Chen S, Kalantidis Y, Li LJ, Shamma DA, Bernstein M, Fei-Fei L (2016) Visual genome: connecting language and vision using crowdsourced dense image annotations. [arXiv:1602.07332](https://arxiv.org/abs/1602.07332)
14. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
15. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324
16. Li X, Yin X, Li C, Zhang P, Hu X, Zhang L, Wang L, Hu H, Dong L, Wei F et al. (2020) Oscar: object-semantic aligned pre-training for vision-language tasks. In: European conference on computer vision, pp 121–137. Springer
17. Li Y, Ouyang W, Zhou B, Shi J, Zhang C, Wang X (2018) Factorizable net: an efficient subgraph-based framework for scene graph generation. In: Proceedings of the European conference on computer vision (ECCV), pp 335–351
18. Li Y, Ouyang W, Zhou B, Wang K, Wang X (2017) Scene graph generation from objects, phrases and region captions. In: Proceedings of the IEEE international conference on computer vision, pp 1261–1270

19. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125
20. Lin TY, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
21. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: common objects in context. In: European conference on computer vision, pp 740–755. Springer
22. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: single shot multibox detector. In: European conference on computer vision, pp 21–37. Springer
23. Mahajan D, Girshick R, Ramanathan V, He K, Paluri M, Li Y, Bharambe A, Van Der Maaten L (2018) Exploring the limits of weakly supervised pretraining. In: Proceedings of the European conference on computer vision (ECCV), pp 181–196
24. Misra I, Shrivastava A, Gupta A, Hebert M (2016) Cross-stitch networks for multi-task learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
25. Pham H, Dai Z, Xie Q, Luong MT, Le QV (2020) Meta pseudo labels. arXiv preprint [arXiv:2003.10580](https://arxiv.org/abs/2003.10580)
26. Ren S, He K, Girshick R, Sun J (2016) Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE Trans Pattern Anal Mach Intell* 39(6):1137–1149
27. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) Mobilenetv2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520
28. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint [arXiv:1409.1556](https://arxiv.org/abs/1409.1556)
29. Sun K, Zhao Y, Jiang B, Cheng T, Xiao B, Liu D, Mu Y, Wang X, Liu W, Wang J (2019) High-resolution representations for labeling pixels and regions. arXiv preprint [arXiv:1904.04514](https://arxiv.org/abs/1904.04514)
30. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
31. Tan M, Pang R, Le QV (2020) Efficientdet: Scalable and efficient object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10781–10790
32. Tang K, Niu Y, Huang J, Shi J, Zhang H (2020) Unbiased scene graph generation from biased training. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3716–3725
33. Tang K, Zhang H, Wu B, Luo W, Liu W (2019) Learning to compose dynamic tree structures for visual contexts. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6619–6628
34. Tao A, Sapra K, Catanzaro B (2020) Hierarchical multi-scale attention for semantic segmentation. arXiv preprint [arXiv:2005.10821](https://arxiv.org/abs/2005.10821)
35. Xie C, Tan M, Gong B, Wang J, Yuille AL, Le QV (2020) Adversarial examples improve image recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 819–828
36. Yang J, Lu J, Lee S, Batra D, Parikh D (2018) Graph r-cnn for scene graph generation. In: Proceedings of the European conference on computer vision (ECCV), pp 670–685
37. Yun S, Han D, Oh SJ, Chun S, Choe J, Yoo Y (2019) Cutmix: regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 6023–6032
38. Zhang P, Goyal Y, Summers-Stay D, Batra D, Parikh D (2016) Yin and Yang: Balancing and answering binary visual questions. In: Proceedings of the IEEE conference on computer vision and pattern recognition
39. Zhao Q, Sheng T, Wang Y, Tang Z, Chen Y, Cai L, Ling H (2019) M2det: a single-shot object detector based on multi-level feature pyramid network. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 9259–9266

40. Zhou B, Khosla A, Lapedriza A, Torralba A, Oliva A (2015) Places2: a large-scale database for scene understanding. Arxiv preprint:[pending]
41. Zhou B, Lapedriza A, Khosla A, Oliva A, Torralba A (2017) Places: a 10 million image database for scene recognition. *IEEE Trans Pattern Anal Mach Intell* 40(6):1452–1464
42. Zhou B, Lapedriza A, Xiao J, Torralba A, Oliva A (2014) Learning deep features for scene recognition using places database
43. Zhou B, Zhao H, Puig X, Fidler S, Barriuso A, Torralba A (2017) Scene parsing through ade20k dataset. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*
44. Zhou B, Zhao H, Puig X, Xiao T, Fidler S, Barriuso A, Torralba A (2018) Semantic understanding of scenes through the ade20k dataset. *Int J Comput Vision*
45. Zhou L, Palangi H, Zhang L, Hu H, Corso J, Gao J (2020) Unified vision-language pre-training for image captioning and vqa. In: *Proceedings of the AAAI conference on artificial intelligence*, vol 34, pp 13041–13049
46. Zoph B, Ghiasi G, Lin TY, Cui Y, Liu H, Cubuk ED, Le QV (2020) Rethinking pre-training and self-training. arXiv preprint [arXiv:2006.06882](https://arxiv.org/abs/2006.06882)

MATIC: Memory-Guided Adaptive Transformer for Image Captioning



Gaurav O. Gajhiye  and Abhijeet V. Nandedkar

1 Introduction

Image captioning task entails the interpretation of visual contents and its description in natural linguistic manner automatically [6, 13, 23, 24, 27]. In the past few years, the topic has gained popularity in the field of artificial intelligence due to the cross-modal interaction of vision and language modality and abundant research is still being conducted to explore the connectivity between vision-language modeling (e.g. dense captioning, visual question answering, video captioning, and cross-modal retrieval) [1, 9, 12, 25]. The image captioning task was fascinated by sequence learning and machine translation [2] and was primarily addressed by encoder-decoder frameworks. The encoder substantially extracts visual characteristics using variants of convolutional neural network (CNN) and the decoder faithfully constructs the caption using forms of recurrent neural networks (RNN). Further, the attention mechanism [16] was equipped with a CNN-RNN structure to attend to prominent visual regions while generating the word sequence. Despite these advances, the association of objects, attributes, and their intrinsic relationship to describe images remains the topic of intense research.

In CNN-RNN-based structures, the pre-trained CNN methods for visual features were used to capture dominant visual attributes but were incompetent for capturing inherent visual knowledge for captioning. The long-short term memory (LSTM) [10] was commonly utilized as a form of RNN to have a long-term dependency on linguistic patterns and generates the next sample by operating on the hidden state of

G. O. Gajhiye (✉) · A. V. Nandedkar
CVPR Lab, Shri Guru Gobind Singhji Institute of Engineering and Technology (SGGSIE&T),
Nanded, India
e-mail: gajbhiyegaurav@sggs.ac.in

A. V. Nandedkar
e-mail: avnandedkar@sggs.ac.in

the current time step. This regressive nature of LSTM does not allow to parallelize the training procedure.

The novel Transformer [21] architecture has shown the significant potential in addressing the sequence modeling tasks like language generation and translation, as well as multi-modal sequential learning [8]. The standard Transformer is consist of an encoder-decoder model, where the encoder represents the stack of self-attention module followed by feed-forward network and the decoder represents the stack of self and cross attention module followed by feed-forward network. The autoregressive nature of Transformer extends its capability with the stack of attention modules and position-wise embedding of input sequences for parallelism. This motivates to investigate the utility of Transformer for describing the contents of a visual scene by extracting inherent scenery knowledge. Inspired by Cornia et al. [8], this work targets the investigation of memory vectors in a visual encoder to determine the correspondence between objects and attributes using CNN features. In this work, a novel Memory-guided Adaptive Transformer is proposed with a memory-guided encoder for preserving intrinsic visual information received from traditional CNN, while the decoder connects the visual and linguistic features by learning inter-modal association with an adaptive gating mechanism for image description. The overall contributions of the work are as follows:

- Single layer of the memory-guided encoder in conjunction with conventional convolution network is presented for finding the inherent relationship (such as colors, positions, gender, and background) within objects and understand scene attributes by updating memory parameters.
- Multiple layers of the decoder with adaptive multi-headed attention modules, correlate the visual and linguistic pattern by assigning adaptive weightage to spatial and language attention for predicting the future word sample.
- A novel Memory-guided Adaptive Transformer for Image Captioning (MATIC) is proposed by incorporating a single memory-guided encoder layer with multiple adaptive attention decoder layers, and its performance is validated on Flickr8k [19] and Flickr30k [28] dataset.

2 Related Work

The image captioning task became one of the vital issues in artificial intelligence and has been widely addressed by numerous methodologies in the past few years with advancements in deep learning algorithms. In this section, based on the architectural design, the literature is divided into two subgroups as (i) CNN-RNN-based models and (ii) Transformer-based models.

2.1 *CNN-RNN Based Models*

In CNN-RNN-based encoder-decoder methods, CNNs were broadly employed as a visual encoder for spatial-regional characteristics and RNNs were adopted as decoders for the generation of word sequences. In earlier study [11, 13, 17, 23], various levels of CNN were used for spatial features and visual regions extraction and trained with a language model consisting of RNN layers to optimise the likelihood probability of word sequences given the image. The notion of attention in machine translation [2] was utilized in image captioning to attend to prominent visual features aligned with each word in sequence [26]. Later, the purpose attention network was enhanced in [27] by combining attending on visual regions and visual semantic attributes with RNN for better caption prediction. To collect more fine-grained information about visual scenes, channel and spatial-wise attention was introduced with CNN [6]. The adaptive mechanism was combined with attention network [16] for providing substantial weightage to visual and linguistic models in order to generate word sequences.

2.2 *Transformer Based Models*

The transformer model has advanced to the cutting-edge of several essential tasks in the artificial intelligence domain, including image captioning. In Yu et al. [29] CNN-based regional encoder with self-attention has merged with Transformer decoder for transforming visual information into textual captions. The Transformer's decoder section was updated in Zhang et al. [30] to describe the visual contents sequentially, by including an adaptive mechanism in the multi-head attention component leveraging the query vector. Li et al. [14] presented a two-way encoder to process visual and semantic information with EnTangled Attention to generate captions by controlling the flow of visual and semantic knowledge simultaneously. A revolutionary captioning network was developed [8], in which memory vectors were incorporated in the visual encoding layer for acquiring co-relative prior information between image regions, and a mesh-like structure was followed to connect encoder and decoder layer outputs. The scene graphs were built and fused with decoder output using the attention module for sequence generation in Chen et al. [5] to grasp better visual semantics relationship.

3 Methodology

3.1 Overview

This work presents the novel end-to-end attentive architecture of the Memory-guided Adaptive Transformer for Image Captioning (MATIC), which comprises of single-layer encoder and a multi-layered decoder. Figure 1 depicts the overall framework, in which the encoder employs spatial information recovered from CNN and the decoder uses textual features based on FastText embedding to generate caption. The encoder learns inherent relationships within the objects using scenery knowledge of visual features, while the decoder adaptively controls the attentive visual and semantic information by conditioning memory-guided encoder output and embedded textual output.

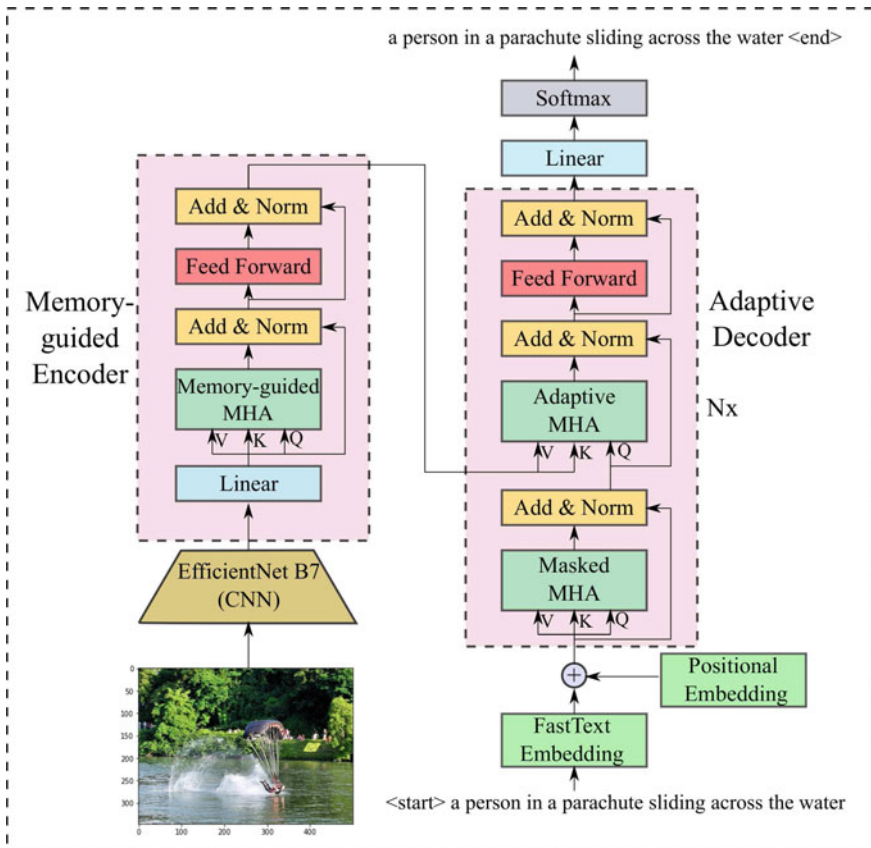


Fig. 1 Memory-guided adaptive transformer for image captioning

3.2 Visual Encoder

The strength of EfficientNet [20] by scaling compound parameters (depth, width and resolution) enhances the efficiency of classification as well as the transfer learning challenge, making it the preferred method for extracting higher-level spatial information from the image. The last convolutional layer employs spatial information by providing the regional feature maps of the image in the form of $V_s = V_1, V_2, \dots, V_D$, where $V_i \in \mathbb{R}^{H \times W}$ (Here, H, W, D represents Height, Width, and Depth of feature maps). Every feature map is flattened to convert the 3D representation into 2D representation, which allows the visual encoder to determine the distinguish relationship within regional features. The 2D representation of spatial features can be rewrite as $V_s \in \mathbb{R}^{F \times D}$, where F represents flatten dimension ($H * W$). These flattened spatial features are provided to the memory-guided encoder to acquire close relationships within various objects.

