Carlos Brito-Loeza
Arturo Espinosa-Romero
Anabel Martin-Gonzalez
Asad Safi (Eds.)

Communications in Computer and Information Science 1187

Intelligent Computing Systems

Third International Symposium, ISICS 2020
Sharjah, United Arab Emirates, March 18–19, 2020
Proceedings

Springer

Carlos Brito-Loeza · Arturo Espinosa-Romero ·
Anabel Martin-Gonzalez · Asad Safi (Eds.)

Intelligent Computing Systems

Third International Symposium, ISICS 2020
Sharjah, United Arab Emirates, March 18–19, 2020
Proceedings

Springer

Editors
Carlos Brito-Loeza
Autonomous University of Yucatán
Merida, Mexico

Arturo Espinosa-Romero
Autonomous University of Yucatán
Merida, Mexico

Anabel Martin-Gonzalez
Autonomous University of Yucatán
Mérida, Yucatán, Mexico

Asad Safi
Higher Colleges of Technology
Sharjah, United Arab Emirates

ISSN 1865-0929 ISSN 1865-0937 (electronic)
Communications in Computer and Information Science
ISBN 978-3-030-43363-5 ISBN 978-3-030-43364-2 (eBook)
https://doi.org/10.1007/978-3-030-43364-2

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Intelligent systems are computational systems with applications that can be seen in almost all disciplines, including biology sciences, computer vision, robotics, search engines, data mining, among others; capable to perceive, reason, learn, and act rationally by optimizing evaluation function. Several algorithmic methods could be used in the implementation of this type of system, with methods such as deep neural networks, Bayesian networks, kernel machines, feature extraction and dimension reduction, reinforcement learning, self-organizing maps, optimization methods in learning, fuzzy systems, and evolutionary computation, to mention a few. An increasing number of new algorithms and applications are developed each year.

With the increase in the number of personal computers, smart phones, and computer servers, the need for the development of intelligent systems has also experienced an important increment. Moreover, the current speed of computer processors allows for the implementation of algorithms that run on bigger databases.

This book contains the written contributions of the Third International Symposium on Intelligent Computing Systems (ISICS 2020) that was held in Sharjah, UAE, during March 18–19, 2020. ISICS 2020 was an international conference intended to attract computing professionals to discuss recent developments and trends in software and computing within their research communities. The aim was to further increase the body of knowledge in this specific area of computer science by providing a forum to exchange ideas and discuss state-of-the-art results. ISICS 2020 was committed to the promotion, preservation, and collaboration of research and practice, focusing on the fields of artificial intelligence, computer vision, and image processing.

For this edition of the conference, we received 46 papers from 12 different countries around the world. Each submission was evaluated by at least three members of the Program Committee. Based on these reviews, 13 papers made it to the final conference program as long oral presentation. In addition to the contributed papers, two keynote speaker presentations were included in the conference program.

We would like to warmly thank all the authors who submitted their work to ISICS 2020. Their contribution to the conference was greatly appreciated and we kindly invite them to continue to contribute to future ISICS conferences. We gratefully acknowledge the professional work of the International Scientific Program Committee, and we also greatly appreciate the contribution and participation of our invited speakers: Prof. Walterio Mayol-Cuevas (University of Bristol, UK) and Prof. Dr. Miltos Petridis (Middlesex University London, UK). We are very grateful to the Higher Colleges of Technology (HCT) of the UAE and the Universidad Autónoma de Yucatán (UADY) of Mexico, for their support in the organization of ISICS 2020.

March 2020

Carlos Brito-Loeza
Arturo Espinosa-Romero
Anabel Martin-Gonzalez
Asad Safi

Organization

General Chairs

Khaled Alhammadi	Higher Colleges of Technology, UAE
Nasser Nassiri	Higher Colleges of Technology, UAE
Samer Aoudi	Higher Colleges of Technology, UAE
Anabel Martin-Gonzalez	Universidad Autónoma de Yucatán, Mexico

Steering Committee

Bassam Ali	Universidad Autónoma de Yucatán, Mexico
Carlos Brito-Loeza	Universidad Autónoma de Yucatán, Mexico
Arturo Espinosa-Romero	Universidad Autónoma de Yucatán, Mexico
Nidiyare Hevia Montiel	Universidad Nacional Autónoma de México, Mexico
Ricardo Legarda-Saenz	Universidad Autónoma de Yucatán, Mexico
Anabel Martin-Gonzalez	Universidad Autónoma de Yucatán, Mexico
Asad Ali Safi	Higher Colleges of Technology, UAE
Israel Sánchez Domínguez	Universidad Nacional Autónoma de México, Mexico
Victor Uc-Cetina	Universidad Autónoma de Yucatán, Mexico
Kefaya Qaddoum	Higher Colleges of Technology, UAE

Scientific Advisory Committee

Asif Malik	Higher Colleges of Technology, UAE
Fuad Alhosban	Higher Colleges of Technology, UAE
Nicolaie Popescu Bodorin	Higher Colleges of Technology, UAE
Prithvi Bhattacharya	Higher Colleges of Technology, UAE
Daoud Daoud	Higher Colleges of Technology, UAE
Afaf Tabach	Higher Colleges of Technology, UAE
Thair Khdour	Higher Colleges of Technology, UAE
Mohammad Iqbal	Higher Colleges of Technology, UAE

Program Committee

Bassam Ali	Universidad Autónoma de Yucatán, Mexico
Noor Badshah	University of Engineering and Technology, Pakistan
Carlos Brito-Loeza	Universidad Autónoma de Yucatán, Mexico
Morgado Dias	Universidade da Madeira, Portugal
Arturo Espinosa-Romero	Universidad Autónoma de Yucatán, Mexico
Jorge Gomez-Montalvo	Universidad Autónoma de Yucatán, Mexico
Benjamín Gutierrez-Becker	Technische Universität München, Germany

Jean-Bernard Hayet	CIMAT A. C., Mexico
Francisco Javier Hernández López	CIMAT A. C., Mexico
Nidiyare Hevia-Montiel	IIMAS-UNAM, Mexico
Angel Kuri-Morales	ITAM, Mexico
Ricardo Legarda-Saenz	Universidad Autónoma de Yucatán, Mexico
Stacey Levine	Duquesne University, USA
Eduardo Lleida	Universidad de Zaragoza, Spain
Elena Loli Piccolomini	University of Bologna, Italy
José Luis López Martínez	Universidad Autónoma de Yucatán, Mexico
Salvador Mandujano	Google Inc., USA
Anabel Martin-Gonzalez	Universidad Autónoma de Yucatán, Mexico
Rolando Medellín	University of Dundee, UK
Raúl Monroy Borja	ITESM-CEM, Mexico
Francisco Moo-Mena	Universidad Autónoma de Yucatán, Mexico
Nicolás Navarro-Guerrero	Aarhus University, Denmark
Luis A. Pineda Cortés	IIMAS-UNAM, Mexico
Lavdie Rada	Bahçesehir University, Turkey
Alonso Ramirez-Manzanares	CIMAT A.C., Mexico
Eduardo Rodriguez-Martinez	Universidad Autónoma Metropolitana, Mexico
Israel Sánchez-Domínguez	IIMAS-UNAM, Mexico
Asad Safi	Higher Colleges of Technology, UAE
Sai Deep Tetali	Google Inc., USA
Victor Uc-Cetina	Universidad Autónoma de Yucatán, Mexico
Flavio Javier Vigueras Gómez	Universidad Autónoma de San Luis Potosí, Mexico
Fernando Von Borstel Luna	Centro de Investigaciones Biológicas del Noroeste, Mexico

Contents

An Investigation on Performance of Attention Deep Neural Networks in Rapid Object Recognition

Zahra Sadeghi(✉)

Department of Cognitive Linguistic and Psychological Sciences,
Brown University, Providence, USA
zahsade@gmail.com

Abstract. Inspite of the huge success of Deep Neural Networks in object recognition, there are still situations in which they cannot reach human performance. In this work, the performance of an attention Deep Neural Network which is cued by important pixel of objects is addressed. First, the effect of color on accuracy of classification is evaluated. Then the network performance is compared with humans by using a set of images from different levels of revelation of important pixels. The results indicate that color information enhances the recognition of objects and there is a correspondence in accuracy of classification as well as correlation in decisions between human and attention networks at middle and low levels of important pixel revelation respectively.

Keywords: Object recognition · Attention · Deep neural networks

1 Introduction

Despite of the advancement in deep neural network architectures and their success in achieving high efficiency in visual object recognition, they are not yet analogous to human brain in terms of utilized features, computational methods [14,15] and especially output performance. There are still many situations in which machine performance is worse than human. One approach to Deep Convolutional Networks (DCNs) understanding is via studying the similarities between human and DCNs in object recognition tasks. Learning about the situations in which DCNs perform poorly can help machine learning and computer vision researchers to further improve deep networks architecture. Recently, research in this area has received attention and the comparison between human and model performance has been addressed in different scenarios [5,7]. One long-standing question in artificial intelligence is about computational models of attention mechanism in primate visual cortex [2,9]. Human benefits both from local and global information for object recognition [1,6,13]. Moreover, while salient parts of images can capture bottom-up attention features through computer vision algorithms, they are not fully consistent with human strategies. Recently, a large attention dataset is created which contains regions of images (namely, importance or attention maps) that human observers attend to (with response time under 550 ms) in a rapid object recognition task. These importance maps are compared to saliency maps by the aid of a psychophysical experiment and it has shown that they contain features that are more helpful for human in the task of rapid object recognition compared to those features that are produced by saliency maps [10]). In addition, a version of excitation and squeeze network is proposed under the name of GALA network which combines two sources of global and local information in one architecture. This attention network has been fed with the importance maps and it was shown that supervisory attention information can enhance the performance of the network. In this work, first, the performance of GALA network on color and grayscale images is compared. Then the performance of the networks in a binary animal/non-animal classification problem is evaluated and the parallelism between human and model in terms of accuracy of recognition and decision scores is studied.

2 GALA Networks

Global-and-Local-attention (GALA) network was first introduced by Linsely et al. [10]. This network extends the squeeze-and-excitation (SE) network [8] by adding a local saliency module. This local module is then combined with global attention module (i.e., SE module) which contains the contextual information of an image. The GALA strategy is incorporated in ResNet-50 deep neural network. In order to add human top-down attention in the loop of recognition process, an attention module has been added to the network. The attention mechanism is embedded in the cost function as a regularization term which is defined in the second term of Eq. 1. The first term in the cost function (L_c) denotes the cross entropy loss value of a model M on inputs with labels y. The second term provides the supervisory signals towards importance maps ($R^l(x)$). This term incorporates the degree of freedom for controlling attention over important regions of image according to the information derived from a psychophysical experiment which is described in the next section. $A^l(x)$ is GALA module activity.

$$L_T(X, y) = L_c(M(x), y) + \lambda \sum \left\| \frac{R^l(x)}{R^l(x)} - \frac{A^l(x)}{A^l(x)} \right\|_2 \tag{1}$$

3 Clickme.ai Experiment

Clickme.ai is a web-based game developed by Linsely et al. [11] with the prospect of collecting the attention maps from the perspective of human visual attention. Each participant in this game is instructed to click on the region of image which could be most informative to the co-player for rapid recognition of objects in the image. Each click paints a region of 14 by 14 area of pixels. This information is labeled as human feature importance map for objects. During viewing each image, the category of image is shown above it. Each image is viewable up until a maximum time limit and ends if the selected regions consist sufficient information for DCN to recognize it correctly. Participants are awarded based on the additional amount of time remaining on timer after DCN successfully recognizes the object in its top-5 prediction results.

4 Behavioral Experiment

In order to test the effect of importance maps collected in clickme.ai experiment, a rapid object recognition experiment was designed [11]. The dataset contains 100 images from two superordinate category of animal and non-animal which are organized in 10 subcategories. Each subcategory contains 10 sample images which are gathered from ILSVRC2012 challenge. Phase scrambled masks are applied to the images in order to cover each image according to the features of importance maps derived from clickme.ai experiment. This ended in eleven versions of each image ordered ascendingly based on their level of pixel revelation of important pixels. The first revelation level is associated to 1% of important pixels of images (in a log-scale space) and the last one includes images in full presentation. A sample image from each level of revelation is shown in Fig. 1.

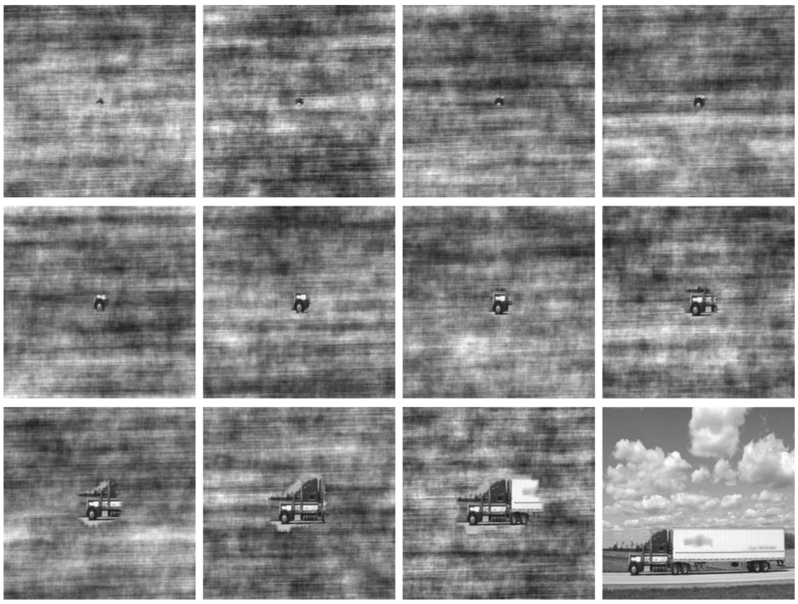


Fig. 1. Example of images at each level of pixel revelation.

5 Method

Three main models, namely, GALA network with clickme attention (gala_click), GALA network without clickme attention (gala_no_click) and basic ResNet models are trained on gray scale images of train set of clickme experiment. The Clickme train set contains 329,036 images which are collected from ILSVRC2012 database and hence all the networks are trained on 1000 classes of ImageNet data. In order to study the influence of lambda value (in the objective function of Eq. 1) on network performance, six extra sub-models of GALA network with clickme attention is trained using Clickme train images. The deep feature vectors are created using the feature representation at last layer of convolutional layer (i.e., before logistic layer) from each of the trained deep neural network models. These feature vectors are then used for training linear classifiers on an animal/non-animal task. In this paper, linear SVM classifiers are used which are widely employed in rapid object recognition problems [4,12]. The c parameter is optimized through cross validation via grid search. For training the SVM classifiers on animal/non-animal task, 80,000 images are utilized from ImageNet dataset. The images are selected from diverse classes and are balanced in terms of the number of images in each class. More details about this image set can be found in [5]. The SVM classifiers are then tested on the image sets that were used in the behavioral experiment (which was explained in the previous section). The accuracy of these classifiers is finally compared with that of humans as obtained from behavioral experiment [10]. Human performance is calculated in two ways: 1 - accuracy of each participant in recognizing all images (i.e., human accuracy), 2 - accuracy of recognition of each image (i.e., human decision). The model performance is evaluated in terms of the average value of these two measurements over all samples. In order to measure model decision scores, the distance to hyperplane decision boundary is considered which is learned by SVM classifiers. The absolute value of distance is considered for the correctly classified images, and its negative is used for misclassified ones.

6 Results

6.1 Accuracy of Recognition

In this section, performance of human and model in the task of rapid object recognition is presented and a comparison is made between them. All the models are trained using Clickme train images and the top-1 and top-5 accuracies are evaluated over Clickme test images. The images of this dataset are a subset of ILSVRC2012 and are chosen from all 1000 categories. Figure 2 shows the evaluation results on clickme test data. The best results are attributed to `gala_click` with coefficient (lambda) value of 6. Next the impact of color information on object recognition is investigated. To this end, three cases are considered: 1 - networks trained on color images and tested on color images. 2 - networks trained on grayscale image and tested on grayscale images. 3 - networks trained on color images and tested on grayscale image. The overall accuracies on test

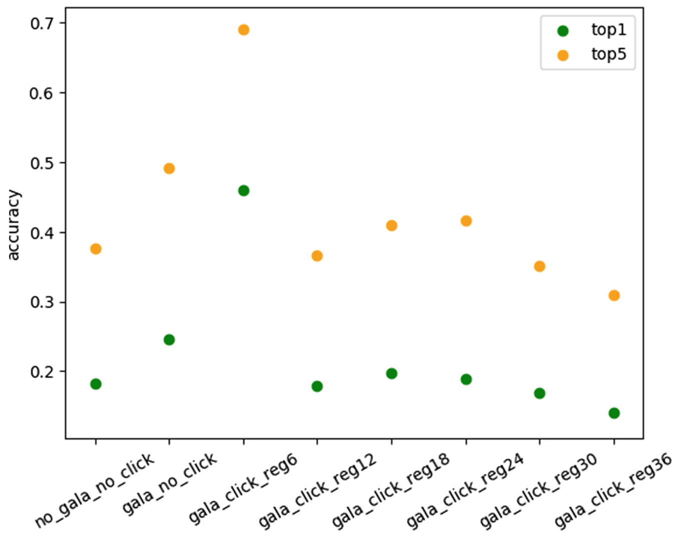


Fig. 2. Accuracy of GALA models on test set of clickme images.

clickme sets in terms of top-1 and top-5 accuracies are presented in Fig. 3. As can be inferred from the results, the best performance in both color and grayscale cases is achieved by `gala_click`, while `gala_no_click` and `no_gala_no_click` obtained second and third best results respectively. In addition, the highest accuracy for all models is attributed to the case in which images are trained on colorful images and tested on colorful images. Average accuracy of the two gala models (i.e., `gala_click` and `gala_no_click`) and ResNet-50 model (i.e., `no_gala_no_click`) is then compared on the behavioral test images at different levels of pixel revelation. Comparison of the accuracy of all three models is shown in Fig. 4a. For the sake of comprehension, polynomial curves are fitted to the performance of two models. As can be understood from Fig. 4a, `gala_click` and `gala_no_click` models achieved similar accuracy. However, `gala_click` model produced superior results compared to all other models in full pixel revelation. The second best performance in full level, is achieved by `gala_no_click` model. In order to analyze the results more specifically, the level of revelation is divided into three stages. Stages 1, 2 and 3 contain images from level 0 to 5, 6 to 8, and 9 to 11 subsequently. Levels 0 and 11 correspond to minimum and maximum revelations respectively. The average performance of each stage in network and human are listed in Table 1 and are plotted in Fig. 4b. As can be observed, in the first stage, no model performs better than human. This indicates the superior performance of human in object recognition when minimum information of objects are revealed. In the second stage, there is no significant difference between the average accuracies. In this stage, both human and models perform similarly. Finally, in the third stage, human accuracy exhibits the lowest result in comparison to the other models. This reflects the dominant power of deep networks

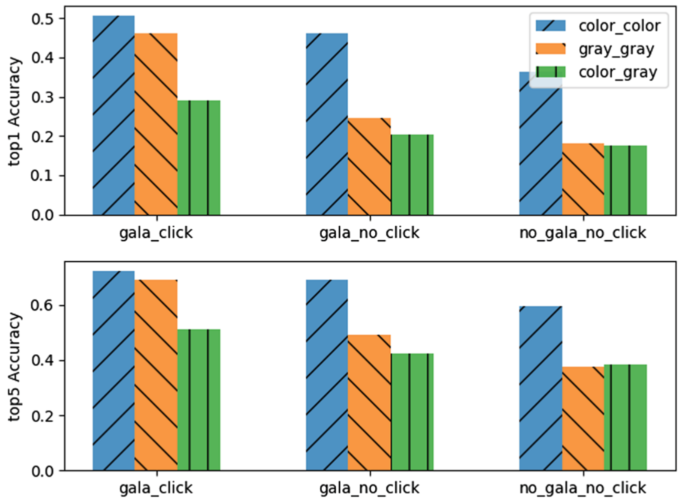


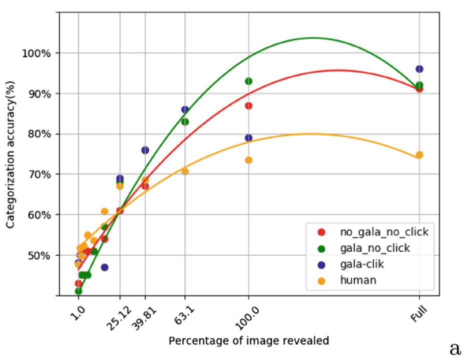
Fig. 3. Effect of color on accuracy of recognition.

in rapid object recognition with highest level of pixel revelation in presence of contextual information which could be explained due to high memory capacity and huge processing capabilities of deep networks.

Table 1. Comparison between accuracy of human and models in different stages of pixel revelation.

	Stage 1	Stage 2	Stage 3
Human	0.52	0.65	0.73
Resnet	0.36	0.61	0.87
Gala	0.41	0.67	0.89
Gala with click	0.48	0.64	0.87

Next, attention coefficient values explored in order to study the performance of a family of `gala_click` models. Attention coefficient (or regularization coefficient) which is denoted by lambda term in Eq. 1 provides a degree of freedom for gala network and makes it suitable for implementing different levels of sensitivity to importance maps. The results which are shown in Fig. 5 suggest that using coefficient 36 obtains most human-level performance at different levels of pixel revelation.



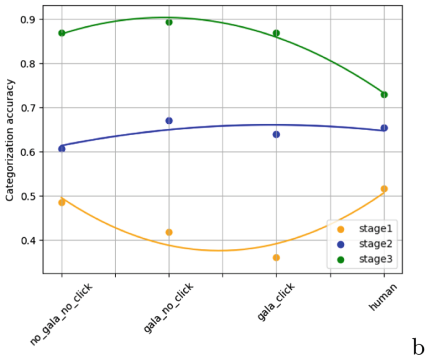


Fig. 4. Comparison between accuracy of human and models in different levels and stages of pixel revelation.

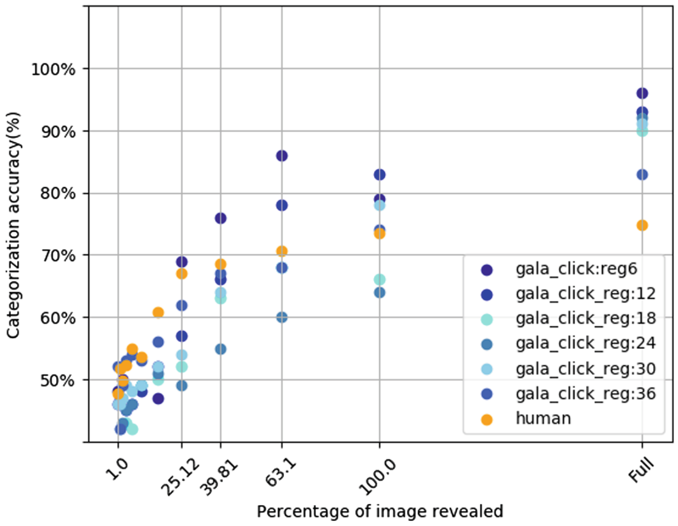


Fig. 5. Accuracy of a family of GALA models obtained by using different attention coefficients (shown as reg values)

6.2 Correlation Between Human and Model Decision

In this section, the correspondence of decision scores across models and human in the task of rapid object recognition is investigated. For this, per-image performance is assessed. Using the spearman's correlation, the degree of decision agreement between human and model in rapid object recognition is evaluated. Spearman's correlation is computed between the confidence of SVM classifier output for each image (i.e., the distance from the hyper-plane aka model decision) and average correct scores of all participants for each image at each level of pixel revelation. To this end, the behavioral data provided by [5] is utilized.

The correlation between attention models and that of humans is computed. The highest correlation is achieved by applying features from the last convolutional layers of `gala_click` with regularization coefficient of 30. The results for the three main models are shown in Fig. 6. The results suggest that both `gala_no_click` and `gala_click` acquire top correlations when the lowest amount of importance maps is presented. In the middle levels of pixel revelations, basic resnet (`no_gala_no_click`) model acts more closely to human decisions. The decision correlations for a family of `gala_click` networks are plotted in Fig. 7. The figure shows that Gala_click model has a higher correlation to human decision pattern in lower levels of pixel revelation. In other words, in situations, when minimum important information of objects is available, `gala_click` takes similar strategy to humans.

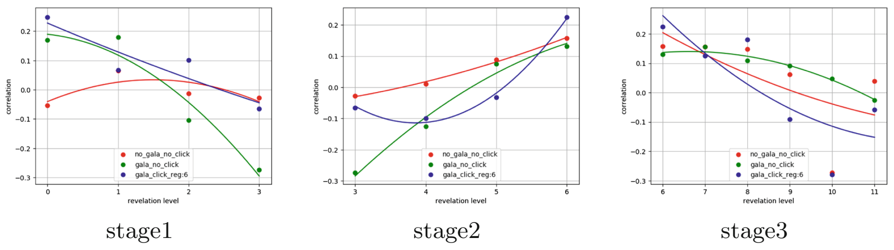


Fig. 6. Correlation between human and model decision scores in different stage of revelation.

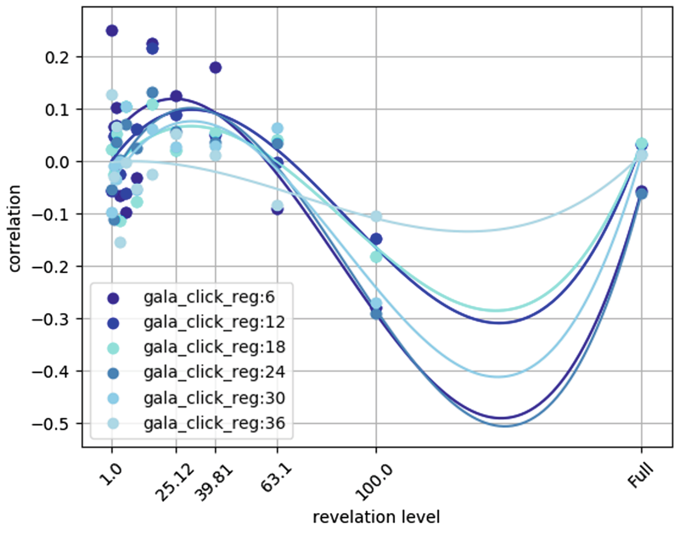


Fig. 7. Decision correlation between human and model in a family of GALA networks.