3.2.1 Memory-Guided Multi-Head Attention

Generally in Transformer, the multi-head attention (MHA) [21] computes the similarity between query (Q) and key (K) vectors and then maps them with value (V) vector to correlate outputs by the parallel projection of the query, key, and value vectors into distinguish head components. It can be represented as follows:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

$$MultiHead(Q, K, V) = Concat(H_1, H_2, ..H_h)W^O \tag{2}$$

$$H_i = Attention(W_i^q Q, W_i^k K, W_i^v V) \tag{3}$$

where d_k is the scaling factor, h represent the number of heads, and W^O, W^q, W^k , and W^v are projected weight parameters.

To preserve complete depthwise regional information, spatial features (V_s) from feature maps with higher dimensionality are linearly projected to intermediate state dimensionality of Transformer model (d_{model}) with ReLU activation. These linearly projected features are further inferred as multi-scale inputs *viz.* Query (Q), Key (K) and Value (V) for memory-guided MHA, where $Q, K, V \in \mathbb{R}^{F \times d_{model}}$. In order to acquire the inherent information within the image, the learnable memory elements (m) are appended to the key and value vector. The spatial input vectors (Q, K, V) are linearly transformed with the projection parameter as:

$$Q = W_{d_{model}}^q Q, K = W_{d_{model}}^k K, V = W_{d_{model}}^v V \tag{4}$$

where $W_{d_{model}}$ represents the projection parameter of Q , K and V . The memory based key (K_m) and value (V_m) vectors are updated by including memory slots (m) of dimension $m \in \mathbb{R}^{m \times d_{model}}$. Thus memory based key and value vectors becomes $K_m, V_m \in \mathbb{R}^{(F+m) \times d_{model}}$:

$$K_m = [K : W_m^k m], \quad V_m = [V : W_m^v m] \quad (5)$$

Here, $[:]$ defines vertical concatenation operation and memory matrix with (m) rows is generated for keys and values by xavier uniform initializer, which gets updated by trainable weight parameters (W_m^k) and (W_m^v) respectively. Analytically, Memory-guided MHA is computed as given below with $Mem_MHA \in \mathbb{R}^{F \times d_{model}}$:

$$Mem_MHA(Q, K, V) = MultiHead(Q, K_m, V_m) \quad (6)$$

3.2.2 Full Encoder

The output of Memory-guided MHA is passed to a position-wise feed-forward network comprising two linear layers with ReLU activation and operates as:

$$FF(x) = W_1(\max(0, W_2x + b)) + c \quad (7)$$

where W_1 and W_2 are outer and internal weight parameters, while b and c are bias terms.

The complete encoder combines Memory-guided MHA and Feed Forward modules by residual additive connection and normalization layer ($Norm$) for yielding encoded output (enc_{out}) as follows:

$$enc_1 = Norm(Mem_MHA(Q, K, V) + Q) \quad (8)$$

$$enc_{out} = Norm(FF(enc_1) + enc_1) \quad (9)$$

3.3 Linguistic Decoder

Following the standard Transformer, the proposed architecture also utilizes N identical layers of decoder, in which multi-modal MHA is modified by including a conditional gating mechanism for weighting the attentive linguistic and spatial information. To acquire the concrete numerical representation of word sequence, FastText [4] model is trained on captions and used to generate word embedding of respective caption ($WE \in \mathbb{R}^{L \times d_{model}}$). Here, L and d_{model} are representing maximum caption length and dimensionality of embedding respectively. In order to access the relative position of the word sequence, positional embedding is added with word embedding output.

The masked MHA sublayer allows the model to attend to all previous time step linguistic information to generate the current time step sample.

3.3.1 Adaptive Multi Head Attention

Inspired from the work [16], adaptive gating mechanism is incorporated with MHA sub-module of Transformer decoder for sequence modeling. The cross-MHA of decoder layer is updated with encoder’s output and attentive previous steps linguistic output. From memory-guided visual encoder, output of feed forward network ($enc_{out} \in \mathbb{R}^{F \times d_{model}}$) is adopted for extracting inherent visual characteristics with relative object information as (V) and (K), while shifted attentive linguistic knowledge from masked MHA of decoder ($dec_1 \in \mathbb{R}^{L \times d_{model}}$) is used as query matrix (Q). The *Attention* in equation (1) is modified by introducing adaptive gating mechanism ($\hat{\beta}$) as shown in Fig. 2 for conditioning spatial knowledge and linguistic pattern. The adaptive gating parameter ($\hat{\beta}$) works similarly as sentinel gate in Lu et al. [16].

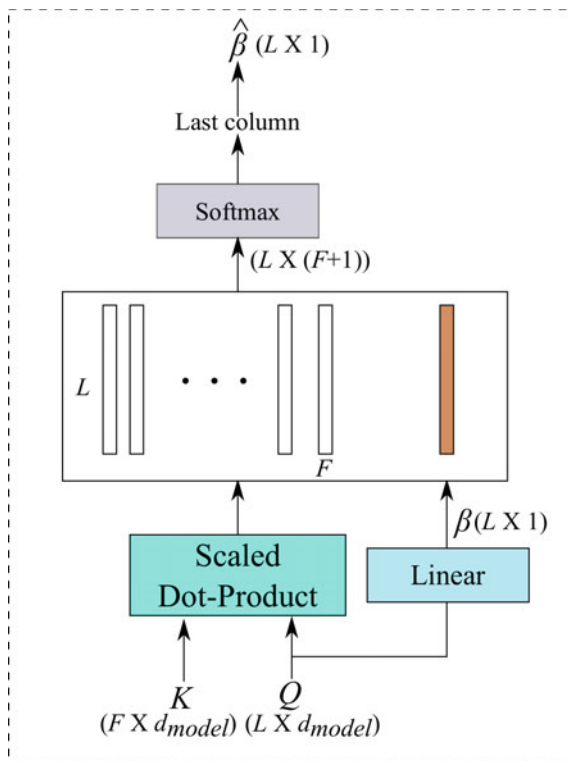


Fig. 2 Adaptive gate

$$Adp_Attention(Q, K, V) = \hat{\beta} * Q + (1 - \hat{\beta}) * Attention(Q, K, V) \quad (10)$$

Here, $(*)$ represents element-wise multiplication and $(\hat{\beta})$ represents scalar value within range $[0, 1]$, where 0 indicates flow of spatial information and 1 implies flow of linguistic knowledge.

In order to co-relate linguistic and spatial information for generation of next word sample, β parameter is introduced, which is computed by projecting the query matrix (Q) linearly into single dimension $(\beta \in \mathbb{R}^{L \times 1})$ as follows:

$$\beta = \tanh(W_{\beta}(Q)) \quad (11)$$

The dot-product (α) is used to obtained adaptive gate $(\hat{\beta})$ using (β) parameter as follows:

$$\alpha = \text{softmax}\left(\left[\frac{QK^T}{\sqrt{d_k}}\right] : \beta\right) \quad (12)$$

$$\hat{\beta} = \alpha[:, -1] \quad (13)$$

Here, $QK^T \in \mathbb{R}^{L \times F}$ and dot-product produces matrix with dimensionality as $\alpha \in \mathbb{R}^{L \times (F+1)}$, from which last column is extracted to retrieve adaptive gate $(\hat{\beta})$. The $(\hat{\beta})$ exhibits the multinomial probability distribution of the query, thus signifying which elements from the query preserve essential linguistic information. Overall, $(\hat{\beta})$ is trained to generate syntactically correct words (e.g. at, on, with, in, through, etc.) by weighting linguistic information, while generating contextual words (e.g. color, gender, position, shape, etc.) by weighting multilevel spatial information.

3.3.2 Full Decoder

The proposed decoder works similarly to a traditional Transformer decoder with autoregressive training properties. The decoder consists of two MHA sub-modules, first module attends on previous textual embedding by hiding future information to generate the current step word sample, while the second module co-relates visual and linguistic patterns. The feed-forward network and residual connections are incorporated to complete the decoder. The FastText embedding of the caption is fed as linguistic input to the decoder, while sinusoidal positional embedding from base transformer [21] is used to co-relate the absolute positioning of each word token. The complete decoder follows the given operations:

$$dec_{emb} = FastText_{emb} + Positional_{emb} \quad (14)$$

$$dec_1 = Norm(Masked_MHA(dec_{emb}, dec_{emb}, dec_{emb}) + dec_{emb}) \quad (15)$$

$$dec_2 = Norm(Adp_Attention(dec_1, enc_{out}, enc_{out}) + dec_1) \quad (16)$$

$$dec_{out} = Norm(FF(dec_2) + dec_2) \quad (17)$$

To generate the sequence of words (caption) for the image, the output of ‘ N ’ layered-decoder module is linearly transformed to vocabulary size ($vocab$) for retrieving the probability distribution of the next word.

$$P(w) = Softmax(W_{vocab}dec_{out}) \quad (18)$$

Here, W_{vocab} defines the trainable weight parameter for vocabulary size.

3.4 Training

The aim of proposed model is to minimize standard cross-entropy loss (\mathcal{L}_{XE}) of word sequence ($y_{1:T}^*$) given spatial features (V_s) of target image as follows:

$$\mathcal{L}_{XE} = - \sum_{t=1}^L \log(P(y_t^* | y_{0:t-1}^*; V_s; \theta)) \quad (19)$$

Here, L is the maximum word length of the caption, while P defines the softmax probability of t -th word as given in Eq. (18) and θ defines model hyper-parameters.

4 Experiments and Results

4.1 Dataset

To evaluate the performance of proposed model, Flickr8k [19] and Flickr30k [28] datasets are utilized, in which each image is associated with 5 human reference captions. Flickr8k is small-scale captioning dataset with 8000 image-caption pairs, while Flickr30k is a large scale captioning dataset with 31783 image-caption pairs. The distribution of training, validation, and test samples are given in Table 1. The maximum word length for captioning is set as 30 and 50 for Flickr8k and Flickr30k datasets respectively. All the captions are transformed to lower case and least occurred words are excluded (less than 3 occurrences for Flickr8k and less than 5 occurrence for Flickr30k) to build the final vocabulary (3427 for Flickr8k and 7037 for Flickr30K). All the images are resized to (300×300) and encoded by the EfficientNetB7 module.

Table 1 Data distribution for Flickr8k and Flickr30K

Dataset	Train	Validation	Test
Flickr8k	6000	1000	1000
Flickr30k	29783	1000	1000

4.2 Training Details

To train the proposed model, the number of heads is set to 8, embedding and submodule dimensionality are selected as 512, while memory slots are varied from 10 to 40 with steps of 10 in memory-guided MHA. The internal layer dimension of the feed forward network is set as 2048. The Adam optimizer with warmup strategy and batch size of 128 is utilized to train the model. The warmup step size and epochs are set as 8000 and 15 and 4000 and 20 for Flickr8K and Flickr30K datasets respectively. Specifically, the proposed MATIC model is trained with a single encoder and four decoder layers ($N = 4$), which ensures better quantitative results. All proposed variants are implemented with TensorFlow 2.2 library on TITAN Xp GPU.

4.3 Quantitative and Qualitative Results

In this section, proposed method is compared with state-of-the-arts and respective quantitative results are summarized. To quantify generated captions, natural language generation (NLG) metrics e.g. n-gram Bleu [18], Meteor [3], Rouge [15] and CIDEr [22] scores are computed from MSCOCO captioning API [7]. In order to generate the fine-level precise caption, heuristic beam search algorithm is employed with beam indexing upto 3. The best metrics outcome are extracted using various beam index and reported in the quantitative results. Table 2 and Table 3 represents the comparative analysis of quantitative results for various methods and proposed MATIC with various memory units (m) on Flickr8k and Flickr30k dataset respectively.

Certain experiments on decoder layers revealed that the proposed MATIC model works better with 4 decoder layers than 6 decoder layers, thus reducing the total trainable parameters and making it a lightweight model. Here, *Base XR* represents the re-implementation of standard Transformer [21] with a single encoder and four decoder layers. Tables 2 and 3 show the quantitative effectiveness of the proposed model for generating captions with 30 memory units for Flickr8k and 40 memory units for Flickr30k dataset.

Table 4 summarises statistical analysis, which shows the amount of trainable parameters and average testing time required by the proposed architecture is approximately equal to that of the Base Transformer. The average testing time is computed by generating captions for 20 test images. With a similar amount of hyper-parameters

Table 2 Comparative analysis of NLG metrics for various methods and proposed MATIC on Flickr8k

Methods	Bleu 1	Bleu 2	Bleu 3	Bleu 4	Meteor	Rouge	CIDEr
DeepVS [13]	57.9	38.3	24.5	16.0	–	–	–
NIC [23]	63.0	41.0	27.0	–	–	–	–
Soft-Att [26]	67.0	44.8	29.9	19.5	18.5	–	–
Hard-Att [26]	67.0	45.7	31.4	21.3	20.3	–	–
g-LSTM [11]	64.7	45.9	31.8	21.2	20.6	–	–
SCA-CNN [6]	68.2	49.6	35.9	25.8	22.4	–	–
Base XR	66.4	47.2	33.1	22.8	21.1	54.2	47.1
MATIC (m = 10)	68.2	49.4	34.9	24.2	22.1	55.6	53.6
MATIC (m = 20)	68.3	49.4	35.1	24.4	22.1	55.4	53.3
MATIC (m = 30)	69.3	50.7	36.5	25.7	23.3	56.1	55.7
MATIC (m = 40)	67.7	48.7	34.3	23.7	22.9	55.3	51.7

XR represents Transformer

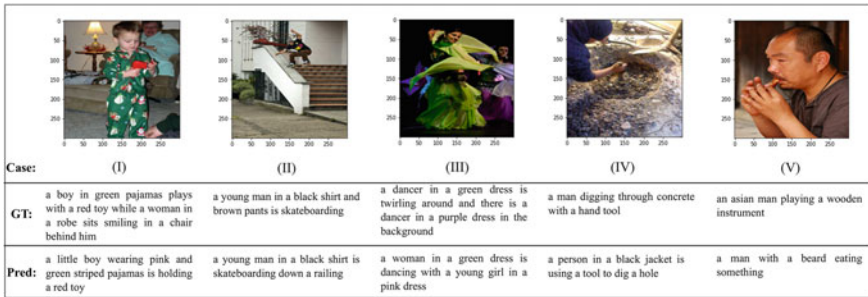


Fig. 3 Generated captions on tricky images from Flickr30K dataset

as of Base Transformer, the MATIC model exhibits a considerable improvement in captioning performance.

To assess the effectiveness of the proposed MATIC model, five tricky test images were chosen from the Flickr30K dataset, and the corresponding generated captions are shown in Fig. 3. The proposed MATIC generates excellent captions for the first four test cases by expressing minute visual contents and relative color-objects-attribute information but misleads in the fifth test instance by generating the incorrect action-based caption. The generated captions by the proposed method demonstrate the application of memory-guided encoder to capture the color, gender, and position of objects, while the adequacy of the adaptive decoder describes all minute details of the image in a semantically convenient manner.

Table 3 Comparative analysis of NLG metrics for various methods and proposed MATIC on Flickr30k

Methods	Bleu 1	Bleu 2	Bleu 3	Bleu 4	Meteor	Rouge	CIDEr
DeepVS [13]	57.3	36.9	24.0	15.7	–	–	24.7
NIC [23]	66.3	42.3	27.7	18.3	–	–	–
mRNN [17]	60.3	41.0	28.0	19.0	–	–	–
Soft-Att [26]	66.7	43.4	28.8	19.1	18.5	–	–
Hard-Att [26]	66.9	43.9	29.6	19.9	18.5	–	–
g-LSTM [11]	64.6	44.6	30.5	20.6	17.9	–	–
Sem-Att [27]	64.7	46.0	32.4	23.0	18.9	–	–
SCA-CNN [6]	66.2	46.8	32.5	22.3	19.5	–	–
Adapt-Att [16]	67.7	49.4	35.4	25.1	20.4	–	53.1
Scene-Graph XR [5]	66.9	49.4	35.4	24.8	20.3	–	53.3
Adapt XR [30]	67.0	49.6	35.5	25.2	20.4	–	53.0
Base XR	66.5	47.5	33.3	23.2	20.8	53.9	48.9
MATIC (m=10)	68.1	48.8	34.6	24.3	20.8	54.0	46.7
MATIC (m=20)	67.2	48.6	35.0	25.0	20.7	54.0	47.6
MATIC (m=30)	68.8	49.4	35.4	24.8	20.7	54.2	49.7
MATIC (m=40)	69.5	50.7	36.5	26.0	20.9	54.9	51.7

XR represents Transformer

Table 4 Statistical analysis of Base Transformer and proposed MATIC on both datasets

	Flickr8k		Flickr30k	
	Base XR	MATIC	Base XR	MATIC
No. of parameters (in Millions)	24.78	25.35	28.48	29.05
Avg. test time (in Seconds)	3.59	3.68	4.63	4.67

5 Conclusion

In this work, a novel Memory-guided Adaptive Transformer for Image Captioning (MATIC) is demonstrated by merging a memory-guided encoder with an adaptive decoder, and its efficacy for generating natural captions is proven on Flickr8k and Flickr30k datasets. Memory-guided encoder is used in conjunction with a conventional CNN network to acquire innate perceptual scenery knowledge, and an Adaptive decoder is employed to align vision-language modality by conditionally weighting visual and semantic information for the generation of word sequence as the caption. In comparison to a typical Transformer with six encoder and six decoder layers, the proposed MATIC has a single memory-based encoder and four adaptive-attention-based decoder layers, which aids in the reduction of overall trainable parameters. As a result, it may be served as a lightweight model and embedded in a device for various captioning applications such as virtual aid, visually impaired individual assisting tools, scene interpretation, and many more. The proposed MATIC has outperformed state-of-the-arts in both quantitative and qualitative findings with 30 memory units for Flickr8k and 40 memory units for the Flickr30K dataset. The proposed MATIC demonstrates the effectiveness of a memory-guided encoder by understanding implicit scenery knowledge aligned with a multi-headed adaptive decoder to describe visual contents in a faithful linguistic manner.

References

1. Anderson P, He X, Buehler C, Teney D, Johnson M, Gould S, Zhang L (2018) Bottom-up and top-down attention for image captioning and visual question answering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 6077–6086
2. Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. CoRR. [arXiv:abs/1409.0473](https://arxiv.org/abs/1409.0473)
3. Banerjee S, Lavie A (2005) METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In: Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, pp 65–72
4. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. *Trans Assoc Comput Linguist* 5:135–146
5. Chen H, Wang Y, Yang X, Li J (2021) Captioning transformer with scene graph guiding. In: 2021 IEEE international conference on image processing (ICIP). IEEE, pp 2538–2542
6. Chen L, Zhang H, Xiao J, Nie L, Shao J, Liu W, Chua TS (2017) Sca-cnn: spatial and channel-wise attention in convolutional networks for image captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 5659–5667
7. Chen X, Fang H, Lin TY, Vedantam R, Gupta S, Dollár P, Zitnick CL (2015) Microsoft coco captions: data collection and evaluation server. [arXiv:1504.00325](https://arxiv.org/abs/1504.00325)
8. Cornia M, Stefanini M, Baraldi L, Cucchiara R (2020) Meshed-memory transformer for image captioning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. IEEE, pp 10578–10587
9. Gao L, Guo Z, Zhang H, Xu X, Shen HT (2017) Video captioning with attention-based LSTM and semantic consistency. *IEEE Trans Multimed* 19(9):2045–2055
10. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780

11. Jia X, Gavves E, Fernando B, Tuytelaars T (2015) Guiding the long-short term memory model for image caption generation. In: Proceedings of the IEEE international conference on computer vision. IEEE, pp 2407–2415
12. Johnson J, Karpathy A, Fei-Fei L (2016) Densecap: fully convolutional localization networks for dense captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 4565–4574
13. Karpathy A, Fei-Fei L (2015) Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp. 3128–3137
14. Li G, Zhu L, Liu P, Yang Y (2019) Entangled transformer for image captioning. In: Proceedings of the IEEE/CVF international conference on computer vision. IEEE, pp 8928–8937
15. Lin CY (2004) Rouge: a package for automatic evaluation of summaries. In: Text summarization branches out, pp 74–81
16. Lu J, Xiong C, Parikh D, Socher R (2017) Knowing when to look: adaptive attention via a visual sentinel for image captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 375–383
17. Mao J, Xu W, Yang Y, Wang J, Huang Z, Yuille A (2015) Deep captioning with multimodal recurrent neural networks (m-rnn)
18. Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, pp 311–318
19. Rashtchian C, Young P, Hodosh M, Hockenmaier J (2010) Collecting image annotations using Amazon’s Mechanical Turk. In: Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon’s Mechanical Turk, pp 139–147
20. Tan M, Le Q (2019) Efficientnet: rethinking model scaling for convolutional neural networks. In: International conference on machine learning. PMLR, pp 6105–6114
21. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser LU, Polosukhin I (2017) Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems, vol 30. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>
22. Vedantam R, Lawrence Zitnick C, Parikh D (2015) Cider: consensus-based image description evaluation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 4566–4575
23. Vinyals O, Toshev A, Bengio S, Erhan D (2015) Show and tell: a neural image caption generator. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 3156–3164
24. Vinyals O, Toshev A, Bengio S, Erhan D (2016) Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. IEEE transactions on pattern analysis and machine intelligence 39(4):652–663
25. Wang B, Yang Y, Xu X, Hanjalic A, Shen HT (2017) Adversarial cross-modal retrieval. In: Proceedings of the 25th ACM international conference on multimedia, pp 154–162
26. Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y (2015) Show, attend and tell: neural image caption generation with visual attention. In: International conference on machine learning. PMLR, pp 2048–2057
27. You Q, Jin H, Wang Z, Fang C, Luo J (2016) Image captioning with semantic attention. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 4651–4659
28. Young P, Lai A, Hodosh M, Hockenmaier J (2014) From image descriptions to visual denotations: new similarity metrics for semantic inference over event descriptions. Trans Assoc Comput Linguist 2:67–78
29. Yu J, Li J, Yu Z, Huang Q (2019) Multimodal transformer with multi-view visual representation for image captioning. IEEE Trans Circuits Syst Video Technol 30(12):4467–4480

30. Zhang W, Nie W, Li X, Yu Y (2019) Image caption generation with adaptive transformer. In: 2019 34rd Youth academic annual conference of Chinese association of automation (YAC). IEEE, pp 521–526

Semantic Map Injected GAN Training for Image-to-Image Translation



Balaram Singh Kshatriya, Shiv Ram Dubey, Himangshu Sarma,
Kunal Chaudhary, Meva Ram Gurjar, Rahul Rai, and Sunny Manchanda

1 Introduction

Deep learning has shown very promising growth in the past decade [7, 18]. Convolutional neural network (CNN) has led to huge progress in deep learning [16]. CNNs have been heavily used to deal with the image data for different applications such as image recognition [13, 16], face recognition [20, 29], medical analysis [4, 31], depth estimation [9, 27], and many more.

A generative adversarial network (GAN) is introduced by Goodfellow et al. [10] for generating new samples for any given data distribution. GAN consists of two networks, namely generator and discriminator. The generator network generates the

B. S. Kshatriya · H. Sarma
Computer Vision Group, Indian Institute of Information Technology, Sri City, Chittoor, India
e-mail: balaramsingh.k18@iiits.in

H. Sarma
e-mail: himangshu.sarma@iiits.in

S. R. Dubey (✉)
Computer Vision and Biometrics Lab, Indian Institute of Information Technology, Allahabad,
India
e-mail: srdubey@iiita.ac.in

K. Chaudhary · M. R. Gurjar · R. Rai · S. Manchanda
DRDO Young Scientist Laboratory-Artificial Intelligence (DYSL-AI), Bangalore, India
e-mail: kunal@dysl-ai.drdo.in

M. R. Gurjar
e-mail: mrgurjar@dysl-ai.drdo.in

R. Rai
e-mail: rahulrai@dysl-ai.drdo.in

S. Manchanda
e-mail: sunny@dysl-ai.drdo.in

samples from the random noise vector. The discriminator network facilitates the training of the generator network by classifying the real and generated samples. The training of both generator and discriminator networks is performed in a min-max optimization fashion. The generator network tries to generate realistic samples using the training distribution such that it can fool the discriminator network. However, the discriminator network tries not to get fooled by the generator network by classifying the generated sample into a fake category. GAN has been utilized for different applications such as data generation [11, 22], medical image synthesis [8, 12], minority class oversampling [24, 28], image hashing [5] among others.