7 Conclusion

A number of studies have put forward that human recognition mechanism is benefited from both overt and covert shift of attention. In essences, object recognition in human brain is not only based on selective processing of a target object, but it also utilizes contextual information of scene. The key aspect of GALA Networks is fusing the salient features of images with their contextual information. Most of the developed methods for measuring the local attention are based on localization of the salient regions based on the intrinsic features of images such as illumination, color and orientation [3]. However, it is not confirmed that human visual system follows the same strategy for saliency detection. Clickme.ai experiment was designed with the aim of creating importance maps (or Clickme maps) by collecting natural salient information from human. The importance maps are embedded as a supervisory signal in GALA architecture and has shown to be effective in enhancing the accuracy of ResNet network for object recognition. In this paper, the line of work is concerned with the performance of different models of GALA networks on grayscale and color images as well as the parallels between human and models which are cued by the importance maps. A fine-tuning strategy is deployed via training SVM classifiers in order to reuse the deep features extracted from GALA networks on a binary task of animal/non-animal classification. The SVM classifiers performance is compared with human participants on images which are covered with masks in an increasing order of pixel revelation. The results demonstrate the higher accuracy of GALA models with Clickme attention and the enhancing effect of color in animal/non-animal recognition task. Moreover, the comparison of model and human indicates more similarity in accuracy in middle levels of pixel revelation and a higher agreement between human and model decisions in lower levels of pixel revelation.

Acknowledgment. This research has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 665919. Thanks to Thomas Serre and Drew Linsley.

References

1. Bar, M., et al.: Top-down facilitation of visual recognition. Proc. Nat. Acad. Sci. **103**(2), 449–454 (2006)
2. Borji, A., Cheng, M.M., Jiang, H., Li, J.: Salient object detection: a benchmark. IEEE Trans. Image Process. **24**(12), 5706–5722 (2015)
3. Borji, A., Itti, L.: State-of-the-art in visual attention modeling. IEEE Trans. Pattern Anal. Mach. Intell. **35**(1), 185–207 (2013)
4. Cadieu, C.F., et al.: Deep neural networks rival the representation of primate IT cortex for core visual object recognition. PLoS Comput. Biol. **10**(12), e1003963 (2014)
5. Eberhardt, S., Cader, J.G., Serre, T.: How deep is the feature analysis underlying rapid visual categorization? In: Advances in Neural Information Processing Systems, pp. 1100–1108 (2016)

6. Egeth, H.E., Yantis, S.: Visual attention: control, representation, and time course. Annu. Rev. Psychol. **48**(1), 269–297 (1997)
7. Geirhos, R., Janssen, D.H., Schütt, H.H., Rauber, J., Bethge, M., Wichmann, F.A.: Comparing deep neural networks against humans: object recognition when the signal gets weaker. arXiv preprint arXiv:1706.06969 (2017)
8. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
9. Itti, L., Koch, C.: A saliency-based search mechanism for overt and covert shifts of visual attention. Vision. Res. **40**(10–12), 1489–1506 (2000)
10. Linsley, D., Scheibler, D., Eberhardt, S., Serre, T.: Global-and-local attention networks for visual recognition. arXiv preprint arXiv:1805.08819 (2018)
11. Linsley, D., Shiebler, D., Eberhardt, S., Karagounis, A., Serre, T.: Large-scale identification of the visual features used for object recognition with ClickMe.ai. J. Vision **18**(10), 414–414 (2018)
12. Serre, T., Oliva, A., Poggio, T.: A feedforward architecture accounts for rapid categorization. Proc. Nat. Acad. Sci. **104**(15), 6424–6429 (2007)
13. Torralba, A., Oliva, A., Castelhano, M.S., Henderson, J.M.: Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. Psychol. Rev. **113**(4), 766 (2006)
14. Ullman, S., Assif, L., Fetaya, E., Harari, D.: Atoms of recognition in human and computer vision. Proc. Nat. Acad. Sci. **113**(10), 2744–2749 (2016)
15. Waldrop, M.M.: News feature: what are the limits of deep learning? Proc. Nat. Acad. Sci. **116**(4), 1074–1077 (2019)

Breast Cancer Detection and Localization Using MobileNet Based Transfer Learning for Mammograms

Wajeeha Ansar, Ahmad Raza Shahid, Basit Raza(✉), and Amir Hanif Dar

Medical Imaging and Diagnostic (MID) Lab, National Centre of Artificial Intelligence (NCAI), Department of Computer Science, COMSATS University Islamabad (CUI), Islamabad 45550, Pakistan

Wajeeha.ansar@yahoo.com, {ahmadrshahid,basit.raza, daramir}@comsats.edu.pk

Abstract. Breast cancer is the major cause of death among women. The best and most efficient approach for controlling cancer progression is early detection and diagnosis. As opposed to biopsy, mammography helps in early detection of cancer and hence saves lives. Mass classification in mammograms remains a major challenge and plays a vital role in helping radiologists in accurate diagnosis. In this work, we propose a MobileNet based architecture for early breast cancer detection and further classify mass into malignant and benign. It requires less memory space and provides faster computations with 86.8% and 74.5% accurate results for DDSM and CBIS-DDSM, respectively. We have achieved better results than other deep CNN models such as AlexNet, VGG16, GoogleNet, and ResNet.

Keywords: Breast cancer · Mammography · Deep learning · MobileNet · Localization

1 Introduction

The second most prevalent cause of death from cancer among women is breast cancer. Based on the recent statistics from the American Cancer Society, it is estimated that 41,760 women in the United States are expected to die from breast cancer in 2019 [1]. Mammography is one of the most commonly used techniques of breast cancer screening which has made a significant contribution to reducing mortality rates through early cancer detection. Mammography involves exposing the breasts of a patient to low levels of X-ray radiation. The complexity of mammograms and the high volume of examinations per radiologist, however, can lead to a fake diagnosis [2].

Computer-Aided Detection (CAD) systems are used to help radiologists for cancer detection and diagnosis. Studies have demonstrated the effectiveness of CAD systems; however, accurate breast cancer detection remains a challenging task. Due to mammograms complexity and huge number of examinations by radiologist, results in false diagnosis. CAD system aims to decrease false diagnosis, which ultimately reduce unnecessary biopsies, patient anxiety, additional cost of health care and extra evaluation by radiologists [3]. Current CAD constraints show the need for fresh and more accurate techniques of detection. Standard Machine Learning techniques (ML) used for CAD systems, include a hand-crafted feature extraction phase, which is challenging as it requires domain knowledge of expert radiologists which is costly and time consuming.

Moving on from classical ML, more recent effort has been geared towards use of Deep Learning (DL) techniques They diagnose suspected lesions more accurately through quantitative analysis [4]. DL models are designed for large and diverse datasets (e.g. imageNet dataset), whereas for mammograms we have small datasets publicly available. Consequently, the capability and complexity of such networks can lead to significant adverse effects on model performance when learning from scratch with small training samples.

To overcome afore mentioned issues, Transfer Learning (TL) techniques are widely used for mammograms. In TL we train deep models on large dataset and then test them on smaller dataset with fine tuning. In this work, we present our preliminary outcomes on the use of transfer learning to classify breast cancer. We use a MobileNet [5] based technique, which is efficient, require less memory and computational cost. This paper comprises of the following:

- We proposed a customized DL architecture that is better suited to mobile devices because of its compact size, reduced computational costs and competitive performance relative to existing models.
- We also perform methods such as pre-processing of images, data augmentation, choice of hyper parameters (e.g. batch size, learning rate), which are attractive alternatives for reducing overfitting and increasing the generalization of the TL technique for mammograms.
- Comparative assessment of DL models such as VGG [6], AlexNet [4], ResNet [7], GoogleNet [8] with MobileNet and MobileNet-v2 [9] in terms of their efficiency and computational power.
- Our aim is to provide automatic CAD system to manage millions of routine imaging examinations, exposing prospective cancers to the radiologists who undertake follow-up operations. This can be used to help and serve as a second eye for radiologists.

The rest of the paper is organized as follows: literature review in Sect. 2 on the use of transfer learning in the classification of defects mammograms as benign or malignant in the current study. Section 3 introduces our proposed experimental technique and dataset. Section 4 provides the outcomes of experimental information. Finally, conclusions and future work are drawn in Sect. 5.

2 Previous Work

Current CNN models are intended to enhance the capacity of radiologists to locate even the smallest breast cancers at their earliest phases. Due to the accessibility of large information repositories, the power of parallel and distributed computing, use of DL techniques has resulted in breakthroughs in pattern recognition and classification tasks.

It is very difficult to train a deep CNN model with a small amount of medical information. Unfortunately, publicly available datasets for mammograms are not large enough for deep learning models. In recent years, several models of convolutional neural networks such as: AlexNet, VGG, GoogLeNet and Resnet, and others have been created to resolve such issues. Researchers have also begun to investigate the use of these models for transfer learning to classify mammograms [10]. For transfer learning, fine tuning is the most commonly used technique. In this case, with the new data, only a few of the model's last layers are retrained.

AlexNet was the first convolutional neural network (CNN) in the field of object detection and classification to exhibit performance beyond the state of the art. Jiang et al. [11] and Huynh et al. [12] used AlexNet model for applying TL technique on mammograms. Both authors used pre-trained AlexNet for the issue of mass diagnosis without further fine-tuning. Using SVM for classification, they evaluated classification performance using characteristics from multiple intermediate network layers. They contrasted their outcomes with two methods: a classifier working on hand-crafted characteristics and a soft voting ensemble of both. Rampun et al. [13] used a mildly altered, pre-trained and fine-tuned version of AlexNet on CBIS-DDSM. They selected the three highest performing models during inference and merged their predictions.

The impact of network depth was explored in VGG while maintaining a smaller size for convolution filters. VGG-19 is an updated version of VGG-16, and they showed that by increasing the depth from 16 to 19 layers, important improvement can be accomplished. The benefit of VGG is that the efficient receptive field of the network is enhanced by stacking various convolution layers with small kernels, while decreasing the number of parameters compared to using less convolutional layers for the same receptive field with bigger kernels. Hamidinekoo et al. [14] used VGG16 and GoogleNet. Similarly, Chougrad et al. [15], for instance, used VGG16, InceptionV3, and ResNet. They showed that the precision of the fine-tuned model decreases when the number of convolutional blocks exceeds 2.

Residual Networks (ResNet) consist of reformulated convolutional layers that learn residual features based on inputs. It is simpler to optimize this sort of networks by considerable reduction in depth such as the "residual block" implementation. Google has created the InceptionV3 model which is also known as GoogleNet. It's computational cost and memory requirements are much smaller than that of VGG and ResNet, making it a popular model being used for big data.

Morrell et al. [16] used InceptionV3 and deformable convolutional net for private mammographic dataset. This method is regarded to be computationally complicated, as TL is done with two different models. Cameiro et al. [17] worked on the whole mammogram image models in quest for an end-to-end design. Carneiro et al. [18] used a pre-trained CNN that was fine-tuned using unregistered mammograms and microcalcification and mass segmentation to estimate the risk of BIRADS-based breast cancer. They found that the pre-trained models are superior to those initialized at random.

Rather than training from scratch, more recent work has focused on use of pre-trained networks. However, pre-trained CNN architectures are designed, trained and tested on large datasets, which are diverse and different in nature than the available mammographic datasets. Consequently, the capability of such networks and their complexity can far exceed the requirements of larger datasets, resulting in significant negative effects when training from scratch and limit their use in mobile devices. Using such large networks on mobile devices is hard on memory and computational power which in turn decreases the performance. To resolve this issue, we propose a transfer learning-based technique using MobileNet based model with few parameters. As a result, they require less memory space and provide good and faster result as compared to other large VGG and inception models.

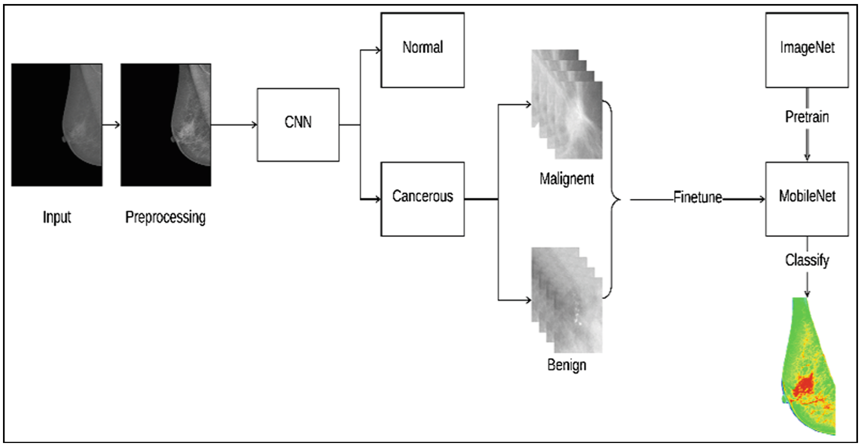


Fig. 1. The proposed framework for automated detection and localization of masses in full mammograms, where the first shallow CNN take pre-process full mammogram and performed binary classification between cancerous and non-cancerous. If cancer present in input then is further pass to MobileNet based model, which is pre-trained on ImageNet dataset. To reduce parameter, we used ROI for fine-tuning. At the end to localize cancer in full mammogram, we used class activation mapping technique.

3 Methodology

Mammograms cannot be considered as a simple image classification task, because there are defects in tiny areas within an entire image. For instance, a typical complete mammogram with 3000 $\times$ 4600 resolution (width and height in pixels) includes only about 200 $\times$ 200 (pixels) of abnormality. To use a pre-trained deep model for TL, it is necessary to resize and normalize the input to the same format that the network was initially trained on (e.g. 224 $\times$ 224 for VGG models). Therefore, to address this challenge, we are proposing to train extensive CNN framework with fewer parameters on cropped image patches (labeled ROIs) and adapt them to complete mammogram images. Figure 1 shows our approach's data flow. A binary shallow CNN classifier is trained as shown in Fig. 2, followed by a MobileNet based architecture with training image patches from benign and malignant mass tissues.

A pre-trained MobileNet is altered to have two output classes in output layers. The output layers are then finely tuned while the network's initial layers are kept frozen. The performance of the model can be tracked by visualizing the learned features at different layers of MobileNet during the training process. In this manner, characteristics corresponding to distinct scales were acquired. In addition, it helps us to stop early training and reduce the size of learning parameters. The fine-tuned neural patch network is then used to locate mammographic defects in mammograms of full size by generating heat-map at the end of output layer.

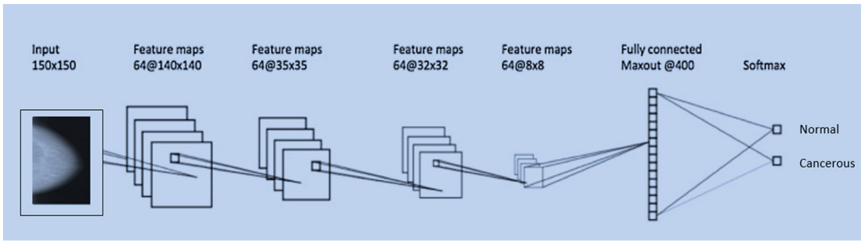


Fig. 2. CNN architecture for binary classification of mammogram into Normal or Cancerous.

3.1 Data Set Used for Experiments

There is a shortage of datasets for standard assessment in mammography and are not publicly accessible, therefore most CAD algorithms are assessed on private datasets. It presents a challenge in comparing method efficiency or replicating previous results. The databases most frequently used are Digital Database for Screening Mammography (DDSM) [18] and Curated Breast Imaging Subset of DDSM (CBIS-DDSM) [19].

DDSM is the biggest publicly accessible mammography dataset. The database involves roughly 2,500 trials, each of which includes two views for each breast Cranial Caudal (CC) view captured from the top of breast and other is Medium Lateral Oblique (MLO) captured from side angle at 45° as shown in Fig. 3. Text data about the patient and image is also available in dataset. Images comprising suspect areas connected “ground truth” pixel-level with the places and kinds of suspect regions. To test and train our automatic segmentation techniques, we only use images taken from the CC perspective. Both the characteristics of MLO and CC are used to test and train the model of mass lesion diagnosis.

An updated and standardized DDSM version for mammography is CBIS-DDSM. The dataset includes 753 cases of calcification and 891 cases of mass. We use CBIS-DDSM patch images to classify and test for localization in complete mammograms. We combine the two discussed datasets for training and testing and perform 80/20 split for training/testing sets respectively. In CBIS-DDSM dataset, experts cropped the ROIs of abnormality portion and also removed 339 questionable mass cases manually. Breast mass classification into malignant and benign is a challenging task due to the diverse characteristics of abnormal regions and overlapping with the dense breast tissues. Mass samples from CBIS_DDSM are shown in Fig. 4.

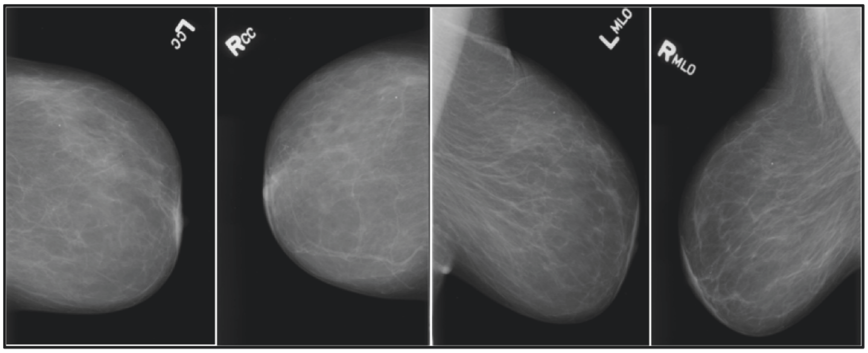


Fig. 3. CC and MLO views of a patient from DDSM dataset (From left to right: left CC view, right CC view, left MLO view, right MLO view).

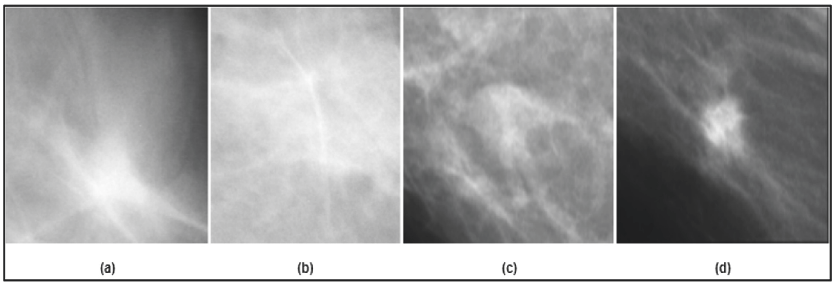


Fig. 4. Example of Mass benign cases (a) and (b); Mass malignant cases (c) and (d) from CBIS-DDSM dataset.

3.2 Image Pre-processing and Data Augmentation

Before training CNNs, mammogram's pre-processing is an important task. Mammograms images contain artifacts, labels, pectoral muscles and have low contrast, which affects training and hence reduces proper diagnosis. To remove noise and enhance mammograms contrast, we applied adaptive mean filter Contrast Limited Adaptive Histogram Equalization (CLAHE) and median filter [20, 21].

Since CBIS-DDSM's ROI images are of distinct dimensions, a shift in image size was essential. Considering the aspect ratio as defined in r$\frac{w}{h}$ where h, w and r are the image's height, width and aspect ratio respectively. From dataset we removed images with an aspect ratio of less than 0.4 and greater than 1.5. We considered aspect ratio to resize the image as a parameter, which helps in both up-sampling and down-sampling procedures to preserve the best quality possible from the original images. Most pre-trained techniques use 224 $\times$ 224 (width, height) target sizes. Cubic interpolation was used for up-sampling, while area interpolation yields highest outcomes for down-sampling.

Due to relatively smaller size of the available dataset, CNN models memorized all the dataset rather than learning it, hence the model over fit the data and decreased its performance on future unseen data. To avoid over fitting, we have performed data augmentation, which generated more instances for training artificially by applying transformations to the actual data such as flipping and rotation. We flipped and rotated patches by 90, 180, and 270°, sharing by 0.2, zooming and random scaled. Such data increase appropriate training samples because tumors can occur in different directions and sizes. Thus, methods of augmentation do not alter the masses fundamental pathology.

4 Experiments

In this section we discuss evaluation metrics, experimental setting and results in detail.

4.1 Evaluation Metrics

In this paper, we have used traditional approaches to assess our model performance such as accuracy, recall and precision as indicated in Eqs. 1, 2 and 3. Where *P* and *N* are the total of positive and negative class instances respectively. In our first simple classifier *P* is cancerous and *N* is non-cancerous, while in second MobileNet model *P* is Mass malignant and *N* is mass benign. TP, TN, FP and FN stand for True Positive, True Negative, False Positive and False Negative respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

4.2 Experimental Setup

Hyper-parameters are variables that determine the structure of the network (e.g. number of hidden layers) and variables that determine how the network is trained (e.g. number of epochs, rate of learning, batch size, decay etc.). Before training the CNNs, hyper-parameters are selected manually.

Learning Rate. Learning rate (LR) is one of the most significant hyper-parameters that affects the efficiency of the CNNs. Typically, stochastic gradient descent optimizer is used in deep learning models. Variants of stochastic gradient descent are RMS Prop, Adam, Adagrad, etc. All these optimizers allow the learning pace to be set by customers. To control and minimize losses of a model, all these optimizers use learning rate. Many iterations are required as learning rate is too small. However, if learning rate is too high, it can cause undesirable divergent behavior and destroy pre-trained weights of model. The popular strategies on learning rates are step decline, quadratic decline, square root decline, and linear decline [19]. For mammograms and patches we have used Adam with decay rate of 0.99 per epoch, and a regularization coefficient of 10^{-5} for training their CNN.

Batch Normalization. A batch normalization layer standardizes input variables over a mini-batch (a subset of the training data set) in a CNN model. It reduces the network initialization sensitivity and speeds up the CNN training process. We chose the largest batch size our GPU memory would allow for batch size. This is a batch size of 64, although we had more computing energy. A greater batch size implies that the information set is better represented. Which implies that our model will converge more quickly and will have less variety when the model tries to discover the optimum minimum in our gradient.

Model Check Point. A model checkpoint was a significant technique we used. This saves the weights of the model for every epoch with the best score. We can use these weights that during practice were saved at a midpoint. This enables us to guarantee that the weights of the model we use do not fit too well on the training data and generalize well into fresh information. Maintaining the model weights with the highest outcomes is also significant. Only saving best weights saves our memory.

4.3 Transfer Learning Model Comparison

Architectures like VGG and ResNet are either big in size or involve a lot of mathematical computations, although they have attained very high accuracies in the Imagenet dataset. Thus, these architectures may not be effective for embedded based vision applications and mobile devices. Deep learning-based architecture for mobile devices that is computationally effective, tiny in size and achieve higher accuracy is MobileNet, proposed by Google in 2017.

MobileNet's primary concept is that instead of using the standard 3 * 3 convolution filters, the procedure is divided into 3 * 3 depth-wise separable convolution filters, followed by 1 * 1 point-wise convolutions. The new architecture needs fewer operations and parameters while achieving the same filtering and mixture method as a regular convolutional architecture.

Let input image size is $H * W$ with Nin number of channels (e.g. 3 for RGB image), filter size $F * F$ and Number of output channels is $Nout$, then number of operations can be computed using Eq. 4 as follow (here we use F = 3):

$$H * W * Nin * F * F * Nout \tag{4}$$

$$H * W * Nin * (9 * Nout) \tag{5}$$

While in the case of depth-wise separable convolution with filters of size $F * F$ followed by point-wise 1 * 1 convolution, number of operations is given in Eq. 6 (here we use F = 3):

$$(H * W * Nin * F * F) + (H * W * Nin * Nout) \tag{6}$$

$$H * W * Nin * (9 + Nout) \tag{7}$$

From Eqs. 5 and 7, we can see that the conventional convolution layer needed ($9 * Nout$) where only ($9 + Nout$) operations were needed as a depth-separable convolution layer followed by point-wise convolutions. Multiplication is a costly operation with respect to addition for computers. We also used mobileNet-v2 for mass benign and mass malignant classification, which combine inverted ResNet architecture with depth-wise separable and point-wise convolutions.

In ResNet architecture, 3×3 convolution is performed on a reduced number of channels, whereas in MobileNet-v2 architecture the 3×3 convolution layer is replaced by a 3×3 convolution layer with an increased number of channels, but with a smaller number of parameters. The results of our experiments using pre-trained models are listed in Table 1. MobileNet-v1 gives us best accuracy for both DDSM and CBIS-DDSM datasets. Although AlexNet achieve same results as MobileNet-v1, but with huge number of parameters and thus required a lot of computation power.

Table 1. Result comparison. In the table Acc, Rec, and Pre presents Accuracy, Recall and Precision respectively.

CNN	Year	Number of parameters (million)	Mass malignant vs mass benign					
			CBIS-DDSM			DDSM		
			Acc (%)	Rec	Pre	Acc (%)	Rec	Pre
AlexNet	2012	60	73.0	0.74	0.71	86.5	0.90	**0.86**
VGG-16	2014	138	70.2	0.72	0.69	83.0	0.84	0.85
VGG-19	2014	140	70.7	0.72	0.70	84.3	0.85	0.87
ResNet-50	2015	23	63.7	0.66	0.50	85.6	0.89	0.74
GoogLeNet	2014	10	60.0	0.70	0.58	83.0	0.86	0.81
MobileNet-v1	2017	3.2	**74.5**	**0.76**	**0.70**	**86.8**	**0.95**	0.84
MobileNet-v2	2018	2.5	**73.0**	**0.77**	0.65	**85.2**	**0.92**	**0.85**

5 Conclusion and Future Work

In this paper, we provide efficient and accurate CAD system that can help and serve as a second eye for radiologists. For mobile devices and vision-based tools, we provide MobileNet based CNN architecture which takes very less memory space and less computational power. We achieved good accuracy for DDSM, as it has more sample for training and testing. However, CBIS-DDSM is subset of DDSM dataset with high quality images, but less in numbers. Therefore, for model it is difficult to properly learn and generalize well by using less training samples. Accuracy can be achieved by using other preprocessing techniques available in literature or by increasing dataset size. Although MobileNet-v1 give us highest performance but latest version named as MobileNet-v2 is much batter as it performs only 1% less in results by using very less parameters.

In future we aim to provide efficient and accurate mobile and web based clinical-decision support system which is truly independent and deals with all kind of breast cancer screening modalities like mammograms, 3D breast tomosynthesis, MRI, ultrasound. Our system will not only detect cancer but also localize and segment cancer, prove information about cancer size and shape. In addition, generate report for risk assessment with solution treatments like suggest medicine and biopsy treatment. Also remind patient for regular check-up and to visit hospital for test. Exploring other characteristics not restricted to breast tissue would be interesting. Studies have shown powerful correlations with breast cancer in age, gender and family history Future research will also concentrate on domain adaptation and transfer learning techniques from one medical domain to other (e.g. use large dataset for brain and other tumors to train a model from scratch and then fine-tune that model for breast cancer). Furthermore, we will explore cancer prediction techniques which can help and generate alarm before cancer generation. Additionally, developments in CNNs can not only assist radiologists, but eventually also allow independent reading of MGs by diagnostic instruments in the close future.

Acknowledgement. This work has been supported by Higher Education Commission under Grant # 2(1064) and is carried out at Medical Imaging and Diagnostics (MID) Lab at COMSATS University Islamabad, under the umbrella of National Center of Artificial Intelligence (NCAI), Pakistan.

References

1. Siegel, R.L., Miller, K.D., Jemal, A.: Cancer statistics, 2019. CA Cancer J. Clin. **69**, 7–34 (2019)
2. Feig, S.A.: Screening mammography benefit controversies: sorting the evidence. Radiol. Clin. **52**, 455–480 (2014)
3. Welch, H.G., Passow, H.J.: Quantifying the benefits and harms of screening mammography. JAMA Internal Med. **174**, 448–454 (2014)
4. Lee, J.-G., et al.: Deep learning in medical imaging: general overview. Korean J. Radiol. **18**, 570–584 (2017)
5. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
8. Szegedy, C., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
9. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
10. Jiang, F., Liu, H., Yu, S., Xie, Y.: Breast mass lesion classification in mammograms by transfer learning. In: Proceedings of the 5th International Conference on Bioinformatics and Computational Biology, pp. 59–62 (2017)

11. Huynh, B.Q., Li, H., Giger, M.L.: Digital mammographic tumor classification using transfer learning from deep convolutional neural networks. J. Med. Imaging **3**, 034501 (2016)
12. Rampun, A., Scotney, B.W., Morrow, P.J., Wang, H.: Breast mass classification in mammograms using ensemble convolutional neural networks. In: 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom), pp. 1–6 (2018)
13. Hamidinekoo, A., Suhail, Z., Denton, E., Zwiggelaar, R.: Comparing the performance of various deep networks for binary classification of breast tumours. In: 14th International Workshop on Breast Imaging (IWBI 2018), p. 1071807 (2018)
14. Chougrad, H., Zouaki, H., Alheyane, O.: Deep convolutional neural networks for breast cancer screening. Comput. Methods Programs Biomed. **157**, 19–30 (2018)
15. Morrell, S., Wojna, Z., Khoo, C.S., Ourselin, S., Iglesias, J.E.: Large-scale mammography cad with deformable conv-nets. In: Stoyanov, D., et al. (eds.) RAMBO/BIA/TIA 2018. LNCS, vol. 11040, pp. 64–72. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00946-5_7
16. Carneiro, G., Nascimento, J., Bradley, A.P.: Deep learning models for classifying mammogram exams containing unregistered multi-view images and segmentation maps of lesions. In: Zhou, S.K., Greenspan, H., Shen, D. (eds.) Deep Learning for Medical Image Analysis, pp. 321–339. Elsevier, Amsterdam (2017)
17. Carneiro, G., Nascimento, J., Bradley, A.P.: Unregistered multiview mammogram analysis with pre-trained deep learning models. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 652–660 (2015)
18. Heath, M., Bowyer, K., Kopans, D., Moore, R., Kegelmeyer, W.P.: The digital database for screening mammography. In: Proceedings of the 5th International Workshop on Digital Mammography, pp. 212–218 (2000)
19. Lee, R.S., Gimenez, F., Hoogi, A., Miyake, K.K., Gorovoy, M., Rubin, D.L.: A curated mammography data set for use in computer-aided detection and diagnosis research. Sci. Data **4**, 170177 (2017)
20. Teare, P., Fishman, M., Benzaquen, O., Toledano, E., Elnekave, E.: Malignancy detection on mammography using dual deep convolutional neural networks and genetically discovered false color input enhancement. J. Digit. Imaging **30**, 499–505 (2017)
21. Abdelhafiz, D., Yang, C., Ammar, R., Nabavi, S.: Deep convolutional neural networks for mammography: advances, challenges and applications. BMC Bioinform. **20**, 281 (2019)
22. Mishkin, D., Sergievskiy, N., Matas, J.: Systematic evaluation of convolution neural network advances on the imagenet. Comput. Vis. Image Underst. **161**, 11–19 (2017)

U-Net Based Glioblastoma Segmentation with Patient's Overall Survival Prediction

Asra Rafi, Junaid Ali, Tahir Akram, Kiran Fiaz, Ahmad Raza Shahid, Basit Raza(✉), and Tahir Mustafa Madni

Medical Imaging and Diagnostic Lab, National Centre of Artificial Intelligence, Department of Computer Science, COMSATS University Islamabad (CUI), Islamabad, Pakistan
asrarafi456@gmail.com, junaid199f@gmail.com, tahirakram15@hotmail.com, kiransrdr@gmail.com, {ahmadrshahid,basit.raza,tahir_mustafa}@comsats.edu.pk

Abstract. Glioma is a type of malignant brain tumors which requires early detection for patients Overall Survival (OS) prediction and better treatment planning. This task can be simplified by computer aided automatic segmentation of brain MRI volumes into sub-regions. The MRI volumes segmentation can be achieved by deep learning methods but due to highly imbalance data, it becomes very challenging. In this article, we propose deep learning based solutions for Glioma segmentation and patient's OS. To segment each pixel, we have designed a simplified version of 2D U-Net which is slice based and to predict OS, we have analyzed radiomic features. The training dataset of BraTS 2019 challenge is partitioned into train and test set and our primary results on test set are promising as dice score of (whole tumor 0.84, core tumor 0.80 and enhancing tumor 0.63) in glioma segmentation. Radiomic features based on intensity and shape are extracted from the MRI volumes and segmented tumor for OS prediction task. We further eliminate the low variance features using Recursive Features Elimination (RFE). The Random Forest Regression is used to predict OS time. By using intensities of peritumoral edema-label 2 of Flair, the necrotic and non-enhancing tumor core-label 1 along with enhancing tumor-label 4 of T1 contrast enhanced volumes and patients age, we are capable to predict patient's OS with considerable accuracy of 31%.

Keywords: U-Net · Segmentation · Overall Survival Prediction · Machine Learning

1 Introduction

The use of medical images has greatly increased our knowledge of biological processes for research in medical field. Unfortunately, inaccurate diagnosis of brain tumor is the leading cause of deaths among affected people [5]. Therefore, different types of brain tumors are needed to be diagnosed accurately which is essential for proper treatment. The manual evaluation and examination of medical images takes more time and may prone to errors or omissions. Therefore, computer-based analysis and diagnosis can help radiologists in brain tumor diagnosis.

In computer-based methods, image segmentation is a fundamental way to extract useful information from medical images and to identify the abnormalities. Medical community uses X-rays, Computed Tomography (CT) scan and Magnetic Resonance Imaging (MRI) modalities for cancer diagnosis and treatment planning. In MRI a patient is not subjected to harmful radiation and on top of that it also provides excellent soft tissue contrast. Furthermore, MRI highlights different types of soft tissues by using different imaging sequences, such as T1-weighted, contrast enhanced T1-Contrast Enhanced (T1CE), T2-weighted and Fluid Attenuation Inversion Recovery (FLAIR) images.

In this study, the aim is to develop a computer based method for automated segmentation of brain tumor from MRI and predict survival time of patients affected by glioma. Glioma is usually classified into two categories, namely Low Grade Glioma (LGG) and High Grade Glioma (HGG). LGG is slow-growing and generally treated by surgery or radiation therapy, while HGG is fast-growing and has poor prognosis with an average survival of 15 months [6].

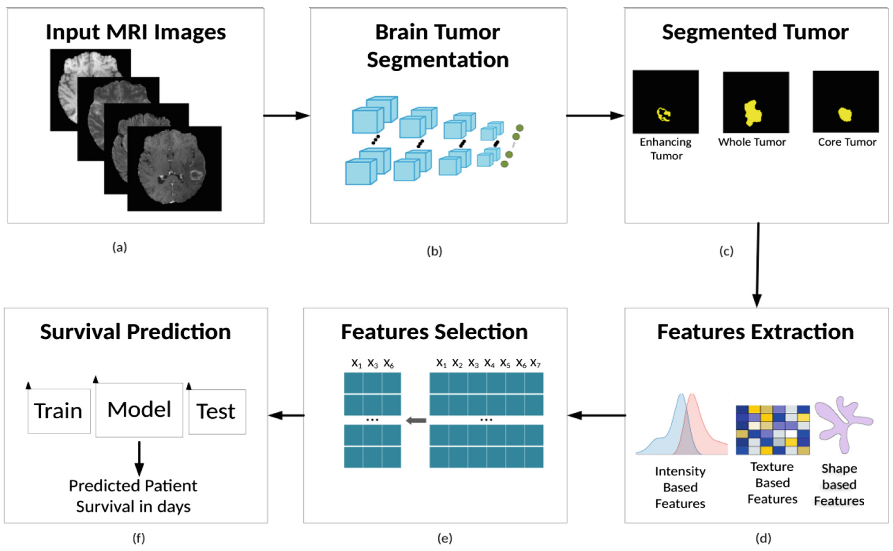


Fig. 1. Proposed System Diagram (a) MRI Scans are given as input to Segmentation model, (b) Brain tumor segmentation of HGG and LGG is done using 3DU-Net model, (c) Segmented tumor results are gathered, (d) Radiomic features based on intensity, texture and shape are extracted from segmented tumor, (e) Significant features are selected using Recursive Features Elimination (RFE), (f) Random Forest (RF) regression model is used to predict overall survival time of patient.

2 Related Work

The task of brain tumor segmentation has recently been dominated by 3D CNN based architecture as it has an additional advantage of depth information from brain MRI volumes. In this regard, one of the early work was proposed by Kamnitsas et al. [7] that uses 3D patches extracted from brain MRI volumes and dual pathway CNNs to extract features from different scales. Although it was using depth information and different scales but it had low inference efficiency. Wang et al. [8] proposed a cascaded architecture which divides the overall multi-class problem into three binary class problems and designed multi CNN based architectures. It has good generalization capabilities and achieved second place on BraTS 2017 challenge. An efficient system based on 2D and 3D CNN based features is proposed by Mlynarski et al. [9]. They combined the features extracted by 2D models with 3D CNN but due to multiple networks, it has high computational overhead.

In [10] Chen et al. proposed a deep convolutional symmetric neural network for segmentation of brain tumor. Their proposed model consist of Deep Convolutional Neural Network (DCNN) with symmetric masks added in different layers. They have mainly focused on run-time segmentation of MRI volumes in seconds but their scores of enhancing tumor, tumor core and whole tumor are 0.58, 0.68 and 0.85 respectively. Pereira et al. in an article [11] have proposed a model in which they have mixed the information with the linear expansion of feature, by which only relevant features are obtained. The model is computationally expensive as it is using two networks (Binary FCN and Multi-Class FCN) for training. They have achieved dice score of 0.86, 0.76, 0.69 whole core and enhancing respectively. Their model have performed poor on tumor core as compare to our model.

In this study, we have evaluated specifically edema-label 2 of FLAIR, the necrotic and non-enhancing tumor core-label 1 along with enhancing tumor label 4 of T1-CE volumes, which provides more information of tumor region leading to better OS prediction. Most of the work in literature for OS prediction is based on radiomic features extracted from MR images. Sanghania et al. have used clinical, shape, texture and volumetric features for 2 class (short, long) and 3 class (short, medium, long) survival group prediction. The Support Vector Machine classification based Recursive Features Elimination (SVM-RFE) method is used for feature selection [12].

A statistical analysis of multi-scale texture features for the characterization of tumor regions has been performed for progression free survival and OS prediction by Ahmad Chaddad et al. [13]. Recently, Sun et al. have addressed the problems of brain tumor segmentation and 3 class OS prediction using ensemble of 3DCNN and Random forest classifier respectively. Radiomic features are used for the survival time prediction [14]. Furthermore, a study has been done by Sanghania et al., where they have evaluated the effectiveness of tumor shape features for OS prognosis in Glioma patients. They have evaluated abnormality regions of FLAIR and T1-CE MR images of patients [15]. They concluded that 3D shape based radiomic features are mostly associated with OS prognosis.

3 Methodology

This article is based on two basic tasks. In task 1, the brain tumor segmentation of two tumor grades HGG and LGG is performed. For that purpose, U-Net [16] is used for the segmentation of LGG and HGG. In task 2 radiomic features has been extracted using segmented tumor and MRI scans. Furthermore, high variance features along with age has been selected using RFE and fed to RF for the patient's OS prediction. The overall system diagram is shown in Fig. 1.

3.1 Data Preprocessing

Data preprocessing is an essential step before training the deep neural network. One MRI volume has $240 \times 240 \times 155 \times 5$ dimension. To make training process efficient, the irrelevant border from each slice is cropped to 224×224. This process of cropping has been done automatically by ignoring 'n' number of pixels from the height and m number of pixels from width, where $n = 8$, $m = 8$. The values of 'n' and 'm' are determined by visualizing various slices of MRI volumes showing, 224×224 area of slice as brain part and remaining as background. The 224×224 slice is obtained by selecting [0 : 8, 232 : 240, :, :] intensity values from [240, 240, :, :] slice. Furthermore, the Z-score normalization [17] is used for normalizing each image modality within brain regions as in this normalization approach the standard deviations of below and above the mean value is measured. Min-Max normalization is also tested for image normalization but it performs poorly on this U-net model. The preprocessing process is shown in Fig. 2.

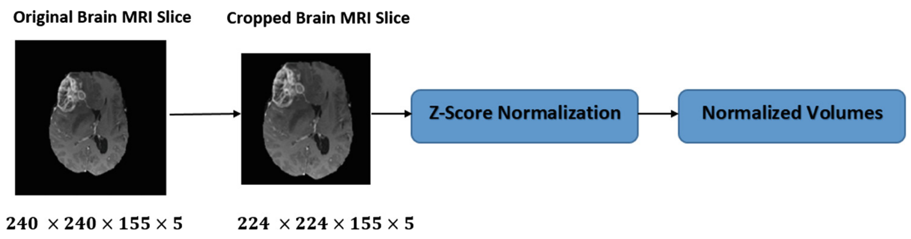


Fig. 2. Process diagram of preprocessing for brain tumor segmentation

3.2 Brain Tumor Segmentation

The segmentation task of BraTS 2019 dataset is challenging mainly because of data imbalance among different classes. The selection of five patients from dataset, to show class imbalance problem is random. Their statistics for each type of glioma are given in Table 1. This class imbalance is more severe in case of LGG and it has been observed that the segmentation networks perform poorly with LGG volumes.

To segment HGG and LGG volumes, a slice based 2D U-Net is used with slight modifications in the sense of reconstruction of image at decoder path shown in Fig. 3. A Batch Normalization (BN) layer is used after all hidden convolution layers. To preserve useful information during training Leaky ReLu (LReLu) is used in all hidden layers. The stride of 2 is used in hidden convolution layers which acts as down-sampling operation in the encoder pathway. In decoder path ConvTranspose Layer is used for UpSampling followed by dropout. Instead of using Upsampling layer as used by [16], ConvTranspose layer is used. The UpSampling layer doubles the input dimension whereas ConvTranspose layer perform inverse convolution operation. ConvTranspose layer have learnable parameters as compare to UpSampling layer. The addition of dropout layer has significantly reduced the over-fitting problem and provides better generalization.

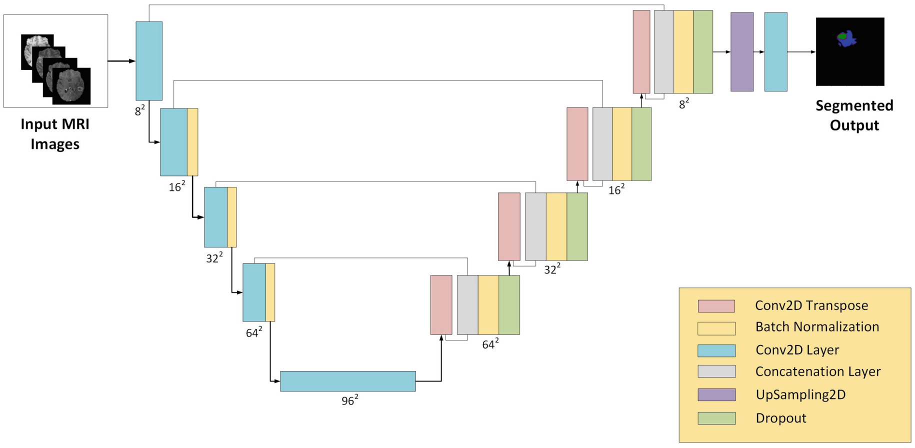


Fig. 3. U-Net architecture with slight changes. After each convolution layer in Down-Sampling path, a BN layer is added and at up-sampling path dropout layer is used after BN to prevent over-fitting.

Table 1. Class distribution of five patients according to LGG and HGG

LGG	Class 0	Class 1	Class 2	Class 4
	99.6500	0.0500	0.2900	0.0000
	99.3000	0.4800	0.2000	0.0000
	99.7800	0.1000	0.1000	0.0000
	97.2200	1.0200	1.0500	0.1069
	99.7800	0.1000	0.1100	0.0000
HGG	99.7000	0.0046	0.1700	0.0500
	97.6400	0.3800	1.4900	0.4700
	97.8900	0.2400	1.4500	0.4100
	98.4400	0.5100	0.8300	0.2200
	99.0900	0.3000	0.3900	0.2100

The data imbalance problem has been reduced by using a specialized tversky loss function [18] as shown in Eq. 1. This loss function generalizes the dice score and it outperforms the simple dice loss. In case of dice loss, the false positives (FPs) and false negatives (FNs) were treated equally which results in low recall and high precision values. The tversky loss focuses more on FNs which provides a better tradeoff between precision and recall. This loss function is used in segmentation task. It is defined as follows:

$$S(P, G; \alpha, \beta) = \frac{|PG|}{|PG| + \alpha|P\backslash G| + \beta|G\backslash P|} \tag{1}$$

Where P denotes predictions and G stands for ground truth. Where α and β are the parameters used to control the magnitude value of penalties for FNs and FPs. The full form of tversky loss function is given in Eq. 2.

$$T(\alpha, \beta) = \frac{\sum_{i=1}^{N} p_{0_i} g_{0_i}}{\sum_{i=1}^{N} p_{0_i} g_{0_i} + \alpha \sum_{i=1}^{N} p_{0_i} g_{0_i} + \beta \sum_{i=1}^{N} p_{0_i} g_{0_i}} \tag{2}$$

Where the probabilities p_{0i} and p_{1i} are related to lesion voxel i and non-lesion voxel i respectively. The g_{1i} is 1 for non-lesion voxel and 0 for lesion voxel and vice versa for the g_{0i}.

3.3 Overall Survival Time Prediction

To identify the clinical relevance of task 1 (segmentation), this section is mainly focusing on OS time prediction by analyzing radiomic features and algorithms based on Machine Learning (ML). In the segmentation task, segmented labels of tumor in the preoperative MR Scans are produced, which are used for feature extraction in OS prediction task. Radiomic feature are extracted from Flair and T1CE modalities.

In the next step, these extracted features are given to the RFE for the selection of significant features. Furthermore, these features are then analyzed through RF model for the prediction of patient's OS time. There are three categories of survivors on the basis of their survival days: the long time survivors ($>$15 months), short time survivors ($<$10 Months) and mid time survivors are those which can live between 10 and 15 months. The overall process of OS prediction has been described in the following manner.

Radiomic Feature Extraction. The location and geometry of tumor hold an important role in deciding the survival days [19]. Four raw MRI volumes described in (Sect. 1) along with segmented tumor are used for feature extraction. In total '3393' features are extracted using python based radiomic feature extraction library named Pyradiomics [20]. There are three labels in brain tumor named Necrotic and Non-Enhancing Tumor core (NCR/NET)-label 1, peritumoral Edema (ED)-label 2 and Enhancing Tumor (ET)-label 4 [21]. The segmented tumor along with Flair and T1CE modalities is used to extract appropriate radiomic/imaging features. For the extraction of these features, ED-label 2 with Flair modality and the NCR/NET-label 1 along with ET-label 4 are used with T1CE specifically. The label types and modality types are needed to be defined in the setting file of Pyradiomics library.

Pyradiomics extracts features in 2D and 3D Images by calculating the mean intensity of each image and region pair. We utilized available features of every feature type that includes: Gray Level Co-occurrence matrix (GLCM), Gray Level Run Length Matrix (GLRLM), Gray Level Size Zone Matrix (GLSZM), Gray Level Dependence Matrix (GLDM), First Order Features, Neighboring Gray Tone Difference Matrix (NRGTM)

[12] and Shape features. We have extracted radiomic features from MRI volumes, and abnormal tumor regions of segmented tumor according to tumor type (i.e. edema-label 2 of FLAIR, the necrotic and non-enhancing tumor core-label 1 along with enhancing tumor label 4 of T1-CE volumes). The size of the tumor and its ratio according to the entire shape was extracted as well. The other features from image named mean, skewness, variance, standard deviation, histogram and entropy feature intensities of the tumor were also extracted.

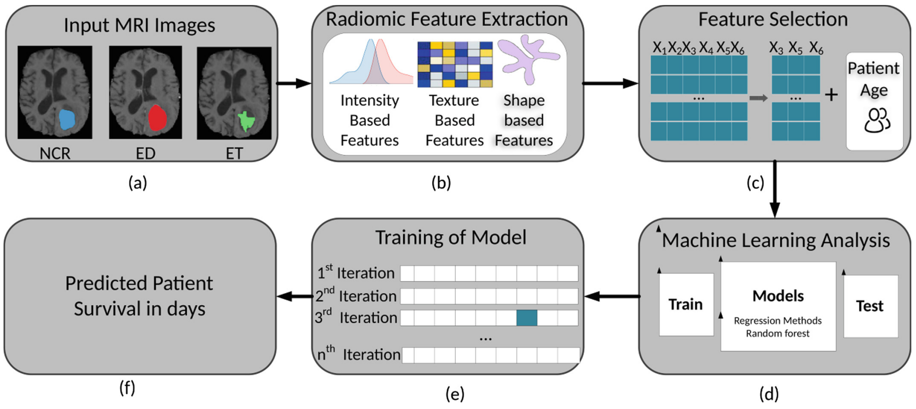


Fig. 4. Overall Survival Prediction Pipeline (a) Segmented tumor as input, (b) Radiomic features extraction from segmented tumor, (c) Important features selection using RFE, (d) RF regression model for OS prediction, (e) 5-fold cross validation to validate the model performance, (f) Predicted result in days.

Feature Selection: To eliminate the irrelevant or least important features we have performed feature selection in three steps. At first to remove redundant features several experiments has been performed using multiple combinations of extracted features and their impact on result has been observed. Furthermore, we have selected 76 significant features using RFE and fed them to the regression model. RFE [22] is a feature selection technique that select relevant features by fitting them on a model and remove irrelevant features until the required number of features are selected as specified in the program.

Regression Model: We have implemented RF regression model for the prediction of overall survival using significant features extracted and selected in previous steps. RF is an approach based on ensemble for the regression, classification and many other tasks depending on the shape and type of data. The RF regression model is used for the prediction task as the target variable named survival is a continuous variable, so that support vector machine and other classification models are not appropriate to this problem of regression. It creates different number of decision trees at the time of model training and randomly selects among them for making best decisions. The model has given the accuracy of 0.31 as shown in Table 2. The overall survival prediction pipeline for regression is shown in Fig. 4. In the OS prediction pipeline MRI volumes and segmented tumor are used as input Fig. 4(a). As shown in Fig. 4(b) radiomic features based on intensity, texture and shape are extracted from segmented tumor, Fig. 4(c) describe feature selection as 76 important features are selected using RFE, including clinical feature age and Gross Resection Status (GTR), In Fig. 4(d), it can be observed that after dividing data into 80% train and 20% test, the RF regression model is used to predict overall survival time of patient, then in the next step, 5-fold cross validation is performed to validate the model performance as shown in Fig. 4(e). Finally, model has returned predicted survival of patients in days as presented in Fig. 4 (f). The results of segmentation for all tumor types are shown in Table 2.

4 Results

In this section experimental results are defined in tabular format. The authors in [1–4] have used BraTS 2018 dataset (subset of BraTS 2019) for the segmentation of brain tumor. However, in this paper, BraTS 2019 dataset is used to train and test the models [23]. The multimode brain MRI scans of 335 patients, 259 Glioblastoma (GBM) HGG and 76 LGG are used for the training. Each patient comprises 4 modalities of image including FLAIR, T1, T1CE, and T2 along with ground truth label. We have used 80:20 split, 80 percent of the data as training set and 20 percent of the data as test set for the model testing. The data has been pre-processed using standard preprocessing methods. All provided scans are without skull, following same anatomical pattern, and are re-sampled to resolution 1 mm^3. The shape of each volume is $240 \times 240 \times 155 \times 5$.

Table 2. Result of U-Net on BraTS 2019 data

Data set	Grade	WT-Dice	TC-Dice	ET-Dice
Training	HGG	0.84	0.80	0.63
	LGG	0.80	0.77	0.60
Validation	HGG	0.81	0.79	0.61
	LGG	0.79	0.76	0.59

The experiments for OS prediction task are performed on 76 patient's volumes from training and 29 patient's volumes from validation dataset, and got 0.31 and 0.27 accuracy respectively as shown in Table 3.