GAN has also shown very appealing performance for image to image translation [1–3, 14, 26, 35]. Broadly, it can be categorized into two parts, viz. paired and unpaired image-to-image translation. In the case of the paired scenario, the source and target images are paired. The Pix2Pix model [14] using conditional GAN relies on the paired data. The paired image-to-image translation requires the paired source and target domain images which is very laborious and infeasible to collect in many real applications. The CycleGAN model [35] is designed for unpaired image to image translation. Two generator and two discriminator networks are utilized by CycleGAN. The forward generator transforms from the source domain to the target while the backward generator transforms from the target domain to the source domain. Thus, an image is transformed from the source domain to the target domain using a forward generator and cycled back to the source domain using a backward generator. A cycle consistency loss is utilized between the original image and the cycled image. A cyclic synthesized loss is included in cyclic synthesized GAN (CSGAN) model [3] between the cycled image in a domain ($A \rightarrow B' \rightarrow A'$) and synthesized image in the same domain ($B \rightarrow A'$). The perceptual loss is used between the original image in a domain and the synthesized image in the same domain by the perceptual cyclic synthesized GAN (PCSGAN) model [1]. The mapping from the source domain to the target domain is constrained in both CSGAN and PCSGAN which is suitable for paired image translation. The CDGAN model [2] improves the CycleGAN framework for unpaired translation by adding the discriminator networks for cycled images. In order to synthesize the objects in a distinguishable fashion, contrastive learning is utilized in the contrastive unpaired translation (CUT) model [26]. The CUT model considers a patch in the generated image and finds the positive and negative pairs of patches from the source domain. It utilizes the contrastive loss on the features of positive and negative pairs to ensure that the patches should be distinguishable from the negative patches which inherently improves the visible quality of the synthesized images. Other GAN models for image to image translation includes DualGAN [33], AttentionGAN [23], CouncilGAN [25], Multi-Scale Gradient-based U-Net (MSG U-Net) [17], etc. The GAN based image to image translation has been also utilized for image colorization [19, 32, 34].

There have been many recent works in the image to image translation which utilize semantic images along with the source domain and the target domain in the translation. Sem-GAN [6] performs image-to-image translation by utilizing the semantic information with the help of the cycleGAN framework. SemGAN includes an additional loss named consistency constraints along with existing GAN loss and cycle

consistency loss. Exemplar-guided unsupervised image-to-image translation with semantic consistency [21] uses feature masks to avoid semantic inconsistency. This allows us to transfer of style information of the target image. Segmentation-guided image-to-image translation with adversarial networks [15] is designed to impose semantic information on the generated images. An additional network named segmentor is incorporated on the existing architecture which provides spatial guidance to the generator network. Example guided style consistent image synthesis from semantic learning [30] is constructed by including a consistency discriminator which functions along with the existing generator and discriminator network to enforce style consistency on the given source images. All these models perform image-to-image translation by including semantic images with the help of additional network(s) to utilize the semantic information. However, the proposed network utilizes semantic information in the training schedule without any additional network.

The existing image-to-image translation methods using GAN are unable to take care of the semantic translation in terms of the structure and color of the object categories. Basically, it is often found that these GAN models of image-to-image translation overfit a particular color or object characteristic across the training dataset, resulting in incoherent translation of the given input image. Specifically, these networks learn the mapping from one domain to other but fail to learn about the regions of each category in the image, which may lead to poor generalization over the test data. It can be visualized in Fig. 1 which illustrates the generated images using CycleGAN [35] in 2nd row and CUT [26] models in 3rd row for the sample input images shown in 1st row. It can be noticed that the green color is overfitted and also some portions of images are still in monochrome (i.e., car in example 2 and buildings in example 3). These examples clearly indicate the limitation of the existing models, which leads to improper image translation.

Motivated by the limitations of the existing GAN models, we propose to utilize semantic map information while training. It can be seen in Fig. 2 that the semantic map can better provide the object specific information. Given the fact that the semantic maps might not be available at test time, the proposed approach uses the semantic map in the form of injected training such that it is not needed at the test time. We observe the improved performance of the proposed semantic injected GAN training approach. Following are the major contributions of this work:

- The proposed approach utilizes the semantic map to learn better category-specific features.
- The proposed approach injects the transformation of the input image to segmentation map during the training of transformation of the input image to the target image.
- The proposed semantic injection improves the generalization ability of image translation.
- The use of semantic maps is required only at the training time not at the test time.
- The performance of the proposed semantic injected training is tested using CycleGAN and CUT models over two benchmark datasets for image-to-image translation shows improved performance.



Fig. 1 Outputs of image to image translation from monochrome to RGB. 1st row: the input images, 2nd row: the generated images using CycleGAN model [35], and 3rd row: the generated images using CUT model [26]

Remaining paper is organized as follows: Sect. 2 presents the proposed method; Sect. 3 summarizes the experimental settings, Sect. 4 illustrates the results and analysis; and Sect. 5 concludes the paper.

2 Proposed Semantic Map Injected GAN Training

The semantic map can be utilized for the training of the GAN models in multiple ways. One of the obvious ways is to simply concatenate the semantic image to the given input image and train the model. This type of training might be effective; however, it leads to the burden of having the semantic map also at the test time, which limits its uses in most real-world applications. Thus, in this paper, we propose a novel way of utilizing the semantic map information at the training time by injecting the semantic map training. Specifically, we alternate the training of GAN models on target images and given semantic images while keeping the same input images in the source domain.

The visual representation of this procedure can be perceived in Fig. 3. Let us consider that the blue block represents the training of the GAN models for

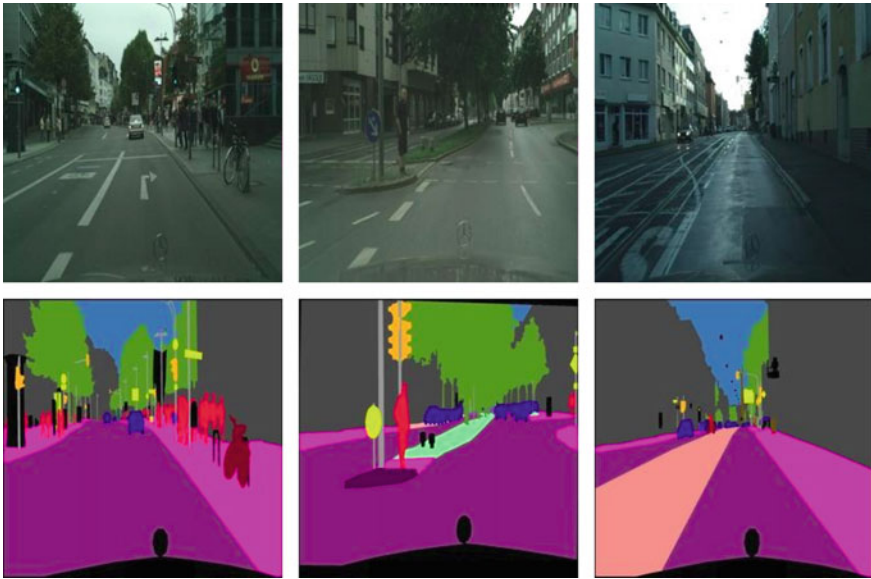


Fig. 2 Samples of semantic maps in the 2nd row corresponding to RGB images in 1st row



Fig. 3 Representation of image to image translation from (left) input domain to target domain and (right) input domain to semantic domain



Fig. 4 Representation of the ratio between the original training and semantic map training (i.e., Original: Semantic). In order (left to right): 90:10, 80:20, 70:30, and 60:40

image-to-image translation from an input domain to the target domain and the red block represents the training of the GAN models for image-to-image translation from the input domain to semantic domain. We propose to inject the semantic training into original training in an interleaved fashion as described in Fig. 4. The injection of semantic training is carried out in the chunks of y number of epochs of training. Thus, based on the number of chunks of semantic training, the ratio between the number of epochs for original training and semantic training varies. If the total number of epochs for training is denoted by n and the number of semantic training injections is s , then the ratio between the number of epochs for original training and semantic training (original:semantic) can be given as follows:

$$\text{original: semantic} = (n - s \times y) : (s \times y) \quad (1)$$

where $(s \times y)$ is the number of epochs of semantic training and $(n - s \times y)$ is the number of epochs of original training. The original:semantic training ratio is one of the hyperparameters in the proposed model. Note that as the semantic training is interleaved with original training, we consider five training models (with a total number of training epochs $n = 100$) with the different original:semantic training ratio. Figure 4 shows the training strategies with different values of original: semantic ratio. Following are the different training schedules used in this paper in terms of the sequence of a number of epochs of original and semantic training:

- 100:0 model (100)
- 90:10 model (45, 10, 45)
- 80:20 model (30, 10, 20, 10, 30)
- 70:30 model (20, 10, 15, 10, 15, 10, 20)
- 60:40 model (15, 10, 10, 10, 10, 10, 10, 10, 15)

where the epochs in blue color represent the number of training epochs for image translation to the target image and red color represents the number of training epochs for image translation to the semantic map. The 100:0 model is without semantic training and is considered for the comparison purpose to show the impact of semantic training injection. The number of semantic training epochs in each chunk (i.e., y) is 10 in the experiments. The number of semantic training chunks (i.e., s) is 0, 1, 2, 3, and 4 in 100:0, 90:10, 80:20, 70:30, and 60:40 training settings, respectively. Note that these training schedules inject the semantic training into the training schedule in an interleaved manner to retain the symmetry and to avoid the biasness towards the semantic translation. Our hypothesis is that the injection of semantic training in this fashion does not deviate from the original training, but rather serves as a regularizer to avoid overfitting and leads to better performance. We also consider another hyperparameter (l) as the learning rate (LR) which is factor between the LR for translation to target image (LR_o) and LR for translation to semantic map (LR_s). Basically, LR_s is given by

$$LR_s = \frac{LR_o}{l}. \quad (2)$$

In this paper, we consider the value of l as 1, 10, and 100 leading to LR Setting 1, LR Setting 10, and LR Setting 100, respectively. The illustration of different LR Settings is shown in Fig. 5 for 80:20 training model.

3 Experimental Setup

Each model is trained for a total of 100 epochs with Learning rate 0.002 ($2e^{-3}$) for a batch size of 2. The resolution of images is 256×256 . The remaining part of this section contains the details of GAN models and datasets used for the

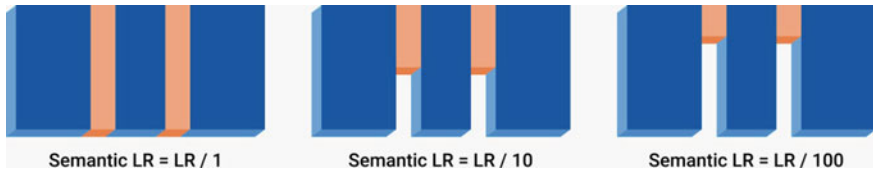


Fig. 5 Representation of learning rate (LR) setting for 80:20 training model. LR Setting 1 (*left*): semantic training LR is same as original training LR, LR Setting 10 (*middle*): semantic training LR is one tenth of original training LR, and LR Setting 100 (*right*): semantic training LR is one hundredth of original training LR

experiments along with the metrics used for evaluation. We use the same loss functions for Semantic to RGB training as used for Monochrome to RGB translation.