Table 3. Results summary for the prediction of patient's survival

Dataset	Patient's cases	Model accuracy	MSE	MedianSE	StdSE	SpearmanR
Training	76	31.65%	198540.6568	63196.073	509513070.6	−0.024
Validation	29	27.25%	208540.6568	72183.073	489513070.6	−0.013

4.1 Evaluation Metrics

The most common evaluation metric in segmentation task is Dice Similarity Coefficient (DSC) and in survival predication, accuracy, Mean Square Error (MSE), Median Squared Error (SE), Standard Deviation of Squared Errors (stdSE) and SpearmanR.

$$DSC(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{3}$$

The DSC is used to find the similarity between ground truth and predicted labels of the model as presented in Eq. (3). Afterword's, we calculate average of those values to find overall dice score.

$$MSE = \frac{1}{n} \sum \left(Y - \hat{Y}\right)^2 \tag{4}$$

$$Accuracy = \frac{(TN + TP)}{(TN + FP + FN + TP)} \tag{5}$$

$$SpearmanR = \rho_{rg_Y rg_{\hat{Y}}} = \frac{covr\left(rg_Y rg_{\hat{Y}}\right)}{\sigma_{rg_X} \sigma_{rg_Y}} \tag{6}$$

$$SE(median) = 1.2533 \times SE(\mu) \tag{7}$$

The MSE in Eq. (4) is used to find how close our line is to the actual points. It is calculated by sum of square of difference between actual and predicted values and divide the average with them. Accuracy in Eq. (5) is used to measure the closeness of the value. It is the proportion of true results (TN + TP) with the overall results. Where TP = True Positive, TN = True Negative, FP = False Positive and FN = False Negative. The spearman rank correlation is used to find correlation between ranks of two values. In Eq. (6), rg_Y is the rank of variable Y and $rg_{\hat{Y}}$ is the rank of variable $\hat{Y}$. Where Y is actual and $\hat{Y}$ is predicted value. It is calculated by dividing the correlation between two ranks with standard deviation of two ranks.

Where, SE (median) is the standard error of the median and $SE(\mu)$ is the standard error of the mean. The MedianSE is shown in Eq. (7). The smaller the MedianSE, the more support there is for the corresponding model. The MedianSE is preferred to the mean squared error in the evaluation of model due to large outliers in the patient's survival data [24]. StdSE is another evaluation metric for error calculation in regression. Standard deviation identifies how correctly mean represents sample data.

5 Conclusion

In this study, we have proposed the U-Net based dense segmentation system which is able to achieve good dice scores for HGG volumes from multi-modal BRATS 2019 dataset. Our primary results on test set are promising as dice score of (whole tumor 0.84, core tumor 0.8 and enhancing tumor 0.63) in glioma segmentation. However, the network performance degraded for LGG volumes which needs to be further investigated. It can be observed from the analysis that the data imbalance problem is more serious in LGG volumes thus special care must be taken in order to perform automatic segmentation of LGG volumes. In future, we are planning to design separate networks for HGG and LGG segmentation. On the other side, for survival prediction task, RFE has eliminated low variance features and selected more relevant features, which are then fed to RF regression model for the prediction of patient's OS. In result, this model give 0.31 and 0.27 accuracy with MSE of 198540.65 and 208540.6568 on the training set and test set respectively. It is observed that, the OS prediction task has scope of improvement which will be provided in future by statistically analysing features extracted from region of interest. Furthermore, the relation of these features will be investigated with OS, which may leads to improved prognosis of survival.

Acknowledgement. This work has been supported by Higher Education Commission under Grant # 2(1064), and is carried out at Medical Imaging and Diagnostics (MID) Lab at COMSATS University Islamabad, under the umbrella of National Center of Artificial Intelligence (NCAI), Pakistan.

References

1. Menze, B.H., et al.: The multimodal brain tumor image segmentation benchmark (BRATS). IEEE Trans. Med. Imaging **34**(10), 1993–2014 (2014)
2. Bakas, S., et al.: Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features. Sci. Data **4**, 170117 (2017)
3. Bakas, S., et al.: Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge (2018). arXiv preprint arXiv:1811.02629
4. Bakas, S., et al.: Segmentation labels and radiomic features for the pre-operative scans of the TCGA-LGG collection. The Cancer Imaging Archive, p. 286 (2017)
5. Selvakumar, J., Lakshmi, A., Arivoli, T.: Brain tumor segmentation and its area calculation in brain MR images using K-mean clustering and Fuzzy Cmean algorithm. In: IEEE-International Conference on Advances in Engineering, Science and Management (ICAESM-2012), pp. 186–190. IEEE, March 2012
6. Liang, D., Schulder, M.: The role of intraoperative magnetic resonance imaging in glioma surgery. Surg. Neurol. Int. **3**(Suppl. 4), S320 (2012)
7. Kamnitsas, K., et al.: Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. Med. Image Anal. **36**, 61–78 (2017)
8. Wang, G., Li, W., Ourselin, S., Vercauteren, T.: Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks. In: Crimi, A., Bakas, S., Kuijf, H., Menze, B., Reyes, M. (eds.) BrainLes 2017. LNCS, vol. 10670, pp. 178–190. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75238-9_16

9. Mlynarski, P., Delingette, H., Criminisi, A., Ayache, N.: 3D convolutionalneural networks for tumor segmentation using long-range 2D context. Comput. Med. Imaging Graph. **73**, 60–72 (2019)
10. Chen, H., et al.: Brain tumor segmentation with deep convolutional symmetric neural network. Neurocomputing (2019). https://doi.org/10.1016/j.neucom.2019.01.111
11. Sérgio, P., et al.: Adaptive feature recombination and recalibration for semantic segmentation with Fully Convolutional Networks. IEEE Trans. Med. Imaging **38**(12), 2914–2925 (2019)
12. Sanghani, P., Ang, B.T., King, N.K.K., Ren, H.: Overall survival prediction in glioblastoma multiforme patients from volumetric, shape and texture features using machine learning. Surg. Oncol. **27**(4), 709–714 (2018)
13. Chaddad, A., Sabri, S., Niazi, T., Abdulkarim, B.: Prediction of survival with multi-scale radiomic analysis in glioblastoma patients. Med. Biol. Eng. Comput. **56**(12), 2287–2300 (2018)
14. Sun, L., Zhang, S., Chen, H., Luo, L.: Brain tumor segmentation and survival prediction using multimodal MRI scans with deep learning. Front. Neurosci. **13**, 810 (2019)
15. Sanghani, P., Ti, A.B., King, N.K.K., Ren, H.: Evaluation of tumor shape features for overall survival prognosis in glioblastoma multiforme patients. Surg. Oncol. **29**, 178–183 (2019)
16. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
17. Patro, S., Sahu, K.K. Normalization: A preprocessing stage (2015). arXiv preprint arXiv:1503.06462
18. Salehi, S.S.M., Erdogmus, D., Gholipour, A.: Tversky loss function for image segmentation using 3D fully convolutional deep networks. In: Wang, Q., Shi, Y., Suk, H.-I., Suzuki, K. (eds.) MLMI 2017. LNCS, vol. 10541, pp. 379–387. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67389-9_44
19. Prez-Beteta, J., et al.: Glioblastoma: does the pre-treatment geometry matter? A postcontrast T1 MRI-based study. Eur. Radiol. **27**(3), 1096–1104 (2017)
20. https://pyradiomics.readthedocs.io/en/latest/features.html. Accessed 2 Aug 2019
21. Chen, W., Liu, B., Peng, S., Sun, J., Qiao, X.: S3D-UNet: separable 3D U-Net for brain tumor segmentation. In: Crimi, A., Bakas, S., Kuijf, H., Keyvan, F., Reyes, M., van Walsum, T. (eds.) BrainLes 2018. LNCS, vol. 11384, pp. 358–368. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11726-9_32
22. https://scikit-learn.org/stable/modules/generated/sklearn.featureselection.RFE.html. Accessed 2 Aug 2019
23. https://www.med.upenn.edu/cbica/brats2019/data.html. Accessed 4 Dec 2019
24. Anacleto Junior, O.: Bayesian dynamic graphical models for high-dimensional flow forecasting in road traffic networks. Doctoral dissertation, The Open University (2012)

MicroPython or Arduino C for ESP32 - Efficiency for Neural Network Edge Devices

Kristian Dokic[1](✉), Bojan Radisic[1], and Mirko Cobovic[2]

[1] Polytechnic in Pozega, Pozega, Croatia
kdjokic@vup.hr
[2] College of Slavonski Brod, Pozega, Croatia

Abstract. In the last few years, microcontrollers used for IoT devices became more and more powerful, and many authors have started to use them in machine learning systems. Most of the authors used them just for data collecting for ML algorithms in the clouds, but some of them implemented ML algorithms on the microcontrollers. The goal of this paper is to analyses the neural networks data propagation speed of one popular SoC (Espressif System company ESP32) with simple neural networks implementation with two different development environment, Arduino IDE and MycroPython. Neural networks with one hidden layer are used with a different number of neurons. This SoC is analysed because some companies started to produce them with UXGA (Ultra Extended Graphics Array) camera implemented and it can be used to distribute computing load from central ML servers.

Keywords: Neural network · ESP32 · Arduino · MicroPython

1 Introduction

The traditional approach of artificial intelligence algorithm usage forces centralised schema. This approach is simpler, but in some cases, it makes the high load on a central server. One solution is to transfer tasks from the central server to smaller processors or microcontrollers on edge, and this is called *edge computing*. This approach reduces network traffic since edge nodes upload reduced data instead of complete input data [1].

In the last few years, many microcontroller producers have worked on artificial intelligence implementation on microcontrollers. Some of them have developed special libraries with AI functions, but the others have implemented special hardware with enhanced AI capabilities.

In this paper, a low-cost Chinese SoC ESP32 is tested with feed-forward neural networks on two different development environment, Arduino IDE and MycroPython. Espressif System, a company from China introduced ESP32 before three years. It does not have machine learning capabilities. It is manufactured using a 40 nm process by TSMC (Taiwan Semiconductor Manufacturing Company) [2]. Many projects have been developed with ESP32 SoC like water pumping solar system [3], or monitors for Liquefied petroleum gas [4]. Some authors suggested that ESP32 will play one of the major role in future Internet of Things devices development [5].

In the second section, some papers that deal with machine learning usage and ESP32 are presented, but in the third section, some development environments for ESP32 are presented. In the fourth section, neural networks forward propagation times with a different number of neurons in the hidden layer are measured. Finally, in section five, discussion and conclusion can be found.

2 ESP32 and Machine Learning

ESP32 is used for various machine learning tasks, but usually only for measuring and collecting data for dislocated machine learning servers. On the other hand, some papers deal with ESP32 usage in projects that have machine learning algorithms implemented. In the next two paragraphs, papers that deal with these two different approaches are analysed.

2.1 Machine Learning on the Cloud

Zidek et al. used ESP32 SoC with different sensors for wireless devices input data optimization. Data was accumulated into the packet and then the hole packet was sent to the cloud service [6].

Rosato and Masciadri have chosen Logistic Regression to detect sitting person with the device based on ESP32 SoC. They made monitoring system and the key component with ESP32 was installed under the person chair. Data has been analysed on the server, and ESP32 has been wirelessly connected to it [7].

Islam et al. designed small device that can record electrical activity of the human heart using electrodes placed on the skin. This device has been based on the Analog Devices AD8232. They used ESP32 SoC to send data to the server. Linux server has been used to analyse data with machine learning algorithms [8].

Fernoaga et al. used ESP32 to take pictures of gas, electricity or water counters. These devices have had implemented cameras and they have sent pictures to the cloud. Numbers from the counters are recognised by neural networks from supplied pictures [9].

Komarek et al. presented a project that provide a layer that uses MQTT (MQ Telemetry Transport) for interfacing sensors and nodes. They used ESP32 SoC to send data to other nodes and the cloud for the ambient intelligence system [10].

Chand et al. used ESP32 SoC for a person behavior and status prediction in one room. They used ultrasonic sensors and data from sensors are sent to a server. The main goal of the server was to decide whether it is normal or abnormal situation. This decision system is based on machine learning algorithms [11].

2.2 Machine Learning on an ESP32

Espressif Systems company developed ESP-WHO framework for face detection and recognition. This framework is based on Multi-task Cascaded Convolutional Networks model and new mobile architecture - MobileNetV2 and it is available on the GitHub.

The main difference between this framework and previously mentioned projects is in the fact that ESP-WHO has face detection and recognition algorithms on the ESP32 SoC [12, 13].

Kokoulin et al. used ESP32 SoC for the system that process video stream and detect presence of faces or silhouettes. Only images with faces or silhouette are sent to the central face recognition server. The main goal of that system were network traffic reduction as well as central face recognition server computing load reduction. Estimated network traffic decrease up to 80–90% [14].

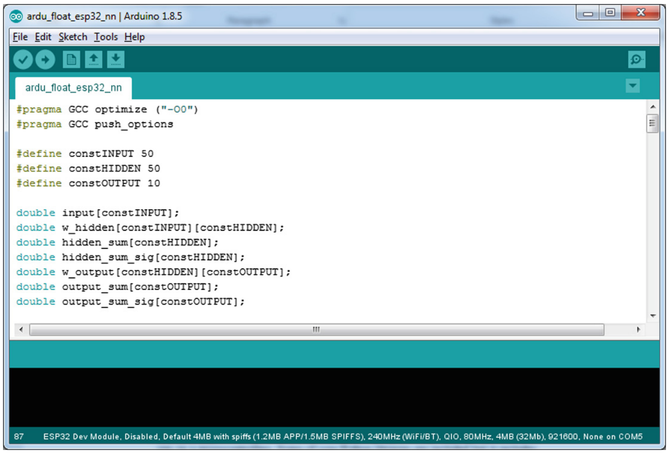
3 ESP32 Development Environments

ESP-IDF (Espressif IoT Development Framework) is the official development framework for the ESP32 SoC. ESP-IDF is the most powerful framework, and it can be used with Linux and Windows OS. Because of ESP32 popularity some other tools are adapted to ESP32. Most popular are Arduino IDE and MicroPython, and they will be analysed.

On the other hand, there are lots of development environments that can be used for ESP32 projects development. KB-IDE is open IDE available on GitHub for ESP32 SoC boards. It supports visual programming similar to Scratch, Arduino style programming, and the official ESP-IDF framework for more experienced users (https://kbide.org/). Lua programming language can be used with ESP32, too. With ChiliPeppr ESP32 Web IDE, a browser can be used for editing/uploading Lua code to ESP32 device. The ChiliPeppr IDE has a serial port JSON (JavaScript Object Notation) Server that can be used locally or remotely to let browser communicate directly to serial port and ESP32 (http://chilipeppr.com/esp32). ESP32 is supported by PlatformIO. It is a cross-platform code builder and library manager. It supports more than 200 development boards, 15 development platforms and ten frameworks.

3.1 Arduino IDE

Arduino IDE is java application used to write programs for Arduino compatible boards. It can be used with ESP32, but first some additional software components have to be downloaded before use. The Arduino IDE supports simplified versions of C and C++ languages. Arduino is well known Italian company with lots of microcontroller boards that are licensed under the LGPL (Lesser General Public License) or the GPL (General Public License). In Fig. 1, the Arduino IDE can be seen.


ardu_float_esp32_nn | Arduino 1.8.5
File Edit Sketch Tools Help
ardu_float_esp32_nn

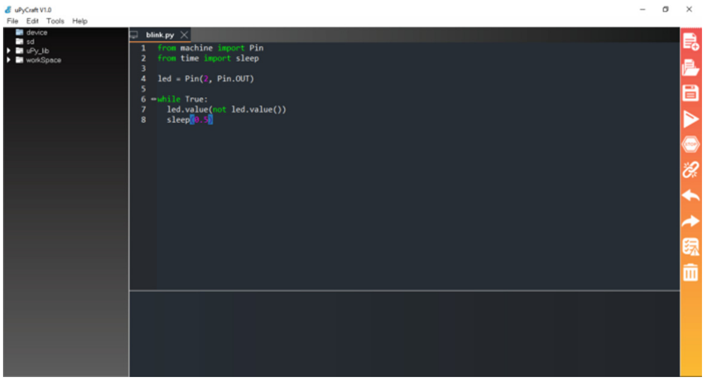
```
#pragma GCC optimize ("-O0")
#pragma GCC push_options

#define constINPUT 50
#define constHIDDEN 50
#define constOUTPUT 10

double input[constINPUT];
double w_hidden[constINPUT][constHIDDEN];
double hidden_sum[constHIDDEN];
double hidden_sum_sig[constHIDDEN];
double w_output[constHIDDEN][constOUTPUT];
double output_sum[constOUTPUT];
double output_sum_sig[constOUTPUT];
```

```
87    ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM5
```

Fig. 1. Arduino IDE

3.2 MicroPython

MicroPython is an implementation of a Python programming language optimised to run on some microcontrollers. Some of the core Python libraries are included, but it includes modules that allow low-level hardware access to the programmer. MicroPython was created by Australian programmer Damien George. It does not have rich IDE, but on their web page spycraft IDE is suggested. In Fig. 2, uPyCraft IDE can be seen.



```
from machine import Pin
from time import sleep

led = Pin(2, Pin.OUT)

while True:
  led.value(not led.value())
  sleep(0.5)
```

Fig. 2. uPyCraft IDE

4 Neural Networks Forward Propagation Speed

Teerapittayanon suggests that their Distributed Deep Neural Networks model can reduce the communication cost by a factor of over 20x but to achieve this, so-called “edge” or “end” devices have to use simple Machine Learning models and process data from sensors [15]. Zhang describes keyword spotting device based on Cortex-M7 based STM32F746G-DISCO development board. Keyword spotting devices are typical examples of edge devices that are based on tiny microcontrollers with limited memory and compute capability. They require real-time response and high accuracy for good user experience [16]. Lai quotes that “Deep Neural Networks are becoming increasingly popular in always-on IoT edge devices performing data analytics right at the source, reducing latency as well as energy consumption for data communication”. Lai used Arm Cortex-M processor board and achieved 4.6X improvement in runtime/throughput and 4.9X improvement in energy efficiency with CMSIS-NN kernels [17].

There are lots of examples where a neural network is trained on a powerful computer and then deployed to tiny microcontroller to analyse some input data. In this paper ESP32 is tested for that task. The simple neural network is programmed with random weights and different number of nodes and data propagation time is measured.

4.1 Data and Methods

Neural networks with 50 input nodes and ten output nodes are programmed with Arduino IDE and MicroPython. The number of hidden nodes in one hidden layer is between 50 and 200. A logistic sigmoid activation function is used (Fig. 3).

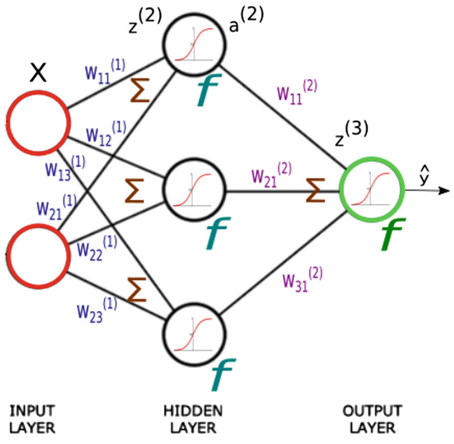


Fig. 3. Neural network with one hidden layer

Setup for testing is in Table 1.

Table 1. Testing setup

	Input layer	Hidden layer	Output layer	uC	Weights format
1	50	50	10	Arduino	float
2	50	100	10	Arduino	float
3	50	150	10	Arduino	float
4	50	200	10	Arduino	float
5	50	50	10	Arduino	double
6	50	100	10	Arduino	double
7	50	150	10	Arduino	double
8	50	200	10	Arduino	double
9	50	50	10	MicroPython	float
10	50	100	10	MicroPython	float
11	50	150	10	MicroPython	float
12	50	200	10	MicroPython	float

4.2 Microcontrollers

Two ESP32 based microcontrollers are used for testing purposes. ESP32 WROOM is in Fig. 4, and PYCOM WIPY 3.0 is in Fig. 5. The main difference between them is that PYCOM WIPY 3.0 has MicroPython installed and ready to be used. ESP32 WROOM can be used with Arduino IDE without any setup.

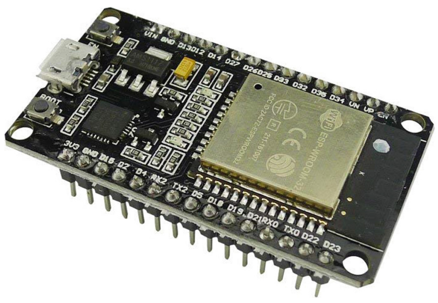


Fig. 4. ESP32 WROOM (Arduino)

Both boards have Xtensa dual-core 32-bit LX6 microprocessor operating on 240 MHz. ESP32 WROOM has only 520 KB RAM, but PYCOM WIPY 3.0 has 520 KB and 4 MB additional memory. WiFi and Bluetooth modules characteristics are same, but ESP32 WROOM has serial to USB controller implemented on board.

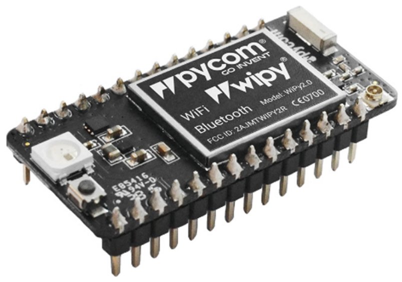


Fig. 5. PYCOM WIPY 3.0 (MicroPython)

4.3 Testing

There are different ways to test the speed of neural network data propagation, but in this research digital oscilloscope is used. Some authors used implemented functions milis() and micros() on both platforms but measuring with oscilloscope is better solution because some processes in microcontroller cannot affect measuring.

To measure the time, GPIO (general-purpose input/output) output of the microcontroller is connected to an oscilloscope. Simple code in Arduino C and MicroPython with tasks listed in Table 2 is prepared. This code has been used to measure frequency of pin output rise and fall but these periods are too small to have any effect on measuring accuracy. Periods can be seen in Table 3. All code listings can be found on GitHub.

Table 2. Code in Arduino C and MicroPython for pin output rise and fall

State	Arduino C	MicroPython
GPIO ON	digitalWrite(LED_BUILTIN, HIGH);	led.value(1)
GPIO OFF	digitalWrite(LED_BUILTIN, LOW);	led.value(0)

The results are in Table 3.

Table 3. Rise/fall period (Arduino C and MicroPython)

Platform	Period
Arduino C	376 ns
MicroPython	11,75 µs

Oscilloscope screenshot can be seen in Fig. 6.

In the next step, neural network code has been implemented two times after GPIO ON and GPIO OFF code on both platforms. The flowchart can be seen in Fig. 7.

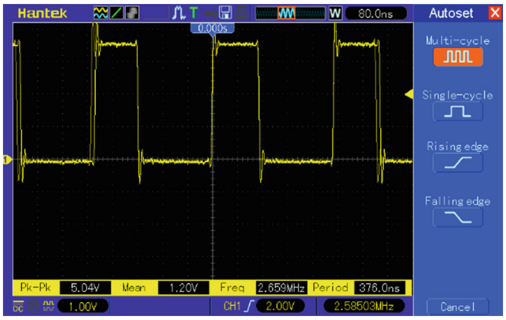


Fig. 6. Oscilloscope screenshot

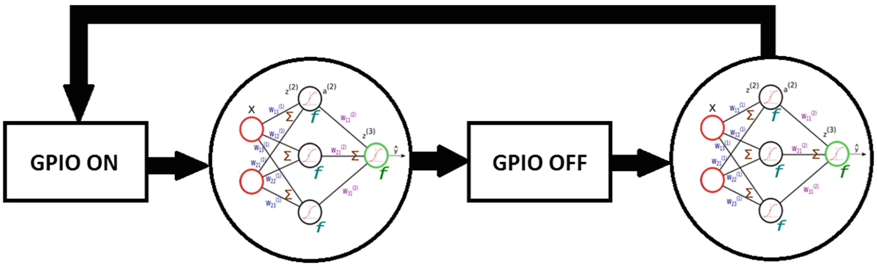


Fig. 7. Flowchart

Table 4. Final results

	Input layer	Hidden layer	Output layer	uC	Weights format	Period
1	50	50	10	Arduino	float	2,58 ms
2	50	100	10	Arduino	float	5,05 ms
3	50	150	10	Arduino	float	7,56 ms
4	50	200	10	Arduino	float	10,02 ms
5	50	50	10	Arduino	double	6,50 ms
6	50	100	10	Arduino	double	13,04 ms
7	50	150	10	Arduino	double	19,63 ms
8	50	200	10	Arduino	double	26,01 ms
9	50	50	10	MicroPython	float	153,84 ms
10	50	100	10	MicroPython	float	304,41 ms
11	50	150	10	MicroPython	float	423,91 ms
12	50	200	10	MicroPython	float	586,51 ms

Results are in Table 4. Arduino C can use two floating-point number precisions, and both of them are used in measuring. New MicroPython firmware support double-precision floating-point numbers but it was unavailable to us, and in Table 4 there are only four rows concerning MicroPython.

Results can also be seen in Fig. 8.

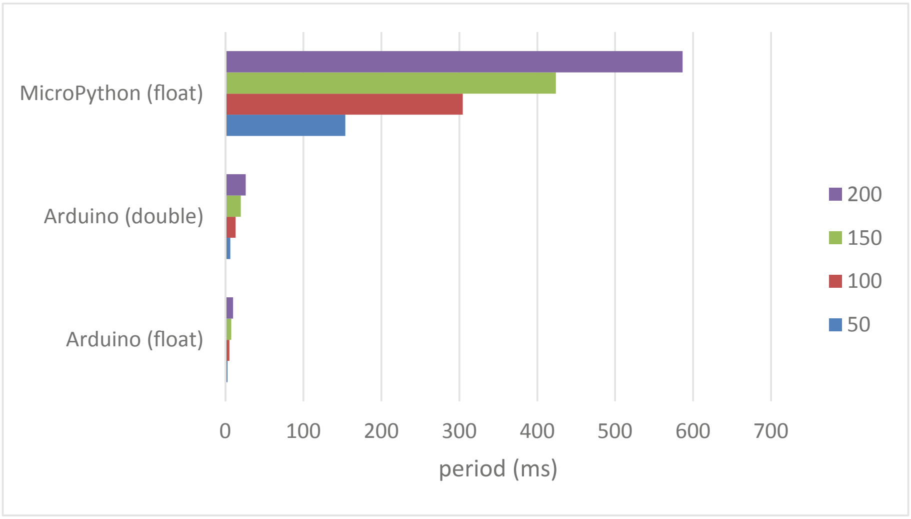


Fig. 8. Periods for different platforms and hidden nodes

5 Discussion and Conclusion

It can be seen that Arduino C based neural network has significant lower data propagation latency than MicroPython based neural network. Some authors indicated this, but they have tested propagation speed with integers. On the GitHub platform number of counts running for 10 s for three different setups can be found. MicroPython on Teensy 3.1 (96 MHz ARM) counts to 1,098,681 for 10 s, but Arduino C on Teensy 3.1: (96 MHz ARM) counts to 95,835,923 for the same period. Author used milis() function for time measuring [18].

Arduino IDE is faster platform than MicroPython for neural networks development on edge devices. This research can be expanded with other development environments, especially with ESP-IDF. Some authors suggested that floating-point neural networks can be replaced with integer neural networks when cost is primary design concern, so it would be interesting to test performance and speed with integer neural networks [19, 20].

References

1. Li, H., Ota, K., Dong, M.: Learning IoT in edge: deep learning for the Internet of Things with edge computing. IEEE Network **32**(1), 96–101 (2018)
2. Espressif System: ESP32 Overview. https://www.espressif.com/en/products/hardware/esp32/overview. Accessed 15 June 2019
3. Biswas, S.B., Iqbal, M.T.: Solar water pumping system control using a low cost ESP32 microcontroller. In: IEEE Canadian Conference on Electrical & Computer Engineering, Quebec City (2018)
4. Abdullah, A.H., et al.: Development of ESP32-based Wi-Fi electronic nose system for monitoring LPG leakage at gas cylinder refurbish plant. In: 2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA), Kuching (2018)
5. Maier, A., Sharp, A., Vagapov, Y.: Comparative analysis and practical implementation of the ESP32 microcontroller module for the Internet of Things. In: 7th IEEE International Conference on Internet Technologies and Applications, Wrexham (2017)
6. Zidek, K., Janacova, D., Hosovsky, A., Pitel, J., Lazorik, P.: Data optimization for communication between wireless IoT devices and cloud platforms in production process. In: 3rd EAI International Conference on Management of Manufacturing Systems, Dubrovnik (2018)
7. Rosato, D., Masciadri, A.: Non-invasive monitoring system to detect siting people. In: Goodtechs, Bologna (2018)
8. Islam Chowdhuryy, M.H., Sultana, M., Ghosh, R., Ahamed, J.U., Mahmood, M.: AI assisted portable ECG for fast and patient specific diagnosis. In: International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering, Rajshahi (2018)
9. Fernoaga, V.P., Stelea, G.-A., Balan, A., Sandu, F.: OCR-based solution for the integration of legacy and-or non-electric counters in cloud smart grids. In: IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME), Iași (2018)
10. Komarek, A., Pavlik, J., Mercl, L., Sobeslav, V.: Hardware layer of ambient intelligence environment implementation. In: Nguyen, N.T., Papadopoulos, G.A., Jędrzejowicz, P., Trawiński, B., Vossen, G. (eds.) ICCCI 2017. LNCS (LNAI), vol. 10449, pp. 325–334. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67077-5_31
11. Chand, G., Ali, M., Barmada, B., Liesaputra, V., Ramirez-Prado, G.: Tracking a person's behaviour in a smart house. In: Liu, X., et al. (eds.) ICSOC 2018. LNCS, vol. 11434, pp. 241–252. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17642-6_21
12. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: MobileNetV2: inverted residuals and linear bottlenecks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City (2018)
13. Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multi-task cascaded convolutional networks. IEEE Signal Process. Lett. **23**(10), 1499–1503 (2016)
14. Kokoulin, A.N., Tur, A.I., Yuzhakov, A.A., Knyazev, A.I.: Hierarchical convolutional neural network architecture in distributed facial recognition system. In: IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Saint Petersburg and Moscow (2019)
15. Teerapittayanon, S., McDanel, B., Kung, H.: Distributed deep neural networks over the cloud, the edge and end devices. In: IEEE 37th International Conference on Distributed Computing Systems (ICDCS) (2017)
16. Zhang, Y., Suda, N., Lai, L., Chandra, V.: Hello edge: keyword spotting on microcontrollers. arXiv preprint arXiv:1711.07128 (2017)

17. Lai, L., Suda, N., Chandra, V.: CMSIS-NN: efficient neural network kernels for arm cortex-M CPUs. The Computing Research Repository (2018)
18. George Robotics Limited: Performance, 1 June 2014. https://github.com/micropython/micropython/wiki/Performance. Accessed 1 Sept 2019
19. Behan, T., Liao, Z., Zhao, L.: Integer Neural Networks On Embedded Systems. In: Recent Advances in Technologies. IntechOpen (2009)
20. Wang, N., Choi, J., Brand, D., Chen, C.-Y., Gopalakrishnan, K.: Training deep neural networks with 8-bit floating point numbers. In: 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal (2018)

Analysis of Frameworks for Traffic Agent Simulations

Nedaa Baker Al Barghuthi(✉) and Madeleine Togher(✉)

Higher Colleges of Technology, Sharjah, United Arab Emirates
{Nedaa.Albarghuthi,mtogher}@hct.ac.ae

Abstract. Traffic jams and congestion are a global challenge for smart cities. Digital transformation is touching all aspects of modern-day life utilities, from energy to health services and from education to smart mobility. A smart city aims to cut carbon dioxide emissions while saving lives by reducing driver or passenger trip time, traffic congestion and accidents. Traffic simulations and advanced traffic management systems attempt to address the urbanization infrastructure, sustainability, and mobility issues. Many road infrastructures are decades old and are therefore prone to an excess of frequent and recurrent traffic bottlenecks. This could be attributed to an increase in driver numbers and higher travel demands, which at the same time is increasing in the overall travel duration, air pollution and road accidents. The drivers of the United States alone travel 4.8 trillion kilometers per year. A smart city targets the efficient and sustainable use of its resources and services by using intelligent and interactive management. With the rapid growth of population density, the use of private cars and public transport, scientists thrive on trying to reduce urban traffic congestion in order to efficiently facilitate access to work, education, emergency services, entertainment and other services. For this purpose, graph algorithms and multi-agent system simulations are used for traffic optimization. While urban traffic congestions cannot be modeling eliminated, several frameworks for traffic simulations were developed to reduce traffic congestion. This paper will investigate and provide a comprehensive review of several frameworks used in these traffic agent simulations that can be found in open literature. It will make a comparative analysis of these frameworks used.

Keywords: Multi-agent systems · Smart city · Traffic management · Traffic optimization · Traffic simulations · Graph algorithm

1 Introduction

With the present and forecasted increase of urban traffic in high-density population cities, dense traffic is one of the most critical problems that the road infrastructure faces in supporting the transportation of humans and goods. This may cause several urban and environmental issues such as traffic jams, road accidents, and congestion. These factors are critical if the transportation department decides to extend the road capacity in an already crowded and built up an environment with high traffic volumes. The statistics show that the share of the urban population in the Gulf Cooperation Council (GCC) region will reside in an urban setting.

The GCC's population is estimated to grow to 53.4 m by 2020, a 33% increase over 12 years, which will require considerable investment in infrastructure, including transport management [1]. The urban population in the GCC has shown a steady average rate of increase of 0.68% between 2005 and 2025, slowing down to an average rate of increase of 0.53% between 2010 and 2030 (See Fig. 1). The most substantial rate of increase is currently taking place between the limit years of 2015 and 2020 and is predicted to have increased to the rate of 1.3% by 2020.

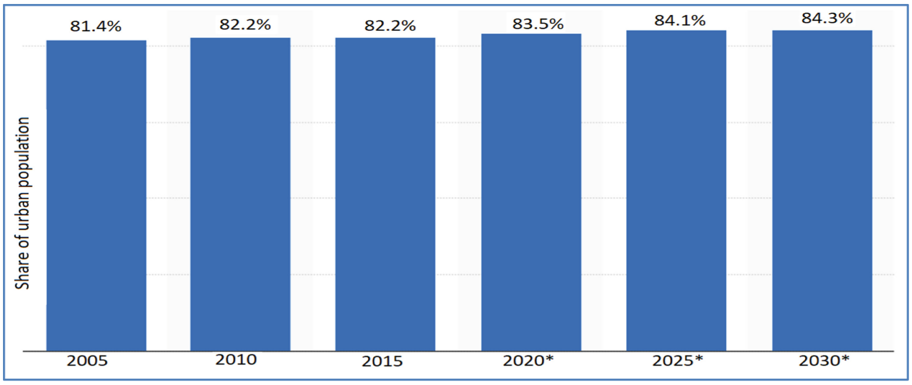


Fig. 1. Share of the urban population in the Gulf Cooperation Council region from 2005 to 2030 [1]

The following chart (See Fig. 2), shows the number of roadworthy vehicles that were in use in the Gulf Cooperation Council region in 2015 by country (in millions) [2]. Saudi Arabia owned the highest number of road vehicles at 6.6 million in 2015 with over three

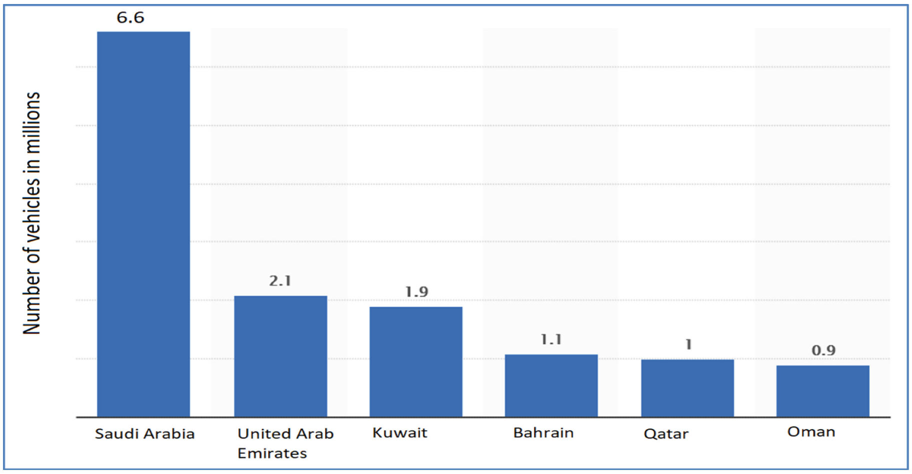


Fig. 2. The number of vehicles in use in the Gulf Cooperation Council region in 2015, by country (in millions) [2]

times more vehicles on the road than in United Arab Emirates (UAE) and over seven times more vehicles than Oman. Bahrain, Qatar, and Oman use a very similar number of vehicles which is around one million, while UAE and Kuwait used around 2 million vehicles.

2 Multi-agent Traffic Simulation in Large Cities

In large cities, the modeling and simulation complexity needs to be fully understood before a proper traffic evacuation protocol can be established [9]. The authors in [3] have tried to assess the impact of a tramway extension plan to cross the intersection for a very large and real city. They proposed a new simulation framework called Multi-Agent-Based Traffic and Environment Simulator (MATES). This framework was used to study the effectiveness of incorporating an extension plan for Okayama city, which is the capital of Okayama Prefecture in the Chugoku region of Japan. The researchers have attempted to provide a detailed and qualitative analysis of the impact of the policy on vehicle traffic. The paper proposed that simulations were carried out using an open-source simulator (MATES). The simulator was utilized as a continuous road model of vehicle traffic. It was used to simulate mixed traffic containing different types of transportation such as cars, trams, and pedestrians. The simulation attempted to use a model that could match the reality quite closely. Firstly, the authors developed a car traffic simulator and after that, authors installed a tram and pedestrian agent model into the developed simulator and set the interaction rules among these agents.

The experimental simulations were carried out during a busy holiday period between the hours of 13:00 to 14:00 h which is known to have the highest traffic volume. Four separate baseline scenarios have been set up and simulated before being compared by the authors in [3]. Scenario 1 had no tramway extension or pedestrians included. Scenario 2 had no tramway extension, with pedestrians included (current status). Scenario 3 included a tramway extension and pedestrians (with a green light for tram: 32 s, one time per cycle) while Scenario 4 included a tramway extension and pedestrians (green light for tram: 32 s, two times per cycle).

The screenshot of the MATES simulator (see Fig. 3) enables the representation of a tram interaction with traffic intersection into a 2D color geometry view. The tram is represented by long black rectangles. Each passenger vehicle is represented by a red rectangle and road pedestrians are represented by green circles. In this study, the interactions between trams, vehicles, and pedestrians were conducted as additional simulation scenarios. During the simulation, the drivers of the vehicles were given a choice to detour the congested tramline areas.

The outputs of these simulations have shown that detouring can be used as a positive solution to reduce any negative impact that a tramline extension might incur. The detour of car behavior demonstrates that as the ratio of cars taking detours increases, the travel time decreases at a steady rate (see Fig. 4). With every additional 5% of cars up to 20% that take a detour, the average rate of decrease demonstrates that the normalized mean link travel time will decrease by approximately 0.049.

Furthermore, authors in [4] discussed the possibilities of studying behavioral activities among different transportation agents in a large city. The potential requirement of

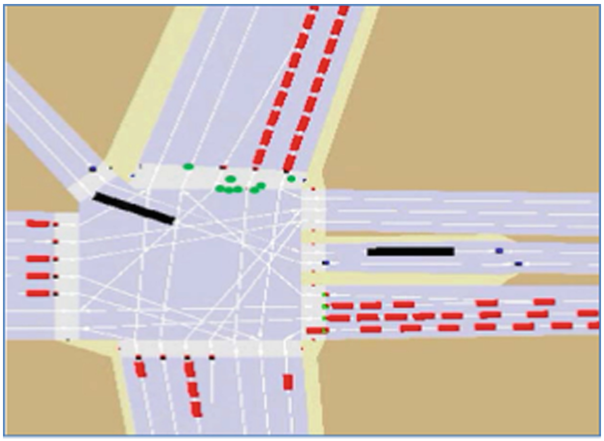


Fig. 3. Tram intersection modeling in MATES Simulation [3] (Color figure online)

extending and expanding the traffic volume in a large city first needs to test the effectiveness of these changes. These extended plans require experimental, practical implementation, simulation, and behavior performance monitoring before any road expansion can take place. The authors have proposed a framework of a multimodal multi-agent (MUST) simulator to simulate behavioral activities of a large volume of agents in a large city. In this research, the authors have used commercial simulation software called Traffic Simulator. This simulator has a high-performance database engine with a friendly user interface.

There are several different managers involved in this simulator software, such as the Configuration Manager, Environment Manager, Agent Manager Components, Routing Manager, and Visualization Manager. Configuration Manager is used importantly to define the overall environment. It contains agents (humans, trains or buses), frequency of the agents based on the amount of time as well as the initialization of the individual agent behaviors [4, 10].

The Environment Manager is used to create the overall environment, such as the layout of the town or city, the traveling routes for agents, and any necessary rules for the simulation. The Agent Manager component is used to set the agents into motion while monitoring their movements. During agent initialization, agent data will be stored into the database with the agent being tracked and its movement and location data being inserted at various points in time. The Routing Manager will decide on how an agent moves between locations (finds the route which the agent should take). This component will use the inputs previous stored by the agent manager to make route decisions. This service manager component is implemented on a machine in the cloud. When the service starts, it will generate network graphs for both buses and trains by using the scheduling timetables for each city. Each node represents a train station or bus stop with the edges representing the route between each node. The edge stores the properties of the timings of the services which interconnect the nodes. The Visualization Manager will display the visualization of the simulation results. This manager can retrieve the overall agent information, such as the locations of trains, buses, and humans from the database [4].

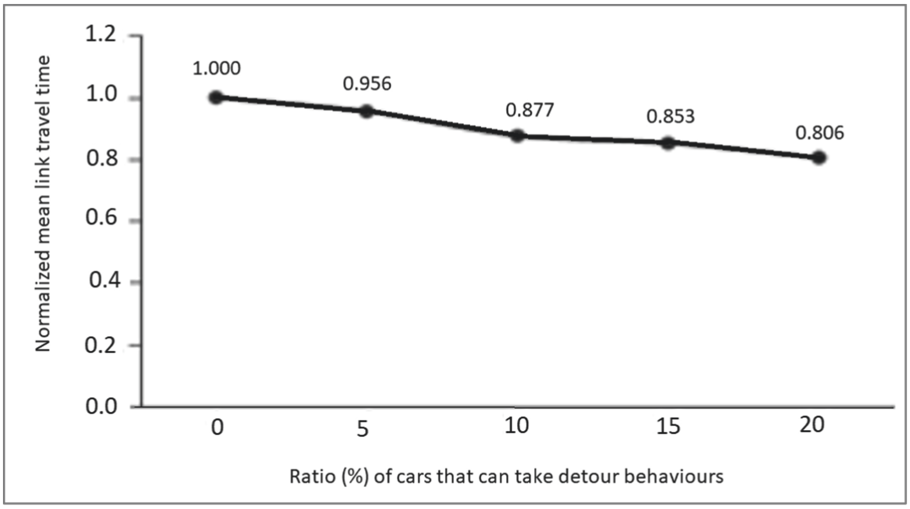


Fig. 4. Detour car behaviors [3]

A detailed description of the structure for a proposed framework has been discussed in this paper. A state representation model of multiple types of traffic objects can be seen in Fig. 5 where objects were presented into a two-dimensional colored box. A vehicle motion state is represented using a continuous blue line, and a purple box is used to represent a trained agent, while a yellow box is used to represent a bus agent. This simulator can import a custom number of maps with three types of agent selections (Human, Train, and Bus) [4].

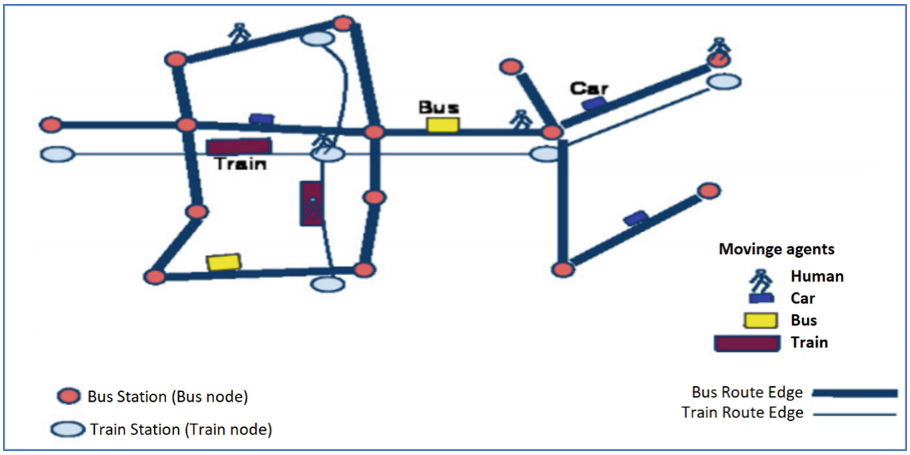


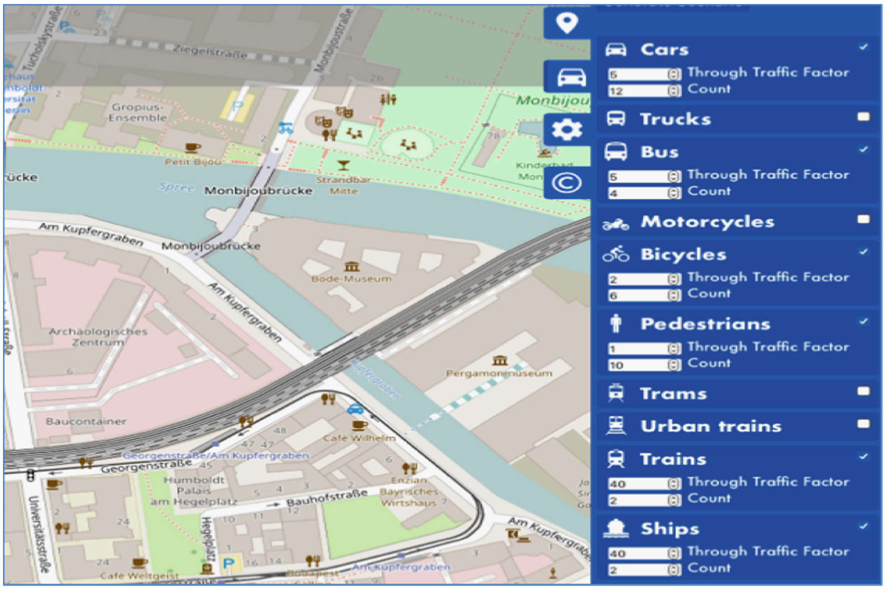
Fig. 5. Implemented scenario in traffic simulator [4] (Color figure online)

3 Real-Time Incident Detection Simulation

Authors in [5, 6, 11, 12] have addressed the challenge of monitoring traffic in real-time and reading measurements of vehicular traffic. By using VANET Vehicular Ad-Hoc Networks Technology, these measurements can be useful for incident detection on the traffic and perform the right action, such as sending an alert message to the right authorities such as police stations or ambulance to take the right action.

Furthermore, the authors in [5] have proposed a real-time incident detection framework which can gather vehicular traffic measurement on the roads efficiently. This proposed framework is via the design of two applications for two SAME & TOME sampling monitoring protocols in VANET networks. Two simulating tools were used in this experiment seeking to detect a real-time incident.

A road traffic simulator SUMO and communication network simulator NS2 were correlated with each other to model, analyze, and validate these collected measurements, as shown in Fig. 6. The measurements are based on extensive real GPS data including geographical coordinates, vehicle ID, real-time traffic speed on specific road and quality of GPS. These data were collected on the same highway road. This framework was designed to detect incidents of traffic. The behavior of two provided protocols was analyzed under exceptional traffic conditions, which were caused by an incident. This model was successful and validated through the simulators.



Cars ✓
5 Through Traffic Factor
12 Count
Trucks
Bus ✓
5 Through Traffic Factor
4 Count
Motorcycles
Bicycles ✓
2 Through Traffic Factor
6 Count
Pedestrians ✓
1 Through Traffic Factor
10 Count
Trams
Urban trains
Trains ✓
40 Through Traffic Factor
2 Count
Ships ✓
40 Through Traffic Factor
2 Count

Fig. 6. SUMO simulator interface [5]

The authors in [6] have used the proposed framework of [5] efficiently. This research has studied the possibility of using current information available on a network as early detection of incidents seeking to reduce traffic congestion. This network information might be from gathered GPS information from smartphones to detect the occurrence of traffic accidents. These gathered GPS data and video captures are analyzed by machine learning agents. These agents are trained to decide whether the traffic has a regular traffic flow, or there is an incident leading to congestion. The researchers have used a Multi-Agent System (MAS) model in the NetLogo simulator to simulate the behavior of vehicles as an independent agent as shown in Fig. 7. Experiment parameters of the simulations were set for light (25 cars), moderate (100 cars), and heavy (126 cars) traffic traversing the road in a given direction. The number of cars traveling in each direction, as well as the crash-chance, speed-up, and slow-down, were set.

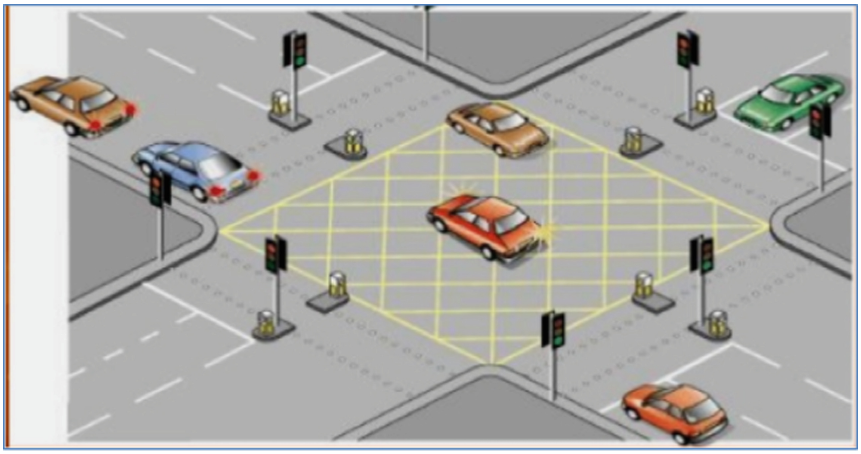


Fig. 7. NetLogo simulator interface [6]

Three types of classifier algorithms were used in this agent. The MAS model can analyze the behavior of both independent agents and system collectives. This simulator has a wizard interface where the researcher can insert symbols easily and control them. This interface can be useful to observe the behavior of vehicles in different scenarios. The simulator was used to generate sample data according to specific conditions. Training data sets were generated from individual simulations for each traffic level combination and corresponding test data sets with and without accidents. For training data, 200 simulations of 101 steps each with crash-chance set to zero for half of them and 100% for the other half generated 20,200 training samples. 1,010 test samples were then generated from 10 simulations of 101 steps, each with zero chance of a crash, and 1,000 samples were generated with 100% crash-chance. These sampled data were fed into these algorithms to determine the presence of the traffic accident. The simulator used a 2D to model the traffic, vehicles, and crashed vehicles. Two simulation models were used, normal mode with no crashed vehicles and congested mode with crashed vehicles.

4 Driver Behavior Agent Simulation

The behavior of the driver plays a vital role in road traffic, as introduced in [7, 8]. In this research, the authors discussed the behaviors of simulated drivers on traffic, particularly on different conditions in a dynamic environment. The researchers used ArchiSim (behavior traffic simulation tool) simulator and introduced the challenges in road traffic. This simulator software performed two scenarios and proposed an approach to simulate and solve their problems. The simulator software presented the object of the vehicle into a 3D notational presentation, as shown in Fig. 8.

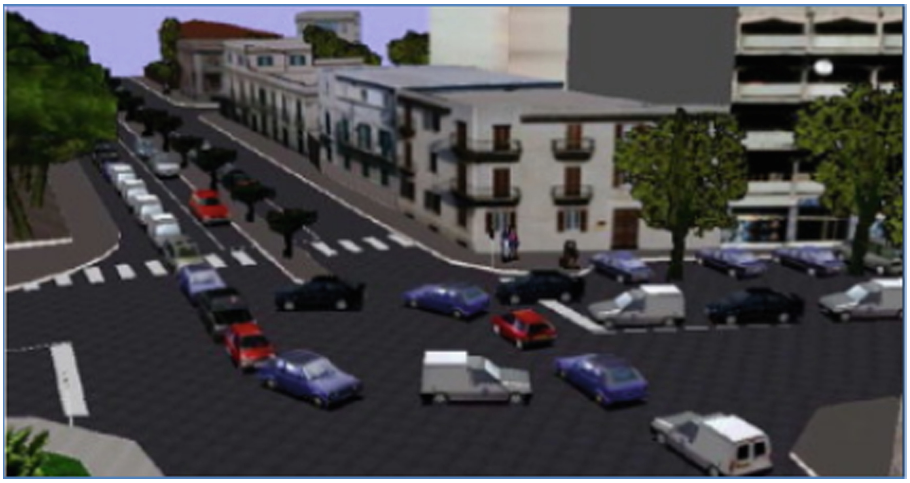


Fig. 8. ArchiSim simulator interface [7]

Each simulated vehicle was considered as a multi-agent using car-following laws and centralized scheduling techniques for each of the vehicles on the road. The agent's decision-making is based on the speed and position of the other vehicles in the area of the intersection or near the place.