3.1 GAN Models Used

In order to depict the impact of the proposed injected semantic training, we use the state-of-the-art GAN models in paired scenarios, including CycleGAN and CUT models. In the paired scenario, the images in different domains are registered. The details of these models are described in below:

CycleGAN Model [35]: The CycleGAN model is used to translate the image from domain A to domain B in an unpaired manner. In order to constrain the mapping, CycleGAN uses two generators, i.e., a forward generator for transformation from domain A to domain B and a backward generator to transform from domain B to domain A. An illustration of CycleGAN is presented in Fig. 6 where an input image

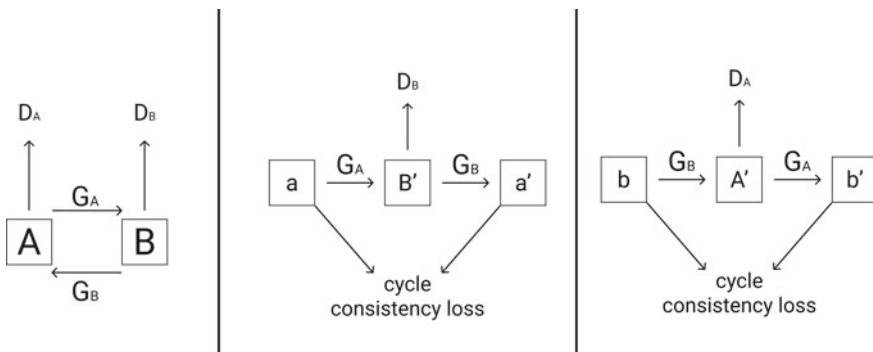


Fig. 6 The illustration of CycleGAN model [35]. The transformation from domain A to domain B is carried out using a generator (G_A) and the transformation from domain B to domain A is carried out using a generator (G_B). Cycle consistency loss is used in both the domains to facilitate the training of CycleGAN model

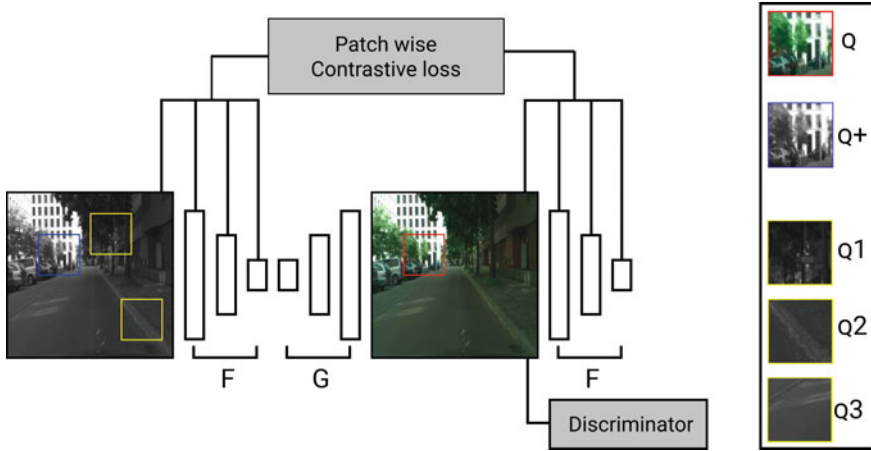


Fig. 7 The illustration of CUT model [26]. The contrastive loss between the positive and negative pairs of patches is used to increase the difference between the visual appearance of positive and negative patches. Note that F and G are Encoder and Decoder networks, respectively

(a) in domain A is transformed into an image (B') in domain B using generator G_A . Further, the generated image (B') is cycled back to the image (a') in domain A. A cycle consistency loss is computed between the original image (a) and cycled image (a') in domain A. Similarly, the image in domain B (b) is transformed to image (A') in domain A using generator G_B and cycled back to the image (b') in domain B using generator G_A . A cycle consistency loss between images b and b' is also computed. CycleGAN also uses two discriminators D_A and D_B in domain A and domain B, respectively. These discriminators distinguish between the generated samples and real samples in the corresponding domains. The final loss for the CycleGAN model includes the adversarial loss and cycle consistency loss in both domains. The adversarial loss consists of generator and discriminator losses. The cycle consistency loss is computed as the reconstruction error.

CUT Model [26]: The contrastive unpaired translation (CUT) model uses the contrastive loss along with the adversarial loss. Basically, it utilizes the similarity between the similar (i.e., positive) and dissimilar (i.e., negative) patches from domain A w.r.t. the patch in domain B. The patchwise contrastive loss is computed by utilizing the features of positive and negative pairs of patches. The features of patches are extracted by the encoder of the generator model. The objective of this loss is to make the synthesized output patch closer to its corresponding input patch and apart from the other input patches. An illustration of the CUT model is presented in Fig. 7. Consider the constructed output patch as Q , the corresponding input patch as $Q+$, and the other random patches as $Q1$, $Q2$, and $Q3$. The final loss in the CUT model is the sum of Adversarial loss of GAN and Patchwise loss in both directions, i.e., from A (input domain) to B (output domain), and vice-versa.

3.2 Datasets Used

In order to demonstrate the image to image translation results, two benchmark datasets are used, namely CityScapes¹ dataset and RGB-NIR² Stereo dataset. The sample images of the datasets are depicted in Fig. 8.

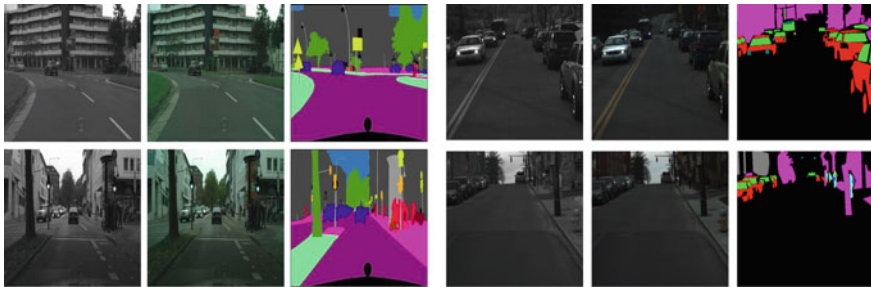


Fig. 8 Example images from CityScapes (1st subfigure) and RGB-NIR stereo (2nd subfigure) datasets. In both subfigures, the input images, target images, and semantic images are shown in left, middle, and right columns, respectively

CityScapes Dataset: CityScapes is a benchmark suite and large-scale dataset used to train and test techniques for semantic characterization at the pixel and instance level. CityScapes contains a wide and varied collection of stereo video sequences shot in 50 cities’ streets. We consider a subset of the dataset containing a total of 2975 training images and 500 test images in each domain. The dataset is a set of both paired and unpaired images of resolution 256×256 . The actual dataset contains only RGB and semantic images. We perform the image colorization over this dataset through image-to-image translation. The source images are monochrome images which are extracted from the corresponding RGB images using openCV. Thus, the final dataset consists of monochrome images, corresponding RGB images, and corresponding semantic images.

RGB-NIR Stereo Dataset: RGB-NIR stereo is a publicly available dataset, which was prepared by collecting 13.7 h of video data in a variety of places in and around a city using a vehicle-mounted RGB-NIR stereo system. The dataset contains materials such as lighting, glass, shiny surfaces, greenery, skin, clothing, and bags and was collected in sunny, overcast, and dark situations on college roads, highways, downtown, parks, and residential areas. This dataset is also a set of both paired and unpaired images and the resolution of images is 582×429 and contains a total of 2100 training images and 900 test images. This dataset contains NIR-Images (monochrome) as the source, corresponding RGB images as the target and corresponding semantic images for the injection training.

¹ <https://www.kaggle.com/dansbecker/cityscapes-image-pairs>.

² <http://www.cs.cmu.edu/~ILIM/projects/AA/RGBNIRStereo/>.

3.3 Metrics Used

In order to evaluate the performance of different models quantitatively, we use the Structural Similarity Index Measure (SSIM), Fréchet Inception Distance (FID), and Kernel Inception Distance (KID) metrics.

SSIM: SSIM measures image quality degradation. SSIM cannot judge the better image among the two, but measures the perceptual difference between the given images. SSIM values are in the range $[0, 1]$, where 1 denotes completely similar images and 0 denotes dissimilar images. For evaluating the performance of the model, we calculate the average of all SSIM values across the test dataset and report in %.

FID: FID stands for Fréchet Inception Distance, which measures the distance between feature vectors calculated for the original and synthetic (generated) images. The score reflects how comparable the two groups of images are in terms of statistics on computer vision aspects of raw pictures calculated with the inception v3 image classification model (inception v3 module represents the given image in a vector of size 2048). Lower scores imply that the two sets of images are more comparable with a perfect score of 0.0 for identical.

KID: KID stands for Kernel Inception Distance, which is a metric similar to FID. KID also calculates the metrics using the inception v3 model representation of given images. FID measures the distance between representations, but KID measures skewness mean and variance between the vector representations. KID uses a polynomial kernel to correct the distributions and has two metrics—KID Mean and KID Variance. Lower scores imply that given two sets of images are better comparable and a perfect score of 0.0 denotes that given images are identical.

4 Experimental Results and Analysis

By considering both the hyperparameters, i.e., original:semantic epochs ratio (i.e., 4 different ratios) and semantic LR factors (i.e., 3 different LR settings), we test 12 combinations. The results using CUT model are reported over CityScapes dataset in Table 1 in terms of the SSIM and FID and in Table 2 in terms of the KID Mean and KID Variance. The results using CUT model are reported over RGB-NIR stereo dataset in Table 3 in terms of the SSIM and FID and in Table 4 in terms of the KID Mean and KID Variance. In each table, the best result is highlighted in **bold**. In each Table, 1st, 2nd, and 3rd row results correspond to LR settings 1, 10, and 100, respectively. Whereas, the results for different original:semantic training ratios are reported in the corresponding columns. It is observed from these results that the performance of the proposed semantic map injected training is better than the original training over CityScapes dataset in terms of the SSIM, FID, and KID measures. Moreover, an improved performance is also observed on the RGB-NIR dataset by the proposed training scheme in terms of the SSIM. The training schedule with ratios 80:20 and

Table 1 The results in terms of the SSIM and FID scores using CUT model over CityScapes dataset for different original:semantic training ratio under different learning rate (LR) setting. Note that LR Setting l means the learning rate for semantic training is $1/l$ of RGB training. The best results are highlighted in bold

LR	SSIM					FID				
Setting	100:0	90:10	80:20	70:30	60:40	100:0	90:10	80:20	70:30	60:40
1	93.78	91.91	94.35	94.53	93.88	32.50	36.43	36.59	38.50	36.30
10	93.78	93.22	93.89	93.81	94.28	32.50	32.16	29.75	34.12	47.25
100	93.78	94.06	94.14	94.28	94.49	32.50	29.76	33.36	34.51	38.16

Table 2 The results in terms of the KID Mean (KIDm) and KID Variance (KIDv) scores using CUT model over CityScapes dataset for different original:semantic training ratio under different learning rate (LR) setting

LR	KID mean					KID variance				
Setting	100:0	90:10	80:20	70:30	60:40	100:0	90:10	80:20	70:30	60:40
1	0.0105	0.0111	0.0125	0.0169	0.0134	0.0006	0.0007	0.0007	0.0007	0.0005
10	0.0105	0.0091	0.0059	0.0104	0.0292	0.0006	0.0006	0.0004	0.0005	0.0011
100	0.0105	0.0060	0.0110	0.1285	0.0182	0.0006	0.0004	0.0007	0.0007	0.0006

Table 3 The results in terms of the SSIM and FID scores using CUT model over RGB-NIR stereo dataset for different original:semantic training ratio under different learning rate (LR) setting

LR	SSIM					FID				
Setting	100:0	90:10	80:20	70:30	60:40	100:0	90:10	80:20	70:30	60:40
1	74.98	73.74	78.62	77.56	76.97	26.93	31.28	28.52	29.53	30.41
10	74.98	77.40	75.50	77.27	75.82	26.93	29.54	29.37	29.03	34.32
100	74.98	77.58	77.62	76.70	74.65	26.93	29.57	29.19	29.18	32.99

Table 4 The results in terms of the KID Mean (KIDm) and KID Variance (KIDv) scores using CUT model over RGB-NIR stereo dataset for different original:semantic training ratio under different learning rate (LR) setting

LR	KID mean					KID variance				
Setting	100:0	90:10	80:20	70:30	60:40	100:0	90:10	80:20	70:30	60:40
1	0.0050	0.0093	0.0070	0.0069	0.0090	0.0005	0.0008	0.0005	0.0005	0.0006
10	0.0050	0.0079	0.0078	0.0070	0.0129	0.0005	0.0005	0.0007	0.0005	0.0008
100	0.0050	0.0073	0.0072	0.0074	0.0104	0.0005	0.0006	0.0007	0.0008	0.0008

Table 5 The results in terms of the SSIM, FID, KID Mean (KIDm) and KID Variance (KIDv) scores using CycleGAN model over CityScapes and RGB-NIR stereo datasets for different original:semantic training ratio

Metric	CityScapes dataset					RGB-NIR dataset				
	100:0	90:10	80:20	70:30	60:40	100:0	90:10	80:20	70:30	60:40
SSIM	94.02	94.21	94.89	94.67	94.17	76.46	76.95	79.51	77.79	74.41
FID	34.10	30.86	29.50	30.35	33.34	40.03	40.88	40.91	45.28	53.02
KIDm	0.0088	0.0071	0.0061	0.0058	0.0068	0.0127	0.0131	0.0155	0.0181	0.0239
KIDv	0.0005	0.0003	0.0003	0.0003	0.0003	0.0006	0.0008	0.0008	0.0007	0.0010

70:30 shows better improvement than 100:0. LR setting 1 is better suited for SSIM, and LR setting 10 is better suited for other metrics.

The results in terms of the SSIM, FID, and KID measures using the CycleGAN model over CityScapes, and RGB-NIR datasets are reported in Table 5 for different original:semantic training ratios. In this experiment the LR setting 1 is followed. A clear improvement is gained using the proposed semantic map injection training over the CityScapes dataset. However, the result over RGB-NIR is also improved in terms of the SSIM.