The statistics from the simulated data were gathered from two geographical places with different conditions. The data were executed and collected by performing two experiments. The first was performed on an Italian intersection using the necessary driver behavior. It seeks to propose a solution of reducing the vehicle speed in particular areas such as intersections. The second was performed on an American roundabout using the proposed behavior approach. The authors in this research have successfully validated the results of the simulation comparing the simulated results with real traffic flow statistics within a similar situation. This paper shows the importance of reducing the speed of vehicles once reaching the area of an intersection to avoid causing an accident and to have a successful cross over of traffic.

A comparison of the different traffic simulators discussed during this paper can be found in Table 1. This comparison table will summarize the problem domain, the analysis type, the availability of the software, what the challenges are, and whether the simulator tool is worth pursuing or not amongst a number of others. The overall summary suggests that all of these models are worthwhile but they may be subject to limitations such as NetLogo which only allows one crash per simulation. Most of the simulators have some form of a challenge from the MUST software being complex to use or the difficulty in modeling intersections with ArchiSim. The MAS simulator requires more machine learning techniques to detect accidents ahead of passive traffic.

Table 1. Comparison of different traffic simulators

Simulator name	MATES continuous road model	NetLogo MAS simulator 5.2	ArchiSim	MUST Multi-modal multi-agent
Problem domain	Proposed a new framework to assess the impact of a tramway extension plan for a real city	Possibility of using a smart device with GPS enabled to detect traffic accidents	A new model for simulating a driver model in a framework using multi-agent techniques	Studying behavioral activities among different transportation agents in a large city
Analysis type	Qualitative analysis of the impact of the new policies on car traffic	Behavior analysis	Behavior analysis	Behavior analysis
Simulator language availability	Open Source Software (OSS)	Open Source Software (OSS)	Open Source Software (OSS)	Not provided to the public
Simulator object model	1D - car-following model 2D - discrete choice model for pedestrians	2-D	3-D	1-D
Simplicity (Yes/No)	Yes	Yes	Yes	Yes
Near to reality (Yes/No)	NA	NA	Yes	Yes
Proposed successful plan (Yes/No)	Yes – Tramway on existing car traffic would not be severe, and by extension, implied that the proposed framework could help stakeholders decide on expansion scenarios for tram users and private car owners	Yes – Most accurate results using medium-light traffic scenario, followed by a light-light scenario	Limitations in that the data for the validation of traffic simulation in an intersection are complicated to obtain	Yes

(*continued*)

Table 1. (*continued*)

Simulator name	MATES continuous road model	NetLogo MAS simulator 5.2	ArchiSim	MUST Multi-modal multi-agent
Simulation testing environment	Simple road environments for the road network in Okayama City	Simulates a monitored stretch of road	Intersections and roundabouts	Large volume of agents in a large city
Challenges	Adapting tramway on existing car traffic cross. Concerns that proposed solution would have a negative impact on car traffic flow	Machine learning techniques to detect the presence of a traffic accident from passive traffic of vehicles not involved in the incident	Intersections	Complex software to use
Worth pursuing or not (Limitation)	Yes	Yes The model only allows one crash per simulation	Yes Good simulator for driver behavior simulation	Yes

5 Conclusion

During this review of traffic models currently available, an attempt has been made to suggest which of the simulators should be considered as reliable and pursued to assist in urban traffic planning. The paper reviewed the models using the following group categories; Multi-Agent Traffic Simulation in Large Cities, Real-time Incident Detection Simulation and Driver Behavior Agent Simulation. From the models reviewed in this paper; it is recommended to use the open-source framework called the MATES model due to its following characteristics:

- The framework model has demonstrated promising results.
- It proposed a new framework to assess the impact of a tramway extension plan for a real city scenario.
- It uses qualitative analysis to assess the impact of new extension plans on car traffic.
- It is capable of using both a 1D - car-following model and 2D - a discrete choice model for pedestrians.
- This is a simple model to use and has already been successful in helping stakeholders decide on expansion scenarios for tram users and private car owners.

- During simulations, the drivers of vehicles were given a choice to detour the congested tramline areas, which makes it close to reality.
- Has been used in simple road environments for the road network in Okayama City.
- However, this model still needs additional work to be able to deal and cope with a larger volume of agents.

References

1. GCC: share of urban population 2030 | Statistic: Statista. https://www.statista.com/statistics/957368/gcc-urban-population/. Accessed 15 Sept 2019
2. GCC To 2020, The Gulf And Its People: Graphics.eiu.com (2019). http://graphics.eiu.com/upload/eb/Gulf2020part2.pdf. Accessed 08 Sept 2019
3. Fujii, H., Uchida, H., Yoshimura, S.: Agent-based simulation framework for mixed traffic of cars, pedestrians, and trams. Transp. Res. Part C Int. J. **85**, 234–248 (2017)
4. Pathania, D., Vissapraganda, B., Jain, N., Khare, A.: MUST: MUlti agent Simulation of multi-modal urban Traffic. In: Proceeding AAMAS 2013, Proceedings of the 2013 International Conference on Autonomous Agents & Multiagent Systems, Saint Paul Minnesota, USA, 06–10 May 2013, pp. 1397–1398 (2013)
5. Baiocchi, A., Cuomo, F., Felice, M., Fusco, G.: Vehicular ad-hoc networks sampling protocols for traffic monitoring and incident detection in intelligent transportation systems. Transp. Res. Part C Int. J. **56**, 177–194 (2015)
6. Thomas, R., Vidal, J.: Toward detecting accidents with already available passive traffic information. In: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (2017)
7. Doniec, A., Mandiau, R., Piechowiak, S., Espie, S.: A behavioral multi-agent model for road traffic simulation. Eng. Appl. Artif. Intell. Int. J. **21**(8), 1443–1454 (2008)
8. Yuan, S., Chun, S., Spinelli, B., Liu, Y., Zhang, H., Adam, N.: Traffic evacuation simulation based on a multi-level driving decision model. Transp. Res. Part C Int. J. **78**, 129–149 (2017)
9. Hsu, Y., Peeta, S.: Behaviour-consistent information-based network traffic control for evacuation operations. Transp. Res. Part C Int. J. **48**, 339–359 (2014)
10. Hadfi, R., Ito, T.: Multilayered multiagent system for traffic simulation. In: Proceeding AAMAS 2016, Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, Singapore, 09–13 May 2016, pp. 1474–1476 (2016)
11. Ngoduy, D., Jia, D.: Enhanced cooperative car-following traffic model with the combination of V2V and V2I communication. Transp. Res. Part B Int. J. **90**, 172–191 (2016)
12. Gyeruaym, M., Billot, R., El Faouzi, N., Monteil, J., Armetta, F., Hassa, S.: How to assess the benefits of connected vehicles? A simulation framework for the design of cooperative traffic management strategies. Transp. Res. Part C Int. J. **67**, 266–279 (2016)

Blockchain-Based Authentication Approach for Securing Transportation System

Maher Salem(✉)

Computer Information Science, Higher Colleges of Technology, Al Ain, United Arab Emirates
msalem1@hct.ac.ae

Abstract. Connected vehicles, in the transportation system and cloud-based infrastructure (all together are enterprise IoT infrastructure), communicate with each other to obtain services and perform updates via the internet. However, this will emerge new vulnerabilities which in turn will emerge attack vectors like Advanced Persistent Threat attack (APT) to gain unauthorized access to the transportation system. APT causes disruption and data loss as well as integrity violations. This leads to overhead the network and makes it harder to verify each and every transaction between the vehicles. Because of these issues and other security-related shortages like processing time, the need for a solution that specifies all requirements and implementations to be protected against APT attacks and any zero-day attack or any unknown hidden threat emerged from the new vehicle communications, is becoming essential. This paper is to utilize Blockchain technology to secure and protect the transportation system. It will propose an authentication approach using a random number generator to verify vehicle identity and authority, improve the security, integrity, and immutability of the transportation system. An experimental scenario has been also demonstrated to clearly explain the new authentication method. It shows that it is an effective approach to increase security defense lines and provide a secure infrastructure for the transportation system.

Keywords: Blockchain · IoT · Security · Transportation systems · Authentication

1 Introduction

Connected vehicles in the transportation system is a major item of the IoT infrastructure. It shows a rapid growth in utilizing information technology to control and govern the overall system. Vehicular Ad-Hoc Network (VANET) has become an active area of research, standardization, and development because it has tremendous potential to improve vehicle and road safety, traffic efficiency, and convenience as well as comfort to both drivers and passengers [1]. In intelligent transportation systems, each vehicle takes on the role of sender, receiver, and router [2] to broadcast information to the vehicular network or transportation agency, which then uses the information to ensure safe, free-flow of traffic. For communication to occur between vehicles and Road Side Units (RSUs), vehicles must have a radio interface or On-Board Unit (OBU) that enables short-range wireless ad hoc networks to be formed [3]. Vehicles must also be fitted with hardware that permits detailed position information such as Global Positioning System (GPS) or a simply a telematics unit such as the one integrated into the connected gateway. Fixed RSUs, which are connected to the backbone network, must be in place to facilitate communication. The number and distribution of roadside units is dependent on the communication protocol is to be used. In short, vehicles are communicating with each other's and with the RSU as the infrastructure part of the transportation system. Where the RSU communicates with the cloud infrastructure to provide authentication, secure data exchange and other services to the vehicles. Figure 1 shows an overview of the connected vehicles in the IoT infrastructure.

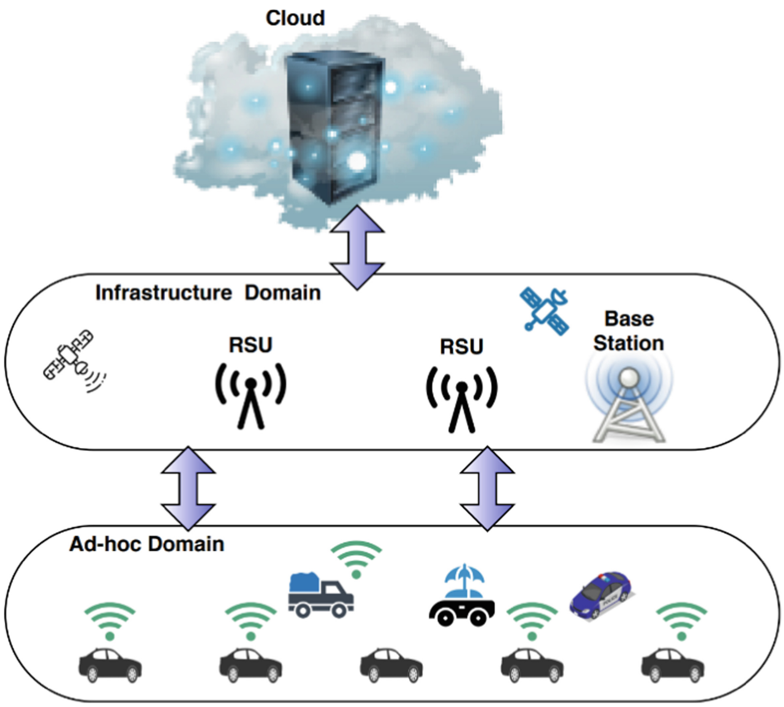


Fig. 1. Overview of connected vehicles in the IoT infrastructure

The communication between these elements in figure one relies on the authentication to avoid any unauthorized access and validate data integrity between all elements in the IoT infrastructure. Vehicles communicate with neighboring vehicles using On-Board Units (OBUs) form an ad-hoc network that allows communications in a distributed manner. One of the main objectives of the VANET is to communicate with other vehicles using safety messages to report events, such as accident information, safety warnings, etc. Failure in accuracy or time-critical information might lead to collateral damage to drivers and neighboring vehicles. So it is necessary to periodically authenticate nodes and verify their membership in the IoT [4].

The vehicle communicates with the RSU using a wireless channel as IEEE 802.11p, which use a wireless access in vehicular environments (WAVE). This includes data exchange between high-speed vehicles and between the vehicles and the roadside infrastructure. However, the authentication process between the vehicle and the RSU is not well defined and some approaches like [5] use VANET to authenticate new vehicle with the RSU by using a Trust Authority (TA), State Level Trust Authority (STA) and City Trust Authority (CTA). By expanding the authentication process, some weakness will emerge and allow hackers to exploit them and perform an APT attack to gain unauthorized access to the transportation system. The APT as given by US National Institute of Standards and Technology (NIST) is defined as [6]: "An adversary that possesses sophisticated levels of expertise and significant resources which allow it to create opportunities to achieve its objectives by using multiple attack vectors (e.g., cyber, physical, and deception). These objectives typically include establishing and extending footholds within the information technology infrastructure of the targeted organizations for purposes of exfiltrating information, undermining or impeding critical aspects of a mission, program, or organization; or positioning itself to carry out these objectives in the future. The advanced persistent threat: (i) pursues its objectives repeatedly over an extended period of time; (ii) adapts to defenders' efforts to resist it; and (iii) is determined to maintain the level of interaction needed to execute its objectives".

To overcome any shortages or weaknesses due to authentication issues in the transportation system, this article handles the authentication issue in the transportation system by using the Blockchain technology to preserve the authentication and protect the data between all entities in the IoT based transportation system. The RSUs will be considered nodes in the Blockchain network, the server farm(s) will be considered the master node, and all other members need to be authenticated by the node, so, in this case the APT attack will be mitigated and hence unauthorized access will be eliminated.

The function of a Blockchain is straightforward. As it is a peer-to-peer network, a user needs to start a transaction. Once done, a block is allocated to the said transaction. The transaction block is also broadcasted to the network, and all the nodes in the network get the said information. The block is then mined and validated. It is also added to the chain, followed by a successful transaction [14].

The rest of this article is organized as follow. Section 2 presents some related work to the same topic. Section 3 illustrates the method of using Blockchain to overcome the before mentioned problems. Section 4 present some scenario and discusses the rationale. Section 5 shows more details of the approach, and Sect. 6 concludes and show a future improvement on the idea.

2 Related Works

Dorri et al. [7] have proposed a scalable lightweight Blockchain architecture to protect the privacy of users and to increase the security of the vehicular ecosystem. Wireless remote software updates and other emerging services such as dynamic vehicle insurance fees are used to illustrate the efficacy of the proposed security architecture. However, the authentication of the nodes and members is not discussed and protecting the system of any kind of attacks is also not included. Leiding et al. [8], proposed the Blockchain technology for vehicular ad-hoc network (VANET). They have combined Ethereum' Blockchain based smart contracts system with vehicle ad-hoc network. They divided their proposal into two applications, mandatory applications (traffic regulation, vehicle tax, vehicle insurance) and optional applications (applications which provides information and updates on traffic jams and weather forecasts) of vehicles. Shrestha et al. [4] have proposed a secure message dissemination concept using Blockchain in VANET to protect data exchange in the transportation system. The idea is very effective in securing data and the trustworthiness of the nodes but still not clear about the authenticity of each member and how to verify and identify any rogue member in the network.

Kang et al. propose a two-stage soft security enhancement solution: (i) miner selection and (ii) block verification to overcome the challenge of data exchange and the reputation of the miner in the Blockchain [9]. In the first stage, design a reputation-based voting scheme to ensure secure miner selection. This scheme evaluates candidates' reputation using both historical interactions and recommended opinions from other vehicles. The candidates with high reputation are selected to be active miners and standby miners. Where in the second stage, they adopt the contract theory to model the interactions between active miners and standby miners, where block verification security and delay are taken into consideration. This idea is to get a high reputation miner and avoid collision, however, authentication is still negotiable point.

The same idea has been also proposed by Yang et al. [10]. They propose a decentralized trust management system in vehicular networks based on Blockchain techniques. In this system, vehicles can validate the received messages from neighboring vehicles using Bayesian Inference Model. Based on the validation result, the vehicle will generate a rating for each message source vehicle. With the ratings uploaded from vehicles, RSUs calculate the trust value offsets of involved vehicles and pack these data into a "block." Then, each RSU will try to add their "blocks" to the trust Blockchain which is maintained by all the RSUs. By employing the joint proof-of-work (PoW) and proof-of-stake consensus mechanism, the more total value of offsets (stake) is in the block, the easier RSU can find the nonce for the hash function (PoW). In this way, all RSUs collaboratively maintain an updated, reliable, and consistent trust Blockchain. The method is validating the messages and not the identity of the members or nods.

The most interesting article that was the trigger for inspiration of this article, is proposed by Li et al. [11]. In their proposed method, they assign a unique ID for each individual device and record them into the Blockchain, so that they can authenticate each other without a central authority. They also design a data protection mechanism into the Blockchain where any state changes of the data can be detected immediately. The previous work has been improved in this article and concentrates on using the Blockchain to verify identity in a random mechanism to avoid any unauthorized access. Guan et al. [12] have presented an AuthLedger a domain authentication scheme based on Blockchain technology. They proposed a domain authentication scheme to reduce the level of trust in CAs, implemented a system using Ethereum smart contract, and evaluated security and performance of the proposed system. This approach. In AuthLedger, Domain initiates an identity binding request in the Blockchain, which Validates authority via full nodes verify the requests. Then a domain updated a trusted CAs in the Blockchain. The domain name sends a CA list that is trusted by Blockchain network to complete certificate issuance process. So, client initiates a secure connection through the browser-plug-in to obtain the certificate own by the domain name. as a result, browser then requests trusted CAs list from Blockchain and compare the validity of the information obtained.

Lin et al. [13] design a cryptographic membership authentication scheme (i.e., authenticating graph data) to support Blockchain-based identity management systems (BIMS). They, specifically, introduced a new transitively closed undirected graph authentication (TCUGA) scheme, which only needs to use node signatures.

And there are many other approaches handle and discuss the issue of authentication in the IoT. Most of proposed work focus on the message or data transactions to validate its integrity and authenticity as well as immunity. In this article, the major idea is before verifying the data integrity or the transaction authenticity, it must be first verifying that the nodes in the Blockchain infrastructure are still authentic, and all communications are preserving integrity. Then ensure that no rogue or impersonating cases are allowed to gain unauthorized access and perform any of APT attacks.

3 Proposed Methodology

By following the same concept in [14], the main idea in this article is to improve the authentication mechanism in the transportation system by utilizing Blockchain technology. However, the Blockchain nodes are the main RSU units in the infrastructure, Server farm is the master node, where all vehicles are communicating with the RSU to get authenticated and to exchange data. Since Sect. 2 has discussed how to use RSU for a secure and immutable data exchange, here the authentication will take place and will be discussed using Blockchain. Figure 2 illustrates the overall architecture for the proposed Blockchain-based transportation system (some acronyms call it Internet of Vehicle, IoV).

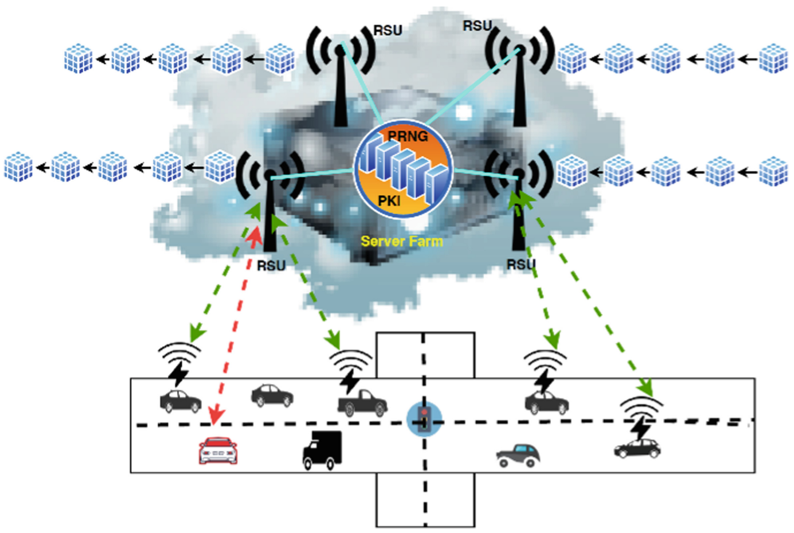


Fig. 2. Proposed blockchain-based transportation system (Color figure online)

The proposed idea in [15] had discussed the trust issue between the vehicles and the infrastructure and clarified it in an effective approach, this article will discuss only the authentication issue between the vehicles and the infrastructure and how the Blockchain will verify the identity of each vehicle to protect the transportation system against attacks such as APT.

As shown in Fig. 2, the Blockchain layer consists of RSU nodes and the server farm, master node, of the transportation system. By distributing a ledger among all members of the network, Blockchain authentication eliminates someone from maliciously altering the ledger. All members in the transportation systems have key pairs (K_{public}, $K_{private}$), whereas public keys are stored in the Public Key Infrastructure (PKI) in the server farm of the transportation system. Other major function of the server farm is to generate random numbers irregularly and disseminate them – signed and encrypted – among the Blockchain nodes. Each Node will then verify the identity of the server farm using its public key and decrypt the random number using their private keys. The same process will be then repeated between the nodes and the vehicles in the transportation system. Having this process done, all members and nodes in the transportation system are identified and hence they can perform some transactions. The following flowchart in Fig. 3 shows the process of authentication.

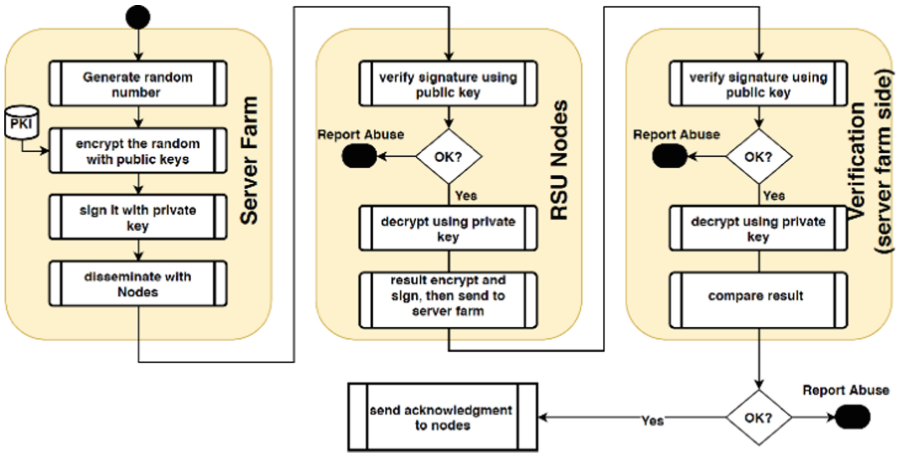

Server Farm
Generate random number
PKI
encrypt the random with public keys
sign it with private key
disseminate with Nodes
RSU Nodes
verify signature using public key
Report Abuse
OK?
Yes
decrypt using private key
result encrypt and sign, then send to server farm
Verification (server farm side)
verify signature using public key
Report Abuse
OK?
Yes
decrypt using private key
compare result
Report Abuse
OK?
Yes
send acknowledgment to nodes

Fig. 3. Blockchain-based authentication process

Figure 3 clearly presents the verification process between the nodes in the Blockchain. In the first step, the server farm generates a random number and sign it using its private key as follows:

$$S := E\big(H(Rand), K_{Server\_private}\big) \tag{1}$$

Where S is the signature, E is the encryption method used, H is the hash function, $Rand$ is the random number, and K is the key. The random number will be then encrypted in the second step by using the node's public key as:

$$C = E\big(Rand, K_{Node\_public}\big) \tag{2}$$

Where C is the cipher text or the encrypted message. In the last step, the server farm will send the pair *(S, C)* to the intended node for verification. The node in the Blockchain, in return, will verify the signature as:

$$S' := D\big(H(Rand), K_{Server\_public}\big) \tag{3}$$

Then, the node compares the hashes to verify the signature and decrypts the message to find out the random number as:

$$C' = D\big(Rand, K_{Node\_private}\big) \tag{4}$$

Where D is decryption, C' is the decrypted message, S' is the verified signature.

Now, if the two entities end up with having $\left(S = S'\right)$ and $\left(C = C'\right)$ then the authentication is successfully achieved. To fulfill this goal, the node will perform the same processes in (1) and (2) and the server farm will also perform the same process in (3) and (4), of course each with its private and public keys. After performing (4) on the server side, if the random number is the same as generated by the server, then it sends an acknowledgment message to the node.

Throughout the authentication process shown in Fig. 3, if any mismatch occurs from any side, an abuse report will be generated and sent to the authority office of the transportation system to consider it as an attempt to gain unauthorized access. Since this is out of the article scope, the abuse report is a temporary solution in this case. However, some other articles like [15] and [9] propose some solution to this problem.

When the verification process between the server and the nodes in the Blockchain ends successfully, the node will then repeat the process with the communicating vehicles and verify their identity to avoid any attempt to attack the transportation system.

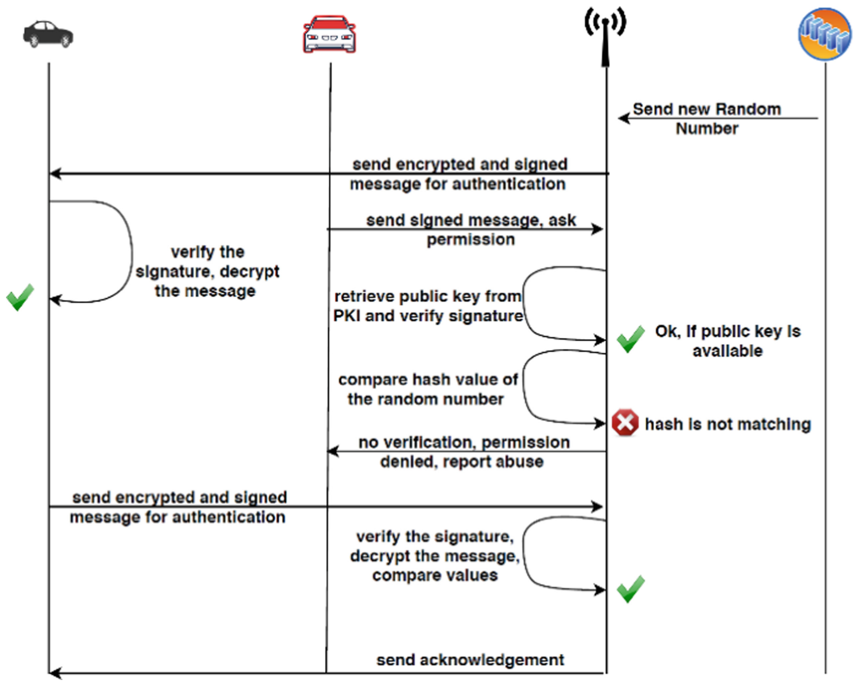
In the next section, a certain scenario will be explained and discussed.

4 Experimental Scenario

Figure 2 is the reference to the following scenario. In Fig. 2, assume the RSU on the left bottom side is communicating with two vehicles (green arrows) and one suspicious vehicle is trying to communicate with the same RSU. The RSU receives from the Blockchain a new random number to keep verifying vehicles identity. The following message sequence diagram displays the process of identification between the RSU and the vehicles.

The scenario in Fig. 4 is just to illustrate the idea of detecting suspicious vehicle and consider it an attacker attempt to gain unauthorized access. Principally, the server of the transportation system stores all authorized vehicles public keys in the PKI and govern the Blockchain immutability using this PKI. Assuming a new vehicle is trying to gain access by sending request to the RSU, then the RSU will try to authenticate it and verifies its identity.

Hence the RSU needs to encrypt the random number with the new vehicle public key. But the public key of this new vehicle is not within the PKI server, so the RSU will reject the request and report an abuse. Even that the public key of the new vehicle is stored in the PKI in some way, then the RSU will verify the signature and compare the



Send new Random Number
send encrypted and signed message for authentication
send signed message, ask permission
verify the signature, decrypt the message
retrieve public key from PKI and verify signature
Ok, if public key is available
compare hash value of the random number
hash is not matching
no verification, permission denied, report abuse
send encrypted and signed message for authentication
verify the signature, decrypt the message, compare values
send acknowledgement

Fig. 4. Message sequence diagram for a suspicious scenario

hash value (that the new vehicle encrypted using its private key) to check if it matches the hash value of the random number. Since the new vehicle has no clue about the random number, the hashes will be different, so RSU will reject and report abuse.

Even if the RSU signs and encrypts the random number and sends it to the new vehicle for verification. The new vehicle does not have the public key for the RSU and hence is not able to verify the signature or encrypt any message from the RSU, which in return will report an abuse.

To get in more discussion details about the proposed idea, the next section discusses some key aspects and shows pros and cons of the proposed approach.

5 Discussion

In the transportation systems, which are part of the enterprise IoT environment, all vehicles are communicating to each other and to the infrastructure, so called RSU. The medium, if communication is mostly a wireless communication channel, is using different protocols like the WiFi or LTE, etc. When the Blockchain technology is deployed to provide integrity, immutability, and data security in the transportation system, it is very essential to think about protecting the transportation system from any attack that attempts to gain unauthorized access on the network. If the transportation system is governed by a Blockchain technology, then it should be decentralized and distributed ledger all over the country and all RSUs are considered nodes. “A distributed ledger is a database that is consensually shared and synchronized across multiple sites, institutions or geographies.

It allows transactions to have public “witnesses,” thereby making a cyberattack more difficult. The participant at each node of the network can access the recordings shared across that network and can own an identical copy of it”.

Which means all members (things, vehicles, etc.) participating in this network should be verified and authenticated. The proposed idea in this article is to provide a clarification on how to deploy a simple and cost-effective authentication method using the Blockchain nodes to avoid any compromise or violation on the transportation system. Hence, the process of authentication in the proposed Blockchain-based transportation system, can be summarized as:

- Public keys for all authorized members should be stored in the PKI by the transportation authority.
- Master node generates a random number and communicate with the RSUs for authentication.
- Authentication process between nodes and master nodes is done.
- Nodes communicate with members for authentication.
- Any error during the authentication process consider the member is suspicious and report it to the authority office.

The major components of the Blockchain such as the smart contract, consensus algorithm, etc. have been avoided here because the focus was to use the Blockchain for authentication. Moreover, this article assumes the Blockchain is already existed and in operation. However, we agree that using a master node in the Blockchain is not the perfect solution since it needs time and cost as well as it may consume the resources of the network. We prioritize the authentication process over the cost and time. On the other hand, the attacker may try to steal data from the communication channel or the member storage; the proposed idea is not handling such type of attacks. The random number generator algorithm could be PRNG or TRNG. The former needs a seed state and hence the numbers are deterministic and efficient. But the latter is a hardware solution that may not work for this approach. Therefore, this article omits the focus on generating a random number.

This article has described the process in an experimental scenario to show the feasibility of the approach. However, it will in the next improvement consider a practical implementation of the concept and hence measure the approach with several performance parameters to emphasize on its feasibility and deployment.

6 Conclusion and Future Work

This paper proposes an authentication method by utilizing Blockchain technology to secure and protect the transportation system. It uses random number generator to verify the identity of all members in the transportation system and their authority. The master node has been considered in the Blockchain and it is the responsible for generating the random number and the dissemination process. An experimental scenario has been also demonstrated to clearly explain the new authentication method. It shows that, it is an effective approach to increase the security defense lines and provide a secure infrastructure for the transportation system.

This approach will be extended to come up with a framework and a state-of-the-art solution for the authentication issue in the IoT, specifically in the United Arab Emirates. It will be further extended to include a smart contract, node reputation voting system, consensus layer, and other related topic to provide a full immunity system against several types of attacks and to enhance the monitoring and surveillance systems.

References

1. Zeadally, S., Hunt, R., Chen, Y.-S., Irwin, A., Hassan, A.: Vehicular ad hoc networks (VANETS): status, results, and challenges. Telecommun. Syst. **50**(4), 217–241 (2012)
2. Sun, J., Zhang, C., Fang, Y.: An id-based framework achieving privacy and non-repudiation in vehicular ad hoc networks. In: MILCOM 2007 – IEEE Military Communications Conference, Orlando, pp. 1–7 (2007)
3. Gillani, S., Shahzad, F., Qayyum, A., Mehmood, R.: A survey on security in vehicular ad hoc networks. In: Berbineau, M., et al. (eds.) Nets4Cars/Nets4Trains 2013. LNCS, vol. 7865, pp. 59–74. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37974-1_5
4. Shrestha, R., Bajracharya, R., Shrestha, A.P., Nam, S.Y.: A new type of blockchain for secure message exchange in VANET. Digit. Commun. Netw. (2019)
5. Chaurasia, B., Verma, S.: Infrastructure based Authentication in VANETs. Int. J. Multimed. Ubiquit. Eng. **6**(2), 41–54 (2011)
6. NIST: Managing Information Security Risk: Organization, Mission, and Information System View. SR800-39. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-39.pdf
7. Dorri, A., Steger, M., Kanhere, S.S., Jurdak, R.: Blockchain: a distributed solution to automotive security and privacy. IEEE Commun. Mag. **55**(12), 119–125 (2017)
8. Leiding, B., Memarmoshrefi, P., Hogrefe, D.: Self-managed and blockchain-based vehicular ad-hoc networks. In: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp 2016), Association for Computing Machinery, New York, pp. 137–140 (2016)
9. Kang, J., Xiong, Z., Niyato, D., Ye, D., Kim, D.I., Zhao, J.: Toward secure blockchain-enabled internet of vehicles: optimizing consensus management using reputation and contract theory. IEEE Trans. Veh. Technol. **68**(3), 2906–2920 (2019)
10. Yang, Z., Yang, K., Lei, L., Zheng, K., Leung, V.C.M.: Blockchain-based decentralized trust management in vehicular networks. IEEE Internet Things J. **6**(2), 1495–1505 (2019)
11. Li, D., Peng, W., Deng, W., Gai, F.: A blockchain-based authentication and security mechanism for IoT. In: 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, pp. 1–6 (2018)
12. Guan, Z., Garba, A., Li, A., Chen, Z., Kaaniche, N.: AuthLedger: a novel blockchain-based domain name authentication scheme. In: Proceedings of the 5th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP, ISBN 978-989-758-359-9, pp. 345–352 (2019)
13. Lin, C., He, D., Huang, X., Khan, M.K., Choo, K.R.: A new transitively closed undirected graph authentication scheme for blockchain-based identity management systems. IEEE Access **6**, 28203–28212 (2018)
14. Salem, M., Mohammed, M., Rodan, A.: Security approach for in-vehicle networking using blockchain technology. In: Barolli, L., Xhafa, F., Khan, Z.A., Odhabi, H. (eds.) EIDWT 2019. LNDECT, vol. 29, pp. 504–515. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12839-5_47
15. Hammi, M.T., Hammi, B., Bellot, P., Serhrouchni, A.: Bubbles of Trust: a decentralized blockchain-based authentication system for IoT. Comput. Secur. **78**, 126–142 (2018)

A Classification Model for Modeling Online Articles

Rula Alhalaseh[1], Ali Rodan[2], and Azmi Alazzam[2](✉)

[1] Department of Information Technology, University of Jordan, Amman, Jordan
rula.alhalaseh@gmail.com
[2] Computer Information Science, Higher Colleges of Technology,
Al Ain Women's Campus, Al Ain, Abu Dhabi, UAE
{arodan,aalazzam}@hct.ac.ae

Abstract. Due to the constant evolvement of the web and the viral spread of online news on social media, predicting the popularity of a news article became a topic of interest to many categories of people ranging from marketing personnel to politicians. In this paper, we focus on comparing four classification algorithms on a dataset consisting of 39000 news articles taken from Mashable website. The articles were classified into two classes: Popular and not popular. Four different machine learning algorithms were used for classification of the data (KNN, Naïve bayes, Adaboost, and decision tree). Finally, the four classification methods were compared with each other.

Keywords: Ada Boost · KNN · Naïve Bayes · Decision Tree

1 Introduction

In recent years, with the viral evolvement of social media, the sharing, commenting on and reading of various kinds of articles, including news and articles of social or political nature, has become the center of people's daily entertainment. As tons of news, rumors and stories are published on a daily basis, there comes a need for predicting whether a news piece can go viral before it is published. Predicting news popularity became a trendy field of research for researchers, authors and advertisers to build their strategies as well as make it reach as many individuals as possible. This will also help in extracting and implementing the features contributing to a viral article outspread. Also some politicians are concerned of the influence of news articles on the population and the effects from spreading such news. In this paper, the dataset used is a real-world dataset taken from UCI Machine Learning Repository [1] that collected over than 39000 articles from Mashable [2] website.

The dataset has various informative features [3], we also intend to compare and analyze the performance of several machine learning algorithms to predict the popularity of news articles. Measurement of popularity is known as the number of times an article gets shared, liked and commented on. For the popularity measure we adopt a common binary task to classify the articles into popular and unpopular and then use the machine learning algorithms to build a classification model that can used to classify new articles based on some features.

There are two main prediction approaches to measure popularity [4].

The first approach is to use features that are only known and observed after publishing an article and the second approach does not use these features. The first approach is more common. An Example of the first approach can be found in [5] where the evolution of user generated content popularity is discussed. Another proposed methodology for predicting online contents popularity in a more precise manner rather than attempting to infer the possibility that a content will be popular can be found in [6].

The statistical analysis of the time of user reaction to a newly opened a discussion thread online which was made on the popular news website Slashdot [7, 8]. It also performed a characterization that enabled predicting intermediate and long-term user behavioral pattern with and acceptable result of precision.

Predicting the popularity of online content was elaborated in [9, 10]. In [11] the authors proposed a framework for modeling and predicting the popularity of online contents that aimed to infer the likelihood with which the content will be popular.

Since the prediction task is easier to implement, higher accuracies in prediction are often achieved. Popularity prediction of articles that do not use features which is not a commonly used approach as a low performance in prediction is expected. And moreover using the features as in the first approach are said to improve the content before having it published.

2 Literature Review

The work mentioned in [3] consists of a robust evaluation of five state of the art models for classification of around 39 thousand articles that were labeled and collected from Mashable website [2]. Experiments on Random forest in [3] conducted the best result having a discrimination power of 73% for binary classification.

A research to address the prediction task both as a regression and a classification problem as well as to predict the number of news tweets was discussed in [12]. The paper illustrated that even though predicting the exact number of tweets may have a high error percentage, there is a possibility for predicting ranges of popularity on news tweets with 84% overall accuracy. Furthermore, it considered four types of features (news source, category of the article, subjectivity language used, and names mentioned in the article) to predict the tweets number that has the article mentioned in them. Three popularity classes were studied which ranged between 1–20 tweets, 20–100 tweets and more than 100, discarding the articles with no tweets.

Another study that used news tweets proposed the passive aggressive algorithm to predict how many times a news link tweet will be retweeted [13]. The study discussed also the some social features like how many users are following the user tweeting, which can determine the number of times an article will be retweeted. Moreover, this research noticed that the number of urls and hashtags could also boost the tweet to make it get to as many people as possible.

Xuandong et al. researched the topic of predicting whether a mashable news article will be viral or not by addressing two dimensions of the problem: multidimensional classification and numerical [14]. They applied linear regression, polynomial regression, GAM with smoothing splines and Lasso to predict the exact amount of shares of a news article. In the paper, GAM with smoothing splines gave the best CV error which was 0.7649. In their paper, they used SVM, Random Forest and Bagging to predict popularity of the news, resulting into four categories for each news article and with Random forest giving best result of 50.4% accuracy in prediction.

The research work in [15] tested two binary classification tasks for prediction: popular and unpopular as well as appealing and non-appealing when compared to articles that were published on the same day used 10 English news outlets that related with one year. The paper used bag of words of the title and description, keywords and characteristics such as date of publishing combined with Support Vector Machine (SVM). The appealing task gave better accuracy results of 62–86% when compared with the popular and unpopular task which gave results ranging from 51–62%.

3 Methodology

In this section, the methodology of the classification algorithms is discussed. There are four classification algorithms that are discussed and implemented to classify the data collected from different articles. The goal of this study is to build a classification model with high accuracy that can be eventually used to predict the popularity of articles before a decision is made whether to publish them or not. The next four Subsect 3.1–3.6 will discuss and elaborate the methodology in details.

3.1 Dataset

The dataset obtained from [1] consists of over 39 thousand articles from Mashable [2] which is one of the largest well known news website. The data was retrieved and prepared by [3] on a 2 years period, from January, 7 2013 and January, 7 2015. Special occasion articles were discarded which was only a small portion of the dataset and did not follow the general structure of HTML as the processing of each occasion would require a specific parser. The collected data was donated to the UCI Machine Learning Repository [1] for public use. The processing and collection process of the Mashable [2] data in [3] was implemented in Python, while we are going to use Weka tool for our work. After preprocessing of the dataset the resulting articles were a total of 39 thousand data points with 60 features.

We summarized the work done on the Mashable [2] dataset before proceeding with the classification. The dataset classification considered 47 features in total that were extracted from html code. The features of the dataset are shown in Table 1 [3]. These features have different types: number-integer value, ration within [0, 1], a bool that can be either a 0 or 1 and a nominal value. Columns with (#) indicate the number of variables within each feature. The number of shares attribute which we will be working on in our paper was concluded in [3] by selecting a large list of characteristics that describe different aspects of the article and that were considered possibly relevant to influence the number of shares as we have the minimum, maximum and the average number of shares in various social networks in the dataset.

In this paper, a binary classifier is used. Two classes are considered popular and unpopular. If the article has more than 1400 shares it is considered as a popular article, and otherwise it is considered as unpopular article. Thus, the classification algorithm will use the existing data to predict the classes based on the 47 attributes of this data.

The data will be divided into two parts: two thirds will be used for building (training the model) and one third for validation in order to avoid over fitting.

In the next subsections we discuss the data labeling process and the different algorithms that were used in building the classification model.

Table 1. Statistical measures of the articles in Mashable dataset [3]

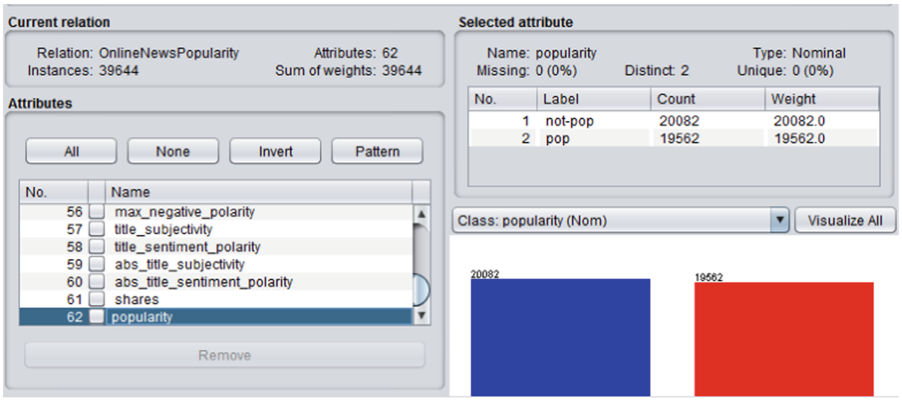
Feature	Type (#)	Feature	Type (#)
Words		**Words**	
-Number of words in the title	Num (1)	-Number of keywords	Num (1)
-Number of words in the article	Num(1)	-Worst keyword (min./avg./max. shares)	Num (3)
-Average word length	Num (1)	-Average keyword (min./avg./max. shares)	Num (3)
-Rate of non-stop words	ratio (1)	-Best keyword (min./avg./max. shares)	Num (1)
-Rate of unique words	ratio (1)	-Article category (Mashable data channel)	Nom(1)
-Rate of unique non-stop words	ratio (1)	**Natural Language Processing**	
Links		-Closeness to top 5 LDA topics	ratio (5)
-Number of links	Num (1)	-Title subjectivity	ratio (1)
-Number of Mashable article links	Num (1)	-Article text subjectivity score and its absolute difference of 0.5	ratio (2)
-Minimum, average and maximum number of shares of Links	Num(3)		
		-Title sentiment polarity	ratio (1)
Digital Media		-Rate of positive and negative words	ratio (2)
-Number of images	Num (1)	-Pos. words rate among non-neutral words	ratio (1)
-Number of Videos	Num (1)	-Neg. words rate among non-neutral words	ratio (1)
Time		-Polarity of positive words (min./avg./max.)	ratio (3)
-Day of the Week	Nom (1)	-Polarity of negative words (min./avg./max.)	ratio (3)
-Published on a weekend?	Bool(1)	-Article text polarity score and its absolute difference to 0.5	ratio (2)

Target	Type (#)
-Number of article Mashable shares	Num (1)

3.2 Process of Labeling Classes and Evaluating Results

We added a new attribute named popularity and modified the dataset with the condition that articles having more than 1400 shares are to be labeled as popular 'pop' and other than that will be labeled as 'not-pop'. Excel function applied was:

Function=IF(BI2>1400, "pop", "not-pop")
We then observe that the classes are balanced on Weka as in the figure (Fig. 1)



Current relation

Relation: OnlineNewsPopularity Attributes: 62
Instances: 39644 Sum of weights: 39644

Attributes

No.	Name
56	max_negative_polarity
57	title_subjectivity
58	title_sentiment_polarity
59	abs_title_subjectivity
60	abs_title_sentiment_polarity
61	shares
62	popularity

Selected attribute

Name: popularity Type: Nominal
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count	Weight
1	not-pop	20082	20082.0
2	pop	19562	19562.0

Class: popularity (Nom)

Fig. 1. Classified dataset based on popularity

The total number of attributes is 61 but we only used 47 as discussed before and also shown in Table 1. We used 10-fold cross-validation testing mode for all selected algorithms. 39644 class instances (rows) used in all algorithms.

3.3 AdaBoost

AdaBoost is short for "Adaptive Boosting", and defined as a machine learning meta-algorithm that was formulated by Yoav Freund and Robert Schapire [16]. Adaboost in conjunction with other learning algorithms are said to improve performance.

Weka AdaBoost M1 [20], which is a class for boosting nominal class classifier and can only tackle nominal class problems. Adaboost M1 was used in combination with J48 classifier.

The confusion matrix also known as an error matrix or a contingency table is a certain table layout which visualize the algorithm performance, every column stand for the instances in a forecasted class, while every row stands for the instances of the actual class. The confusion matrix for any classification algorithm is given by Table 2 [17].

Testing the models with just Accuracy and Sensitivity measures is not adequate to ensure that the classifications have reliable results, so we will use more measures for the model evaluation, which are as follow.

Table 2. Confusion matrix structure

	Actual value		
Predicted value		Positives	Negatives
	Positives	TP (True Positive)	FP (False Positive)
	Negatives	FN (False negative)	TN (True Negative)

To check the classification performance of the algorithm, different measures can be used. Some of the most common measures that can be calculated from the confusion matrix are:

Classification accuracy: The True rate of the model and it's measured as the summation of number for the correct classes divided by the total number.
Sensitivity: The true positive rate, measures the ratio of the actual positives. Sensitivity $= \text{tp}/(\text{tp} + \text{fn})$
specificity: also known as the true negative rate, measures the ratio of negatives. Specificity $= \text{tn}/(\text{tn} + \text{fp})$
Precision: measures the provided accuracy of a certain class that has been forecasted. Precision $= \text{tp}/(\text{tp} + \text{fp})$ [12]

The resulting confusion matrix, where 'A' stands for not-pop class and 'B' stands for popular class is shown in Table 3. Results of running Adaboost and remaining algorithms is detailed in the next section for comparison. It can be seen from the confusion Matrix of the AdaBoost algorithm that 23397 data points were correctly classified. This will give an accuracy of 0.59.

Table 3. Confusion matrix for AdaBoost classification

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	11878	8207
	Negatives(A)	8061	11501

3.4 K Nearest Neighbor K-NN

K-NN is a supervised learning algorithm and an easy algorithm that saves all available instances, and classifies them based on distance functions (similarity measure). K-NN has been used in estimation of statistics and recognition pattern as a non-parametric method. The K-NN classifies each instance based on its neighbors, and assigns each data point to the class with the most similarity gauged by a distance function like the

Euclidian distance function [18]. In our work we used K-NN with K = 1, where the data instance is simply assigned to the class of that single nearest neighbor. K-NN gave us the worst result among all the other tested algorithms when the value of K was 1. Experimenting further with K-NN by increasing the value of k eventually gave the best result among all the tested algorithms. However having a high value of k eventually started to give lower accuracy rate.

Weka KNN classifier scheme used is as follows with different values of K. The results for the performance measures are shown in Table 4. We can see from the table that the best results were obtained when K = 37. The confusion matrix for each value of is shown in Tables 15 and 16.

Table 4. K-NN performance measures for different value of K

K	Correctly classified	Incorrectly classified	Precision	Recall	Time(s) to build model	RMS
1	22681	16963	.572	.572	.01	.6541
3	23339	16305	.589	.589	.25	.5376
5	23860	15784	.602	.602	.08	.5098
7	24193	15451	.609	.646	.09	.4977
11	24514	15130	.619	.618	.11	.4882
15	24571	15073	.620	.620	.07	.4834
33	24881	14763	.628	.628	.01	.4775
35	24900	14744	.628	.628	.13	.4772
36	24894	14750	.628	.626	.12	.4769
37	**24918**	**14726**	**.629**	**.629**	**.01**	**.477**
38	24894	14750	.629	.628	.12	.4441
40	24843	14801	.628	.627	.01	.4768
50	24868	14776	.628	.625	.01	.4763
66	24867	14777	.628	.627	.06	.4758

Table 5. Confusion matrix for K-NN (K = 1)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	11838	8244
	Negatives (A)	8719	10843

Table 6. Confusion matrix for K-NN (K = 3)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	12357	7725
	Negatives (A)	8580	10982

Table 7. Confusion matrix for K-NN (K = 5)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	12743	7339
	Negatives (A)	8445	11117

Table 8. Confusion matrix for K-NN (K = 7)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13207	6875
	Negatives (A)	8255	11307

Table 9. Confusion matrix for K-NN (K = 11)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13207	6875
	Negatives (A)	8255	11307

Table 10. Confusion matrix for K-NN (K = 13)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13292	6790
	Negatives (A)	8283	11279

Table 11. Confusion matrix for K-NN (K = 33)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13419	6663
	Negatives (A)	8100	11462

Table 12. Confusion matrix for K-NN (K = 35)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13413	6669
	Negatives (A)	8075	11487

Table 13. Confusion matrix for K-NN (K = 37)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13404	6678
	Negatives (A)	8048	11514

Table 14. Confusion matrix for K-NN (K = 40)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13816	6266
	Negatives (A)	8510	11052

Table 15. Confusion matrix for K-NN (K = 50)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13701	6381
	Negatives (A)	8420	11142

Table 16. Confusion matrix for K-NN (K = 66)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13567	6515
	Negatives (A)	8262	11300

The accuracy as calculated from Table 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16 is shown in Fig. 2. We can see how the accuracy changes when we increase the value of K in K-NN, best accuracy was at K = 37. The X-Axis showing the k value and the y-Axis showing the accuracy.

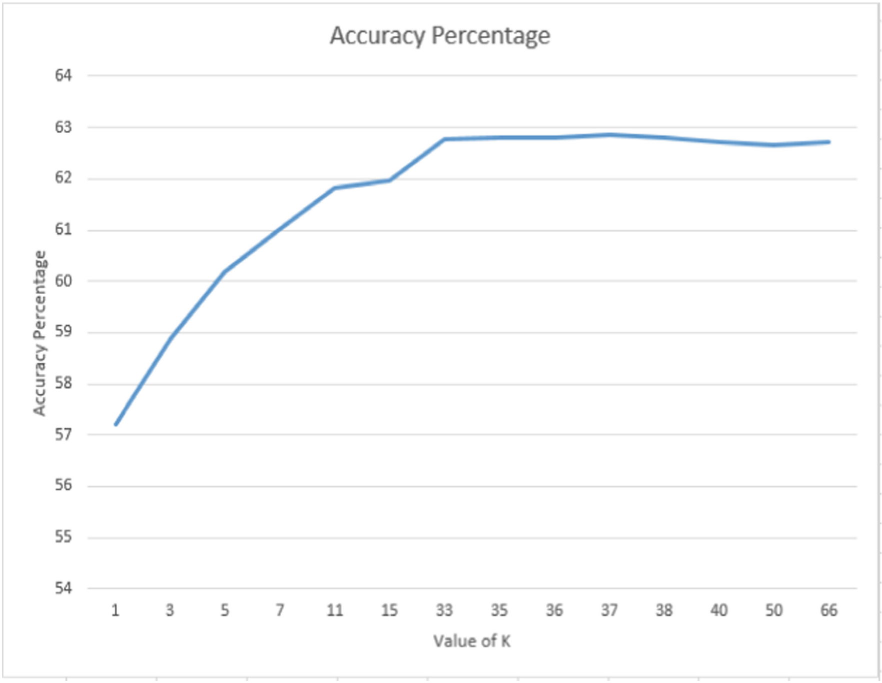


Fig. 2. K-NN accuracy for different values of K

3.5 J48 Decision Tree (Pruned Tree)

Decision Tree structure is a tree-like flowchart in which the internal node perform a test on the attribute. The branch stands for a consequence of the test, and the leaf node represents the class label. The root node in a tree is the topmost node. The Decision tree uses different algorithms. J48 is an algorithm for generating a decision tree developed by Ross Quinlan [19]. J48 is usually used for classification. The J48 was implemented using Weka.