The qualitative results in terms of the generated samples from the CityScapes dataset are illustrated in Fig. 9 using CUT and CycleGAN models for 100:0, 70:30/80:20 training settings. The quality of the generated images is very appealing. It can be seen that semantic learning helps the model to learn the various regions in the given image. It can be also observed that the original model tends to overfit a particular color without accounting for the region of that particular object. Not only the visual results but also the metric values indicate that semantic learning helps the model to generalize better and produces better quality images. Note that the semantic map is not required at the test time.

5 Conclusion

In this paper, we have proposed a semantic map injected training of GAN models for image-to-image translation. The semantic map training injection is performed in an interleaved manner with the original training. The proposed method ensures that the semantic map is not required at test time. The proposed semantic map injection training improves the generalization of the model and reduces overfitting. This approach is tested over CityScapes and RGB-NIR stereo benchmark datasets using CycleGAN and CUT models. We found that 30 and 20% of semantic injection lead to better performance as compared to the vanilla training. Both quantitative and qualitative results point out that semantic map injection training helps the model to understand the different regions of objects, and thus results in better-translated



Fig. 9 The qualitative results on CityScapes dataset. 1st *column*: Ground truth images, 2nd *column*: Generated images using CUT model without semantic training, 3rd *column*: Generated images using CUT model with proposed semantic training having 70:30 epochs for original:semantic, 4th *column*: Generated images using CycleGAN model without semantic training, and 5th *column*: Generated images using CycleGAN model with proposed semantic training having 80:20 epochs for original:semantic

images. The limitation of the proposed model is the need of a semantic map at the training time. The future direction includes the extension of this work on videos and reduction of parameters leading to lightweight model for real-time applications.

Acknowledgements This research is funded by DRDO Young Scientist Laboratory-Artificial Intelligence (DYSL-AI), Bangalore, through grant no. DYSL-AI/01/2020-2021.

References

1. Babu KK, Dubey SR (2020) PCSGAN: perceptual cyclic-synthesized generative adversarial networks for thermal and NIR to visible image transformation. *Neurocomputing* 413:41–50
2. Babu KK, Dubey SR (2021) CDGAN: cyclic discriminative generative adversarial networks for image-to-image transformation. *J Vis Commun Image Represent*

3. Babu KK, Dubey SR (2021) CSGAN: cyclic-synthesized generative adversarial networks for image-to-image transformation. *Expert Syst Appl* 169:114431
4. Basha SS, Ghosh S, Babu KK, Dubey SR, Pulabaigari V, Mukherjee S (2018) Rccnet: an efficient convolutional neural network for histological routine colon cancer nuclei classification. In: 15th International conference on control, automation, robotics and vision (ICARCV), pp 1222–1227
5. Cao Y, Liu B, Long M, Wang J (2018) Hashgan: deep learning to hash with pair conditional wasserstein gan. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1287–1296
6. Cherian A, Sullivan A (2019) Sem-GAN: semantically-consistent image-to-image translation. In: 2019 IEEE winter conference on applications of computer vision (WACV). IEEE, pp 1797–1806
7. Dubey SR (2021) A decade survey of content based image retrieval using deep learning. *IEEE Trans Circuits Syst Video Technol*
8. Frid-Adar M, Diamant I, Klang E, Amitai M, Goldberger J, Greenspan H (2018) GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing* 321:321–331
9. Garg R, Bg VK, Carneiro G, Reid I (2016) Unsupervised cnn for single view depth estimation: Geometry to the rescue. In: European conference on computer vision, pp 740–756. Springer, Berlin
10. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville AC, Bengio Y (2014) Generative adversarial nets. In: NIPS
11. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (2017) Improved training of wasserstein gans. In: Proceedings of the 31st international conference on neural information processing systems, pp 5769–5779
12. Han C, Hayashi H, Rundo L, Araki R, Shimoda W, Muramatsu S, Furukawa Y, Mauri G, Nakayama H (2018) GAN-based synthetic brain MR image generation. In: 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018). IEEE, pp 734–738
13. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
14. Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1125–1134
15. Jiang S, Tao Z, Fu Y (2019) Segmentation guided image-to-image translation with adversarial networks. In: 2019 14th IEEE international conference on automatic face & gesture recognition (FG 2019). IEEE, pp 1–7
16. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Adv Neural Inf Process Syst* 25:1097–1105
17. Laxman K, Dubey SR, Kalyan B, Kojjarapu SRV (2021) Efficient high-resolution image-to-image translation using multi-scale gradient U-net. In: Sixth IAPR international conference on computer vision and image processing (CVIP2021)
18. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436–444
19. Lee J, Kim E, Lee Y, Kim D, Chang J, Choo J (2020) Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 5801–5810
20. Liu W, Wen Y, Yu Z, Li M, Raj B, Song L (2017) Sphereface: deep hypersphere embedding for face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 212–220
21. Ma L, Jia X, Georgoulis S, Tuytelaars T, Van Gool L (2018) Exemplar guided unsupervised image-to-image translation with semantic consistency. [arXiv:1805.11145](https://arxiv.org/abs/1805.11145)
22. Mao X, Li Q, Xie H, Lau RY, Wang Z, Paul Smolley S (2017) Least squares generative adversarial networks. In: Proceedings of the IEEE international conference on computer vision, pp 2794–2802

23. Mejjati YA, Richardt C, Tompkin J, Cosker D, Kim KI (2018) Unsupervised attention-guided image-to-image translation. In: *NeurIPS*
24. Mullick SS, Datta S, Das S (2019) Generative adversarial minority oversampling. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp 1695–1704
25. Nizan O, Tal A (2020) Breaking the cycle-colleagues are all you need. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 7860–7869
26. Park T, Efros AA, Zhang R, Zhu JY (2020) Contrastive learning for unpaired image-to-image translation. In: *European conference on computer vision*, pp 319–345. Springer, Berlin
27. Repala VK, Dubey SR (2019) Dual CNN models for unsupervised monocular depth estimation. In: *International conference on pattern recognition and machine intelligence*, pp 209–217. Springer, Berlin
28. Roy SK, Haut JM, Paoletti ME, Dubey SR, Plaza A (2021) Generative adversarial minority oversampling for spectral-spatial hyperspectral image classification. *IEEE Trans Geosci Remote Sens*
29. Srivastava Y, Murali V, Dubey SR (2019) A performance evaluation of loss functions for deep face recognition. In: *National conference on computer vision, pattern recognition, image processing, and graphics*, pp 322–332
30. Wang M, Yang GY, Li R, Liang RZ, Zhang SH, Hall PM, Hu SM (2019) Example-guided style-consistent image synthesis from semantic labeling. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 1495–1504
31. Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM (2017) Chestx-ray8: hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 2097–2106
32. Wu Y, Wang X, Li Y, Zhang H, Zhao X, Shan Y (2021) Towards vivid and diverse image colorization with generative color prior. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp 14377–14386
33. Yi Z, Zhang H, Tan P, Gong M (2017) Dualgan: unsupervised dual learning for image-to-image translation. In: *Proceedings of the IEEE international conference on computer vision*, pp 2849–2857
34. Zhang R, Isola P, Efros AA (2016) Colorful image colorization. In: *European conference on computer vision*, pp 649–666. Springer, Berlin
35. Zhu JY, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Proceedings of the IEEE international conference on computer vision*, pp 2223–2232

TextGen3D: A Real-Time 3D-Mesh Generation with Intersecting Contours for Text



Ankit Dhiman, Praveen Agrawal, Sourav Kumar Bose,
and Basavaraja Shanthappa Vandrotti

1 Introduction

Humans, as far as we know, have been using signs to express themselves. These signs have evolved into different languages over the years. Also, humans learnt the way to preserve these signs in various forms. Early humans used walls, caves, trees, etc. to conserve their expressions. Later on, we expressed these signs using “paper” as a medium. With the advent of the Internet, digital mediums have become more popular to express one’s thoughts and feelings. In recent times, the 2D text on screen has evolved into 3D text, having its applications in banners, online advertisements, movies, etc. Hence, it’s only logical to extend this experience of visualizing or interacting with the 3D text in Augmented Reality (AR).

AR has been picking up popularity amongst users over the past few years. With AR, augmenting virtual objects into the real world has been achieved. In recent years, AR platforms like ARKit [8] and ARCore [9] have become popular, and doodling has emerged as a ubiquitous feature. This popularity has led to the need for 3D text in AR applications, although the 3D text exists as a standalone feature in other tools such as professional tools like Autocad, Blender, Adobe Photoshop, etc. The existing solutions use pre-loaded 3D mesh for language characters. Hence, these solutions can handle limited variations of font types and languages in the input text. Thus, real-time generation of 3D mesh for any input text is of paramount importance for enabling Text in AR space.

This paper presents an end-to-end approach for generating 3D text. This paper also highlights the problems which arise due to handling different font types and languages. The major problems addressed by this paper are as follows:

A. Dhiman (✉) · P. Agrawal · S. K. Bose · B. S. Vandrotti
Samsung R&D Institute, Bangalore 560037, Karnataka, India
e-mail: ankit.dh.1992@gmail.com
URL: <https://research.samsung.com/sri-b>

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022
U. Mudenagudi et al. (eds.), *Proceedings of the Satellite Workshops of ICVGIP 2021*,
Lecture Notes in Electrical Engineering 924,
https://doi.org/10.1007/978-981-19-4136-8_17

251

1. How to reduce the number of points in an input contour while maintaining the curvature of the original curve?
2. How to remove intersections which occur within a contour itself and with a neighbouring contour?
3. How to identify holes in the text contour?
4. How to produce round edges in a mesh to improve the overall aesthetic?

2 Literature Review

There is a growing number of 3D fonts available on the Internet. Applications can use these 3D fonts to render 3D text. There exist over 7000 languages in the world. Hence, it is a mammoth task to create uniform font types for all the languages. An alternate method is to convert an input text into 2D contours and then generate 3D text from it. Slabaugh et al. [22] propose a method to create a 3D text on 3D triangulated surfaces from a text contour. This method back-projects the 2D vertices of the text contour on the 3D triangulated surface. It then produces a protruded 3D text label on the 3D surface. This work doesn't discuss the intersection between two contours and rounded edges.

With the advancement in deep learning approaches, generating a 3D model from a 2D sketch has seen a lot of improvement in methods like [11, 24, 26, 27]. Jin et al. [11] convert a 2D input contour to its 3D model representation. Jin et al. [11] use variational auto-encoders to encode contours into a latent vector and then into a 3D model. The method doesn't show any results to generate 3D models for an extracted contour from an input text.

There exist methods that correct invalid contours. Lai et al. [13] propose the forward locus tracing method (FLTM) for invalid contour removal. Lai et al. [13] eliminate degenerated segments by the Voronoi method. FLTM presents a 1D interval identification problem that works on a hypothesis that if the first interval has no interference with other objects, then the odd intervals group can form valid loops. Lin et al. [15] find a problem in Lai et al. [13] where the intersection removal can go wrong between two adjacent inner angles. Lin et al. [15] propose a tree analysis (TA) that removes all the invalid global loops in a contour.

Two contours can intersect with each other. In computer graphics, clipping is a ubiquitous method that is present in some form or the other in different applications. Reference [2, 17, 21] are methods to clip a polygon against a rectangular or convex polygon. [6, 16, 20, 23] presents a more generalized algorithm that clips a polygon against concave polygons as well. Greiner et al. [6] presents a polygon clipping method that can handle self-intersections and is faster than Vatti's [23] method. Liu et al. [16] propose a data structure that requires less memory and efficiently clips the polygon.

Occasionally, there are self-intersections in a 3D mesh which leads to unaesthetic visualization. Wong et al. [25] remove these self-intersections by applying volume simulation to the 3D mesh. Professionals use methods like [12] to identify the self-

intersections in a 3D mesh and then repair the identified regions by either merging or decimating such triangles in the 3D mesh. Attene et al. [3] propose a method to remove degenerate and intersecting elements in a 3D mesh by modifying the mesh locally within the neighbourhood of undesired configurations to produce a watertight triangle mesh.

3 Algorithm

The proposed pipeline has three stages: (1) Contour Generator, (2) Mesh generation and (3) Mesh Transformation, as shown in Fig. 1.

3.1 Contour Generator

Input to the proposed pipeline is a text which has variations in language, font, alignment and number of lines. To create a mesh for such input, it needs to be represented in the language that the mesh generation algorithm understands which is a set of points. This is achieved by a font -library which represents text as smooth continuous curves. The contour-generator samples point from these 2D curves and yield closed 2D contours.

This brings to the problem that is addressed in this section: *Number of points versus Time Performance*. High sampling rate ensures high mesh quality but increases the processing time which is a significant factor for real-time applications. To address this trade-off of quality and processing time, the pipeline uses a simple yet effective algorithm as explained in Algorithm 1 which decimates the point along a straight line and preserves points along a curvature. Figure 3b shows the results of the discussed dynamic sampling strategy

3.2 Contour Processor

After reducing points in the set of input contours from the previous stage, the pipeline needs to determine the relationship between contours in the input set. Additionally,

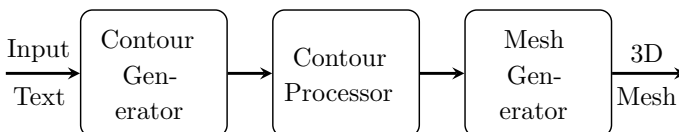


Fig. 1 Proposed pipeline

Algorithm 1: Reduce points in an input contour

Output: \emptyset
for $p \in \text{Input Contour}$ **do**

 p1 \leftarrow previous point to p;

if $\text{Angle}(p1, p) \geq \text{threshold}$ **then** // Angle between two points

- └ Add p to Output

the next stage in the pipeline, mesh triangulation, uses Delaunay's triangulation [14], which assumes the input contours to have the following properties:

1. A contour should not have a self-loop.
2. Two holes inside another contour should not intersect with each other.
3. Holes and their parent contour should not intersect with each other.
4. Either parent contours should not intersect with each other as they will look fallacious.

If the input contours fail these properties, this may lead to erroneous outputs from the triangulation pipeline. Our method alleviates this issue by dividing such scenarios into two categories: Self-intersecting contours (a contour which forms a self-loop) as shown in Fig. 3c; and contours which intersects with other contours as shown in Fig. 3d.

A contour has self-loops when two line segments in a contour have an intersection with each other as shown in Fig. 2a and zoomed-in region of intersection as shown in Fig. 2b. As discussed in Algorithm 2, the method breaks the intersecting point into two new points lying on the angle bisector at the intersection point and outside the original contour. Figure 2d shows the corrected contour and Fig. 2e shows the zoomed-in region. Figure 2e demonstrates the result discussed in Algorithm 2. Figure 2c and f shows the triangulated mesh with the intersecting contour and the same contour after removing the intersection, respectively.

To remove the intersections in overlapping contours, first, the pipeline needs to detect these intersections. Additionally, intersection points between two contours will always happen in a set of pairs, i.e. overlapping contours will have an even number of intersecting points. Algorithm 3 describes the algorithm to generate a relationship graph between contours and remove the overlap between two contours.

Algorithm 2: Removes self-loop in a contour

Input: Contour P (Set Of Points)

for $p \in P$ **do**
for $q \in P, q \neq p$ **do**
if *Line-segments* $(p, \text{Next}(p))$ and $(q, \text{Next}(q))$ *intersects* **then**

- └ Generate two points m and n at the location of intersection Reverse order of points between p and q;

- └ Insert new nodes in the contour;

└

└

└

└

└

└

└

└

└

└

└

└

└

└

└

└

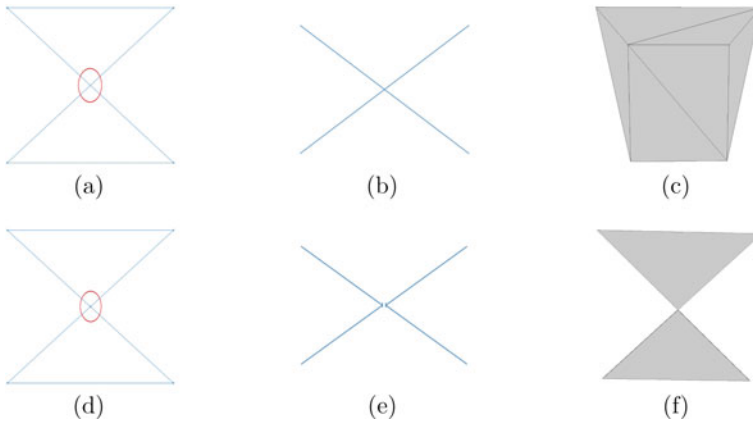


Fig. 2 **a** Contour with self-intersection, **b** Zoomed-in region marked in red in **(a)**, **c** Result after mesh triangulation for **(a)**, **d** Contour after removing self-intersection, **e** Zoomed-in region marked in red in **(d)** and **f** Result after mesh triangulation for **(d)**

This relationship graph is a DAG. From our observations, we found that if you traverse DAG nodes, nodes at even levels are hole contours.

3.3 Mesh Generator

The pipeline receives corrected contours from the previous stage. Also, the contour processor clusters polyline and their associated holes from the input contour set. Delaunay’s algorithm [14] triangulates the set of points in polyline and hole contours. For example, the English character “O” has two contours; the outer contour is a polyline contour and the inner contour is a hole for this primary contour. Thus, the triangulation process will only happen in the region between these two contours.

The pipeline generates triangles for the front plane using the aforementioned process in anti-clockwise winding order [7]. For generating the back-plane mesh, the pipeline extrudes the front-plane triangles at a fixed offset and inverts the winding order as shown in Fig. 4a. This is an efficient way to generate mesh for the front plane and back plane of the input text. To complete the mesh, the next step is to create triangles for the side planes.

The pipeline uses a rounded-edge generation algorithm to generate an aesthetically pleasing 3D mesh. The algorithm generates triangles along the side face of the mesh in a bevelled form such that it simulates a rounded look upon proper normal generation as shown in Fig. 4. This leads to issues like holes and overlapping triangles in the mesh. The concave outward sides lead to the creation of holes. This is resolved by generating additional triangles to fill the holes. The concave inward sides lead to

Algorithm 3: Generate a graph of connected contours()

```

Procedure RemoveIntersection (query, base) :
  - Mark points in query which are inside base;
  - Mark points in base which are inside query;
  - Find the set of intersecting points ;
  if query outside base;
  then
    | Remove points in query which are inside base;
    | Remove edge between base and query in relationship graph;
  else
    | Remove points in query which are outside base;
    | Add full-weight edge between base and query in relationship graph;
  - Generate new points along boundary of base contour;
;
Result: A Relationship Graph Between Contours
Input:  $\omega$  (Set Of contours)
for  $c \in \omega$  do
  for  $b \in \omega, b \neq c$  do
    | if b outside c;
    | then
    | | no connection in graph;
    | else if b partially inside c;
    | then
    | | ResolveIntersection(b,c);
    | else if b inside c;
    | then
    | | Add directed edge from c to b with full weight;
  
```

overlapping side triangles. This is resolved by sharing the vertices generated for the bevel.

Vertex-normal is a geometric normal of the mesh created in the previous stage. These normals are an integral attribute for shading on the 3D mesh. Generally, a normal for a vertex is computed by taking a normalized average of the surface normals of the faces containing this vertex. This will yield aberrated shading for the generated 3D mesh because it has triangles of varying sizes, from small to large to very thin triangles. Thus, the pipeline requires an algorithm which handles such variations in the area of the generated triangles. The pipeline uses an algorithm proposed by [18] to estimate the vertex-normal to approximate a smooth surface, as a weighted sum of the vertex-normal to the facets surrounding the vertex.

In the graphics pipeline, texture mapping (UV Mapping) is a process to map a 2D image onto a 3D surface. Techniques that are used for texture mapping are listed as follows: (1) Spherical mapping, (2) Cylindrical Mapping and (3) designing the mapping by a designer [1]. The pipeline uses a cuboidal mapping to determine UV coordinates of the generated mesh as shown in (To do insert figure). As text-width can vary based on different inputs, the texture repeat rate is proportional to text-width

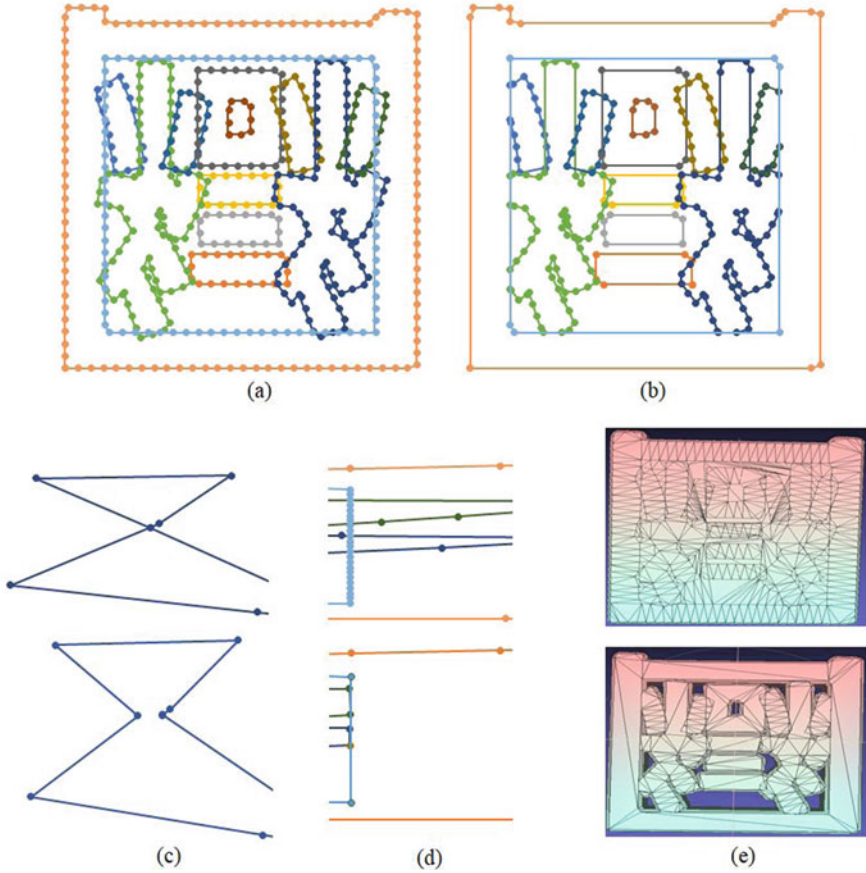


Fig. 3 (Best viewed in colour) **a** Input 2D contours, **b** contour points after dynamic sampling, **c** zoomed-in region of self-intersection (region in marked red circle), **d** zoomed-in region of intersection with other contours (region in marked red rectangle) and **e** final output of the 3D

in the x-direction and is constant in the y-direction. For back-plane vertices, an offset is added to the x-coordinate of corresponding front-plane vertices.

4 Results

In this section, we present results from the proposed pipeline. We discuss the results of the following modules: 1. Dynamic Sampling, 2. Intersection Removal and 3. Quality of the generated mesh. We implement the proposed pipeline on an embedded system with Qualcomm SM8250 Snapdragon chip (Octa-core: one 2.84 GHz, three 2.42 GHz and four 1.8 GHz processors) running Android 10 with Adreno 650 GPU(587 MHz).

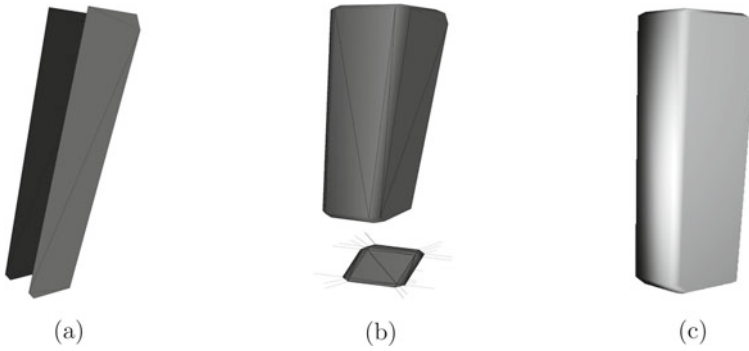


Fig. 4 a Front- and back-plane mesh, b Sides of the rounded edges and top view with vertex-normals and c Mesh with phong shading



Fig. 5 Mesh generated of an excerpt from *To Kill A Mockingbird* in a English, b Simplified Chinese, c Hindi, d French, e Spanish and f Arabic. Snipped from a mesh visualization tool

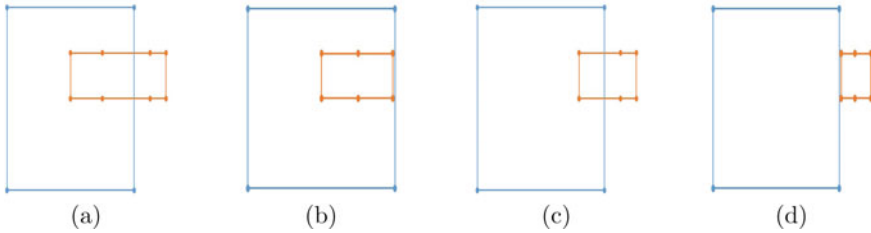


Fig. 6 Intersection removal between two contours

OpenGL pipeline on embedded system renders the mesh. MeshLab [4] is used to visualize the 3D mesh.

4.1 Dynamic Sampling

As discussed in Sect. 1, the dynamic sampling strategy removes redundant points to improve the time performance. We choose an excerpt from the novel *To Kill a Mockingbird* and translated it into Hindi, simplified Chinese, French, Spanish and Arabic using Google translate utility [10] available on the web as shown in Fig. 5. Table 1 shows the improvement in time performance and reduction in the number of vertices in the generated mesh. The dynamic sampling strategy reduces the time performance significantly in English, Chinese, Hindi, Spanish and French languages where characters have less curvature compared to the Arabic language where the impact is less because of more curvatures.

4.2 Intersection Removal

The contour processor module resolves self-intersection in a contour and clips the intersection part of a contour with another contour. Figure 3c illustrates an example of self-intersection contour and its correction. As explained in Sect. 3.2, the proposed method splits the intersection point into two nearby points and preserves the original shape of the contour. Figure 6 shows the results from the intersection removal between two contours. In Fig. 6a, the inner contour is majorly inside the bigger contour compared to Fig. 6c where that same contour is majorly lying outside the bigger contour. As explained earlier, our algorithm clips the smaller contour differently in these scenarios. For case 1, the smaller contour will clip the outside portion compared to case 2 where an inner portion is clipped.

Table 1 Comparison of uniform sampling versus dynamic sampling

Text	Number of characters	Uniform sampling		Dynamic sampling	
		Number of vertices	Time (in ms)	Number of vertices	Time (in ms)
Mockingbird_English	185	58784	160.638	43924	108.134
Mockingbird_Chinese	79	69464	222.621	45138	125.843
Mockingbird_Hindi	202	76754	275.196	52214	153.484
Mockingbird_Spanish	192	64660	164.609	50724	125.329
Mockingbird_French	192	64608	169.222	48512	115.337
Mockingbird_Arabic	177	50464	130.776	49284	123.024

Table 2 Comparison of geometry from different tools

Method	No. of vertices	No. of faces
Blender [5]	4860	5816
Our	2130	4120

4.3 Mesh Quality

We compare our results with a professional editing tool Blender [5], which supports 3D text. As there is no direct metric to compare two meshes without a reference ground truth, we rely on quantitative metrics like the number of vertices to evaluate the geometry of two 3D meshes and qualitative comparison by comparing the two 3D mesh side-by-side against each other. To generate a 3D text from Blender [5], we use the following settings: 3 Resolution Preview, 0.03m Extrusion, 1 Bevel Level and 0.01m Bevel depth. This 3D mesh is then exported as an OBJ file and visualized in Meshlab [4]. For comparison, we use ITCEDSCR font type and “Hello” text as input to both the methods. Table 2 shows a comparison between the geometries of meshes. Our method uses less number of vertices compared to Blender [5]. This result efficacies dynamic sampling approach in our method.

Figure 9 compares mesh qualitatively from two methods. Figure 9a and b illustrates the shaded output from two methods. In Fig. 9c and d, we compare the zoomed-in region of the top-left curve from both the meshes. We observe that the mesh output from [5] is jagged as compared to ours. In Fig. 9e and f, we compare the Gauss curvature [19] from two meshes.

The 3D mesh generated from Blender and the proposed method are compared against each other in Fig. 7. Figure 7a shows that Blender supports dynamic sampling and chamfered edges. But Blender [5] repeats the vertices on side planes and chamfered plane as shown in Fig. 7b, whereas the proposed methods achieve similar quality by not repeating the vertices as shown in Fig. 7c and d. Thus, the quality

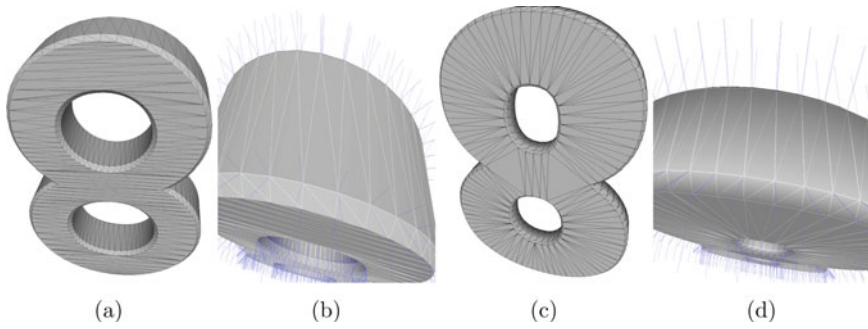


Fig. 7 Triangulation and vertex normal comparison **a** Blender [5], **b** Normal from Blender's output, **c** Our method and **d** Normal from our method

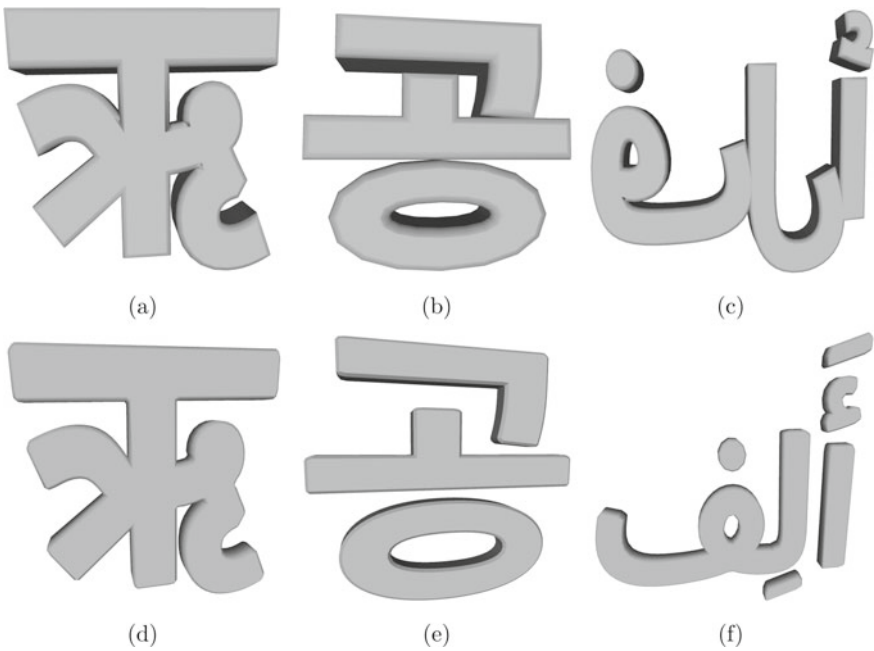


Fig. 8 Generated mesh for character **a** Hindi "Ri" [5], **b** "Korean Gong" [5], **c** "Arabic character alif" [5], **d** Hindi "Ri" (Our method), **e** "Korean Gong" (Our method) and **f** Arabic character "alif" (Our method)

Table 3 Comparison of character in Hindi, Korean and Arabic languages

		Blender [5]	Our method
Hindi “Ri”	No. of vertices	1180	708
	No. of faces	1412	1384
Koran “gong”	No. of vertices	530	960
	No. of faces	628	1900
Arabic “alif”	No. of vertices	2340	990
	No. of faces	2792	1936

of generated mesh from the proposed method is at par with professional tools like Blender [5].

Next, we compare characters from different languages in Table 3 and Fig. 8. We choose Hindi character “Ri”, Korean character “Gong” and Arabic character “alif”. We use “NotoSansDevanagari”, “NotoSansCJKkr” and “NotoSansArabi” font, respectively. To generate results from Blender [5], we use aforementioned settings. In Table 3, we observe that our method has less number of vertices and faces except for Korean character. But it’s very clear from Fig. 8b and e that our method’s mesh quality is superior to that of Blender [5]. Another observation is the final output for Arabic character that has come out wrong from Blender [5] as shown in Fig. 8c.

5 Conclusion

We present a novel pipeline for generating a 3D mesh for text which runs on an embedded system in real time. Our approach eliminates the need to create 3D fonts for different languages as it directly works on 2D contours from the text. Our intersection removal method eliminates the self-intersection in a contour with itself and intersections with neighbouring contours. The results prove high-quality mesh generation with rounded edges and mesh attributes like normals, tangents and texture coordinates. We have experimentally verified the generated 3D mesh with a popular professional tool to illustrate the efficacy of our approach.

With the growing popularity of AR, we expect a need to generate 3D mesh for hand-written text. We also foresee a demand to apply the real-world texture to the generated mesh in the AR world. Apart from this, we expect a need for a deformable mesh model to enable animation and increase the user’s interactions with the 3D mesh. These improvements will count as a step forward in the development of a coherent and advanced AR ecosystem.



Fig. 9 (Best viewed in colour) Qualitative comparison 3D mesh from **a** Blender [5], **b** Our method, **c** Zoomed-in region of mesh in **(a)**, **d** Zoomed-in region of mesh in **(b)**, **e** Blender output's Gauss curvature and **f** Our output's Gauss curvature

References

1. 7, AC, Texture mapping. <http://graphics.cs.cmu.edu/nsp/course/15-462/Spring04/slides/09-texture.pdf>
2. Andreev RD (1989) Algorithm for clipping arbitrary polygons. In: Computer graphics forum, vol 8. Wiley Online Library, pp 183–191
3. Attene M (2010) A lightweight approach to repairing digitized polygon meshes. *Vis Comput* 26(11):1393–1406
4. Cignoni P, Callieri M, Corsini M, Dellepiane M, Ganovelli F, Ranzuglia G (2008) Mesh-Lab: an open-source mesh processing tool. In: Scarano V, Chiara RD, Erra U (eds) Euro-

- graphics Italian chapter conference. The Eurographics Association. <https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136>
5. Foundation B, Blender. <https://www.blender.org/>
 6. Greiner G, Hormann K (1998) Efficient clipping of arbitrary polygons. *ACM Trans Graph (TOG)* 17(2):71–83
 7. Group K, Face culling. <https://www.khronos.org/opengl/wiki/FaceCulling>
 8. Inc A, ARKit. <https://developer.apple.com/augmented-reality/>
 9. Inc G, ARCore. <https://developers.google.com/ar>
 10. Inc G, Google translate. <https://translate.google.co.in/>
 11. Jin A, Fu Q, Deng Z (2020) Contour-based 3d modeling through joint embedding of shapes and contours. In: *Symposium on interactive 3D graphics and games*, pp 1–10
 12. Kettner L, Meyer A, Zomorodian A (2006) Intersecting sequences of dD iso-oriented boxes. *CGAL-32 User Ref Man* 5:1
 13. Lai YL, Wu JSS, Hung JP, Chen JH (2006) A simple method for invalid loops removal of planar offset curves. *Int J Adv Manuf Technol* 27(11–12):1153–1162
 14. Lee DT, Lin AK (1986) Generalized Delaunay triangulation for planar graphs. *Discret Comput Geom* 1(3):201–217
 15. Lin Z, Fu J, He Y, Gan W (2013) A robust 2d point-sequence curve offset algorithm with multiple islands for contour-parallel tool path. *Comput-Aided Des* 45(3):657–670
 16. Liu YK, Wang XQ, Bao SZ, Gomboši M, Žalik B (2007) An algorithm for polygon clipping, and for determining polygon intersections and unions. *Comput Geosci* 33(5):589–598
 17. Maillot PG (1992) A new, fast method for 2d polygon clipping: analysis and software implementation. *ACM Trans Graph (TOG)* 11(3):276–290
 18. Max N (1999) Weights for computing vertex normals from facet normals. *J Graph Tools* 4(2):1–6
 19. Meyer M, Desbrun M, Schröder P, Barr AH (2003) Discrete differential-geometry operators for triangulated 2-manifolds. In: *Visualization and mathematics III*. Springer, pp 35–57
 20. Rappoport A (1991) An efficient algorithm for line and polygon clipping. *Vis Comput* 7(1):19–28
 21. Skala V, Bui DH (2000) Two new algorithms for line clipping in E2 and their comparison
 22. Slabaugh G, Mihalef V, Unal G (2005) A contour-based approach to 3d text labeling on triangulated surfaces. In: *Fifth international conference on 3-D digital imaging and modeling (3DIM'05)*. IEEE, pp 416–423
 23. Vatti BR (1992) A generic solution to polygon clipping. *Commun ACM* 35(7):56–63
 24. Wang F, Kang L, Li Y (2015) Sketch-based 3d shape retrieval using convolutional neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 1875–1883
 25. Wong A, Eberle D, Kim T (2018) Clean cloth inputs: removing character self-intersections with volume simulation. In: *ACM SIGGRAPH 2018 talks*, pp 1–2
 26. Xie X, Xu K, Mitra NJ, Cohen-Or D, Gong W, Su Q, Chen B (2013) Sketch-to-design: context-based part assembly. In: *Computer graphics forum*, vol 32. Wiley Online Library, pp 233–245
 27. Zhong Y, Gryaditskaya Y, Zhang H, Song YZ (2020) Deep sketch-based modeling: tips and tricks. [arXiv:2011.06133](https://arxiv.org/abs/2011.06133)