We can see by the resulting confusion matrix (Table 17) that the accuracy was similar as when we used Adaboost with J48 to classify popularity:

Table 17. Confusion matrix for J48 decision tree

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	11878	8204
	Negatives (A)	8061	11501

3.6 Naïve Bayes

Which is a Machine Learning Algorithm that need to be trained for supervised learning tasks like classification and prediction or for unsupervised learning tasks like clustering.

The resulting confusion matrix, where ‘A’ stands for not-popular class and ‘B’ stands for popular class. We notice a relatively high error rate Table 18.

Table 18. Confusion matrix for Naïve Bayes

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	18714	1368
	Negatives (A)	17201	2361

4 Results and Summary

Four different Algorithms have been used to classify the articles’ data based on their popularity. Table 19 shows a comparison of these algorithms.

From this comparison we can see that we got the same number of correctly classified and incorrectly classified instances when running Adaboost using J48 classifier and J48 alone. While the AdaBoost gave slightly less root mean squared error and took more time to build.

Naïve Bayes gave the worst result amongst all with the least number of correctly classified instances and highest root mean squared error.

KNN gave the best result among all the algorithms when K = 37 with accuracy which proved better than the random forest and SVM in a previous work [3]. K-NN was also the fastest when comparing time it took to build the model. KNN also gave the least root mean squared error value among the other algorithms compared.

Table 19. Comparison of the 4 classification algorithms

Algorithm	Correctly classified	Incorrectly classified	Precision	Recall	Time to build model (S)	RMS error
AdaBoost using j48 classifier	23379	16265	0.590	0.590	42.61	0.6114
KNN K = 37	24918	14726	0.629	0.629	0.01	0.477
J48	23379	16265	0.590	0.590	39.53	0.6149
NB	21075	18569	0.576	0.532	0.59	0.6689

5 Conclusion

In this paper, we have introduced and discussed four machine learning algorithms that are usually used in supervised learning. The four algorithms were: Adaptive boosting, K-NN, Decision Trees and Naïve Bayes. The data classified in this work is related to some articles that have 47 attributes and one target class (popular or unpopular). The article that had more than 1400 shares was considered as popular. We have seen that the K-NN with k = 37 has the best performance among the four algorithms.

For future work we will consider splitting the articles into three target classes and perform the comparison between the mentioned and more classification algorithms such as SVM and Artificial Neural Network.

References

1. UCI machine learning repository. https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity. Accessed 16 Sept 2019
2. Mashable. http://mashable.com. Accessed 20 Sept 2019
3. Fernandes, K., Vinagre, P., Cortez, P.: A proactive intelligent decision support system for predicting the popularity of online news. In: Pereira, F., Machado, P., Costa, E., Cardoso, A. (eds.) EPIA 2015. LNCS (LNAI), vol. 9273, pp. 535–546. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23485-4_53
4. Tatar, A., Amorim, M., Fdida, S., Antoniadis, P.: A survey on predicting the popularity of web content. J. Internet Serv. Appl. **5**(1), 1–20 (2014)
5. Ahmed, M., Spagna, S., Huici, F., Niccolini, S.: A peek into the future: predicting the evolution of popularity in user generated content. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, pp. 607–616. ACM (2013)
6. Lee, J., Moon, S., Salamatian, K.: An approach to model and predict the popularity of online contents with explanatory factors. In: ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Canada, pp. 623–630 (2010)
7. Kaltenbrunner, A., Gomez, V., Lopez, V.: Description and prediction of Slashdot activity. In: Web Conference, LA-WEB 2007, pp. 57–66. IEEE, Latin American (2007)
8. SlashdotMedia: Slashdot: News for nerds, stuff that matters (2016). https://slashdot.org/. Accessed 11 Sept 2019
9. Szabo, G., Huberman, B.: Predicting the popularity of online content. Commun. ACM **53**(8), 80–88 (2010)

10. Tatar, A., Antoniadis, P., De Amorim, M., Fdida, S.: From popularity prediction to ranking online news. Soc. Network Anal. Min. **4**(1), 1–12 (2014)
11. Lee, J., Moon, S., Salamatian, K.: Modeling and predicting the popularity of online contents with cox proportional hazard regression model. Neurocomputing **76**(1), 134–145 (2012)
12. Roja, B., Asur, S., Huberman, B.: The pulse of news in social media: forecasting popularity. In: Proceedings of the 6th International AAAI Conference on Weblogs and Social Media, ICWSM (2012)
13. Sasa, P., Osborne, M., Lavrenko, V.: RT to Win! Predicting message propagation in Twitter. In: ICWSM, Spain(2011)
14. Xuandong, L., Hu, X., Fang, H.: Is your story going to spread like a virus? Machine learning methods for news popularity prediction. In: CS229 (2015)
15. Hensinger, E., Flaounas, I., Cristianini, N.: Modelling and predicting news popularity. Pattern Anal. Appl. **16**(4), 623–635 (2013)
16. Freund, Y., Schapire, R.: Decision-Theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
17. Stehman, S.: Selecting and interpreting measures of thematic classification accuracy. Remote Sens. Environ. **62**(1), 77–89 (1997)
18. Altman, N.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**(3), 175–185 (1992)
19. Quinlan, J.: Programs for Machine Learning. Morgan Kaufmann Publishers, San Francisco (1993)
20. Weka 3 - Data Mining with Open Source Machine Learning Software in Java, Cs.waikato.ac.nz (2016). http://www.cs.waikato.ac.nz/~ml/weka/. Accessed 14 Sept 2019

A Novel Feature Extraction Model to Enhance Underwater Image Classification

Muhammad Irfan[1](✉), Jiangbin Zheng[1], Muhammad Iqbal[2], and Muhammad Hassan Arif[3]

[1] School of Software, Northwestern Polytechnical University, Xian, China
mirfan@mail.nwpu.edu.cn, zhengjb@nwpu.edu.cn
[2] Faculty of Computer and Information Science, Higher Colleges of Technology, Fujairah, UAE
miqbal1@hct.ac.ae
[3] CESAT, Islamabad, Pakistan
mhassanarif@gmail.com

Abstract. Underwater images often suffer from scattering and color distortion because of underwater light transportation characteristics and water impurities. Presence of such factors make underwater image classification task very challenging. We propose a novel classification convolution autoencoder (CCAE), which can classify large size underwater images with promising accuracy. CCAE is designed as a hybrid network, which combines benefits of unsupervised convolution autoencoder to extract non-trivial features and a classifier, for better classification accuracy. In order to evaluate classification accuracy of proposed network, experiments are conducted on Fish4Knowledge dataset and underwater synsets of benchmark ImageNet dataset. Classification accuracy, precision, recall and f1-score results are compared with state-of-the-art deep convolutional neural network (CNN) methods. Results show that proposed system can accurately classify large-size underwater images with promising accuracy and outperforms state-of-the-art deep CNN methods. With the proposed network, we expect to advance underwater image classification research and its applications in many areas like ocean biology, sea exploration and aquatic robotics.

Keywords: Convolutional autoencoder · Deep learning · Convolutional neural network · Underwater images

1 Introduction

Underwater image classification has recently attracted many computer vision researchers because of its applications in marine sciences and autonomous underwater vehicles (AUV). Underwater image classification is a challenging task because of complex underwater environment and poor lighting conditions. Factors such as wavelength dependent absorption and scattering of light degrade the visibility of underwater images and make them poorly contrasted and blur [1].

These factors hinder the performance of underwater image classification applications used in sea exploration, aquatic robotics and sea environmental surveillance [2].

In recent years, most of the studies used for underwater image classification are deep learning based. Hongwei et al. [3] used combination of CNN, support vector machine (SVM), principal component analysis (PCA) and spatial pyramid pooling (SPP) for classification of fish species from Fish4Knowledge dataset. Xin et al. [4] used same Fish4Knowledge dataset for fish classification and tracking from videos. Xu et al. [5] used pre-trained GoogleNet along with augmentation techniques for underwater image classification. Limited labelled data is used by Siddiqui et al. [6] for automatic classification of fish through pre-trained neural networks. Jin et al. [7] used pre-trained AlexNet, trained on ImageNet dataset, for underwater image recognition in small sample size situations.

Convolutional Autoencoder (CAE) is a type of unsupervised learning [8]. CAE extends the basic structure of the autoencoder by using convolution layers instead of the fully connected layers to preserve the spatial locality of input data [9]. It consists of two parts, encoder and decoder [10]. Encoder part consists of pooling layers and convolutional layers. Decoder part consists of up-sampling layers and deconvolution layers. CAE tries to regenerate input at the output by using learned convolution filters. Learned convolutional filters extract non trivial features from the input. Extracted features can be used for classification.

In this study we propose a novel deep classification convolutional autoencoder, named CCAE. CCAE is designed as a hybrid classification convolutional autoencoder architecture. The main objective behind architecture of CCAE is the better feature extraction mechanism by combining classification layer with deep encoder-decoder network during the training as well as the testing phase. It extracts spatially localized features with hint of class. It combines capability of CNN to extract features from images [11] and capability of autoencoder (AE) to compress high dimensional data to low dimension [12] and a classifier to extract features with hint of class. Experiments are conducted, on underwater synsets of benchmark ImageNet dataset [13] and Fish4Knowledge dataset [14] (fish recognition ground truth dataset made by the Fish4-Knowledge project), to verify the architecture of CCAE. Results show that CCAE can correctly classify underwater images with promising accuracy. Classification accuracy, precision, recall and f1-score results of the proposed method are compared with state-of-the-art deep CNN methods such as ResNet [15], DenseNet [16], Inception [17] and Xception [18].

The rest of this paper is organized as follows. In Sect. 2, we review CAE. In Sect. 3, we present details of the proposed deep network. Experimental design is discussed in Sect. 4. Experiment results are presented and discussed in Sect. 5. Conclusion is drawn in Sect. 6.

2 Background

2.1 Convolutional Autoencoder

AE is a special type of neural network mainly used for data dimension reduction [19]. The AE consists of an encoder and a decoder. The encoder part encodes the input data to a hidden representation by using a deterministic nonlinear function, and then the decoder part reconstructs the input back to the original input space [20]. The connections between layers are fully connected.

CAE is categorized as unsupervised learning algorithm [8]. It combines benefits of CNN with unsupervised pre-training of AE [21,22]. It also consists of two parts, encoder and decoder [10]. Encoder part mainly consists of convolutional layers and pooling layers. The encoder convolutionally transforms the input into a latent space. Contrary to traditional AE, CAE preserves the spatial relationship of the input through a bank of 2-D convolutional filters [23]. CAE architecture is mainly used for semantic segmentation [24], image denoising/dehazing [25], deep clustering [26] and for feature extraction [23].

Encoder part is used to extract non-trivial (compressed) features [27]. Decoder part consists of deconvolution layers and up-sampling layers. The decoder tries to regenerate the input by mapping the latent layer data back to the original input space by minimizing the reconstruction error. Extracted compressed features can also be used for classification [23].

For a given input image X the encoder function is defined as

$$Y_i = encoder(X) = \sigma(X * W^i + b) \tag{1}$$

Where b is encoder bias, Y_i is encoding of the input X in a low dimensional space, W^i is 2-D convolutional filter, * is 2-D convolution and σ denotes activation function such as ReLU [28].

In decoding phase, Y_i is the input of the decoding function, which can be defined as

$$G = decoder(Y_i) = \sigma(Y_i * \tilde{W}^i + \tilde{b}) \tag{2}$$

Where $\tilde{W}^i$ is 2-D convolutional filter in decoder and $\tilde{b}$ is bias of decoder,

Mean square error (MSE) function E, as described below, is used as cost function to make output G as close as possible to input X.

$$E(\theta) = 1/n\sum_{i=1}^{m} (G_i - X_i)^2 \tag{3}$$

Back propagation algorithm is used to calculate the gradient of the error function with respect to the parameters as

$$\frac{\delta E(\theta)}{\delta W^i} = X * \partial Y_i + \tilde{Y}_i * \partial G \tag{4}$$

∂G and ∂Y are deltas of the reconstruction and latent space.

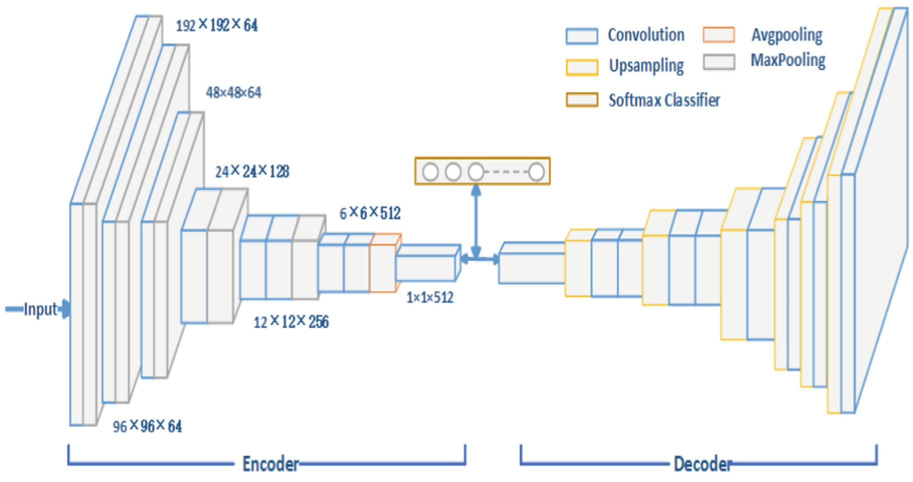


Fig. 1. CCAE architecture diagram

3 Proposed Method

Proposed deep neural network CCAE combines benefits of unsupervised learning and supervised learning. It consists of deep encoder network and corresponding decoder network, combined with a classification layer. CCAE layer wise architecture is elaborated in Fig. 1. Each convolution layer consists of different number of channels. CCAE encoder part takes an image as an input, convolves the input with convolution filters trained during the training phase and extracts non-trivial features of the input image. After the last layer (average pooling) of encoder, a classification layer is attached. The classification layer i.e. softmax, is an integral part of encoder-decoder network during the training as well as during the testing phase. This architecture is used to leverage better features with hint of class. Output of the average pooling layer of encoder is taken as an input to the decoder module. Proposed layers configuration of the encoder module and number of channels in each layer, offer improved feature extraction capability for better classification results. Table 1 summarizes configuration of CCAE layers in terms of number of channels, filter sizes, layer type, output size and input to each layer. Layers of encoder and decoder are separately arranged in Table 1. Keeping in view limitation of space, batch normalization layer used after every convolution layer is not mentioned in Table 1.

As convolution autoencoder is a type of unsupervised learning, so it is trained in an unsupervised way to extract features. In proposed network encoder-decoder network is trained through ensemble of a classification layer with the encoder during the training phase. Moreover, the decoder part is an integral part of the network during the training and testing phases. This training mechanism improves feature extraction by extracting non-trivial features with hint of class, which results in improvement in classification accuracy.

The following subsections describe the proposed module in detail.

3.1 Encoder

The encoder part of CCAE takes an image as an input and extracts the non-trivial features of the input image as described in Eq. (1). The encoder module consists of 8 convolution layers as shown in Fig. 1. Size of feature map output of each layer decreases from the first conv1 (192 $\times$ 192) to the last convolution layer con8 (6 $\times$ 6). Number of channels of convolution layers of encoder module increases from the first convolution layer conv1 (64 channels) to the last convolution layer conv8 (512 channels), as shown in Table 1. Each convolution layer performs the convolution operation to generate a set of feature maps. For element-wise non-linearity ReLU $max(0, x)$ [28] is used with each convolution layer. Regularization of network is done by decreasing the inner co-variate shift of data by using batch normalization layer after every convolution layer [29].

Pooling is used in the encoder network to filter noisy activations in lower layers and to retain robust activations in upper layers [30]. It also reduces the number of connections between convolutional layers which results in computational efficiency. Max-pooling layer with stride 2 (non-overlapping) and window size 2 $\times$ 2 is used after each convolution block to sub-sample output by factor of 2. For robust feature extraction, translation in-variance is achieved by using several max-pooling layers. After the last convolution layer (conv8) of the encoder module, an average pooling layer is used instead of a max pooling layer. Softmax classifier is used as activation non-linearity at the end of the encoder module. Output of the average pooling layer is first flattened (reshaped) and then it is given as an input to softmax classifier.

3.2 Decoder

The decoder module tries to reconstruct the same input at output layer through the combination of up-sampling and de-convolution as described in Eq. (2). It up-samples the encoded feature maps, by using a trainable filter bank. In the decoder, there are de-convolutional and up-sampling layers corresponding to each convolution and pooling layer in encoder. Hence, decoder part also consists of 8 de-convolution layers.

In the decoder, de-convolution operation is achieved using convolution operation along with up-sampling [31]. Contrary to convolution and pooling operations, the output of the de-convolutional operation is an enlarged and dense activation map. The output of an unpooling layer is an enlarged, yet sparse activation map. The convolution layer is used to densify the sparse activations obtained as output of unpooling. Each convolution layer performs convolution operation to up-sampled input by a factor of stride value with padding through trainable decoder filter banks. A batch normalization layer is employed after each convolution layer. Inspired by DeepLab [32], bilinear interpolation algorithm is employed in up-sampling layer.

Table 1. CCAE layers configuration

Encoder			
Layer Type	Filter	Output Size	Connected to
Input	–	192 × 192 × 3	–
conv1	3 × 3	192 × 192 × 64	Input
maxpooling1	2 × 2	96 × 96 × 64	conv1
conv2	3 × 3	96 × 96 × 64	maxpooling1
maxpooling2	2 × 2	48 × 48 × 64	conv2
conv3	3 × 3	48 × 48 × 64	maxpooling2
maxpooling3	2 × 2	24 × 24 × 64	conv3
conv4	3 × 3	24 × 24 × 128	maxpooling3
maxpooling4	2 × 2	12 × 12 × 128	conv4
conv5	3 × 3	12 × 12 × 256	maxpooling4
conv6	3 × 3	12 × 12 × 256	conv5
maxpooling5	2 × 2	6 × 6 × 256	conv6
conv7	3 × 3	6 × 6 × 512	maxpooling5
conv8	3 × 3	6 × 6 × 512	conv7
avgpooling1	6 × 6	1 × 1 × 512	conv8
softmax	.		avgpooling1
Decoder			
conv9	1 × 1	1 × 1 × 512	avgpooling1
upsampling1	6 × 6	6 × 6 × 512	conv9
conv10	3 × 3	6 × 6 × 512	upsampling1
conv11	3 × 3	6 × 6 × 512	conv10
upsampling2	2 × 2	12 × 12 × 512	conv11
conv12	3 × 3	12 × 12 × 256	upsampling2
conv13	3 × 3	12 × 12 × 256	conv12
upsampling3	2 × 2	24 × 24 × 256	conv13
conv14	3 × 3	24 × 24 × 128	upsampling3
upsampling4	2 × 2	48 × 48 × 128	conv14
conv15	3 × 3	48 × 48 × 64	upsampling4
upsampling5	2 × 2	96 × 96 × 64	conv15
conv16	3 × 3	96 × 96 × 64	upsampling5
upsampling6	2 × 2	192 × 192 × 64	conv16
conv17	3 × 3	192 × 192 × 3	upsampling6

In decoder module, the trained filters in convolutional layers correspond to bases to reconstruct shape of an input image. Therefore, similar to the encoder module, a hierarchical structure of convolutional layers are used to capture different level of input image details.

3.3 Loss Function

CCAE has two output layers, so it is a challenging task to train this deep network. Let L represents the total loss function of CCAE, which is can be calculated as follow:

$$L = L_r + L_c \tag{5}$$

Where, L_r represents mean square error (MSE) and used to make output of convolutional autoencoder as close as possible to input, as discussed in Eq. (3). L_c represents categorical cross entropy loss used for classification.

$$L_c = -\sum_{i=1}^{N} \sum_{k=1}^{K} t_{ik} log(y_{ik}) \tag{6}$$

Where, t_{ik} is the target, y_{ik} is the calculated output probability. During the training process, in order to minimize loss L, CCAE tries to minimize L_r and L_c simultaneously.

4 Experimental Design

We conducted experiments on underwater synsets of benchmark ImageNet dataset [13] and Fish4Knowledge dataset to demonstrate the effectiveness of proposed method for better classification. We perform comparisons with state-of-the-art deep CNN methods such as ResNet [15], DenseNet [16], Inception [17] and Xception [18].

4.1 Datasets

Experiments are conducted on Fish4Knowledge dataset [14] and underwater synsets of benchmark ImageNet dataset. Scattering and absorption of underwater light causes color distortion and visibility degradation to underwater images. Water impurities and high water density augment the complexity of underwater image classification task. In both datasets images vary significantly in size, object orientation, scale and underwater opacity level [33].

Fish4Knowledge. Fish4Knowledge dataset consists of underwater images of different fish species. The detail of classes included in this dataset is shown in Table 2. This underwater live fish dataset is prepared from live videos recorded from the under sea. There are total 23 species and total 27,370 fish images. Images in dataset are manually labeled by expert marine biologists. Images vary in size ranging from about 20 × 20 to about 200 × 200 pixels. The number of fish images in different classes are quite imbalanced. The number of images in big class is almost 1000 times of the least one. So it is quite difficult to achieve a high accuracy over the whole dataset.

Table 2. Fish4Knowledge dataset

ID	Class	Images	ID	Class	Images
1	Dascyllus reticulatus	12112	13	Plectroglyphidodon dickii	3683
2	Chromis chrysura	3593	14	Amphiprion clarkii	4049
3	Chaetodon lunulatus	253	15	Chaetodon trifascialis	190
4	Myripristis kuntee	450	16	Acanthurus nigrofuscus	218
5	Neoniphon sammara	299	17	Abudefduf vaigiensis	98
6	Hemigymnus fasciatus	241	18	Pomacentrus moluccensis	181
7	Canthigaster valentini	147	19	Hemigymnus melapterus	42
8	Lutjanus fulvus	206	20	Scolopsis bilineata	49
9	Scaridae	56	21	Pempheris vanicolensis	29
10	Zanclus cornutus	21	22	Neoglyphidodon nigroris	16
11	Zebrasoma scopas	90	23	Balistapus undulatus	41
12	Siganus fuscescens	25	...	...	

ImageNet Underwater Synsets. There are forty five (45) synsets related to underwater environment in ImageNet dataset [5]. These synsets consists of images both from underwater and on water surface. The detail of these underwater synsets is shown in Table 3. We take all forty five underwater synsets for experiments, and divided them in three (03) groups i.e. G I, G II and G III. So, each group consists of fifteen (15) synsets. One group is taken at a time for a single experiment.

Table 3. ImageNet underwater synsets

Group I synsets	Images	Group II synsets	Images	Group III synsets	Images
Great White Shark	1242	Lobster	1206	Steller Seal	1342
Scubadiving	1507	Hammer head shark	1094	Liner	1385
Australian Seal	1200	Snorkel Diving	1257	Nuclear submarine	1161
Star Fish	1396	California seal	1245	Whale Shark	1185
Sea Fan	1270	Sea slug(Hermissenda)	2211	Dugong	1018
Attack Submarine	1014	Prawn	1156	Skin diving	1211
Sea snake	1108	Corel Reef	1706	Lion fish	1513
king crab	1019	Wreck	1240	American lobster	1123
Shrimp	1236	Sea pen	1084	Rock crab	1215
Sea turtle	1485	Sea cucumber	1167	Sea urchin	1186
Sea bed	1071	Sea horse	1272	Walrus	1101
Aircraft career	1321	Battle ship	1185	Sea otter	1385
Sea hare	1138	Manatee	1360	Sea slug	711
Chiton	971	Sea cow	424	Bivalve	865
Sea lampery	670	Oster	882	Rockfish	1415

4.2 Parameters Settings

To leverage computational efficiency all images of ImageNet underwater synsets are resized to 192 × 192 pixels. All images of Fish4Knowledge dataset are resized to 96 × 96 pixels.

For fair comparison all methods are trained only on synsets used in experiments without any data augmentation technique. CCAE is trained using ADAM [34] with learning rate=0.001 and batch size 32. For fair comparison, all methods used for comparison are trained with same parameters. All methods run for 120 epochs for ImageNet dataset and for 50 epochs for Fish4Knowledge dataset. In all experiments 70% of the available data is randomly allocated for training and remaining 30% for testing. We implemented CCAE using TensorFlow with Keras as deep learning framework on Google Colab having Tesla K80 GPU.

5 Results and Discussion

5.1 Results

To make comparison, we implemented ResNet [15], DenseNet [16], Inception [17] and Xception [18] as per settings recommended by respective authors.

Table 4 shows the results of classification accuracy, precision, recall and f1-score of all methods for ImageNet underwater synsets of Group I. Results show that proposed approach achieved accuracy of 73.75% and outperformed all other methods. DenseNet performed better than other CNN methods and achieved accuracy of 70.54%. Accuracy of Inception remained low among all methods with score of 58.95%. CCAE achieved f1-score of .72, whereas both DenseNet and Xception net achieved f1-score of .71.

Table 4. Classification accuracy, precision, recall and f1-score comparison for ImageNet synsets group I

Method	Accuracy	Precision	Recall	F1-Score
CCAE	0.7375	0.73	0.72	0.72
DenseNet	0.7054	0.72	0.71	0.71
ResNet	0.6129	0.67	0.63	0.65
Inception	0.5895	0.61	0.60	0.60
Xception	0.6988	0.73	0.70	0.71

Classification accuracy, precision, recall and f1-score of ImageNet synsets of Group II are elaborated in Table 5. Proposed method achieved 70.98% accuracy, best among all methods. Among other methods DenseNet achieved better accuracy of 68.38%. Inception remained again at bottom level with 57.14% accuracy.

Results for classification accuracy, precision, recall and f1-score Group III synsets are depicted in Table 6. Proposed method again achieved best accuracy

Table 5. Classification accuracy, precision, recall and f1-score comparison for ImageNet synsets group II

Method	Accuracy	Precision	Recall	F1-Score
CCAE	0.7098	0.72	0.71	0.71
DenseNet	0.6838	0.70	0.69	0.69
ResNet	0.5966	0.64	0.62	0.63
Inception	0.5714	0.60	0.59	0.59
Xception	0.6718	0.69	0.68	0.68

Table 6. Classification accuracy, precision, recall and f1-score comparison for ImageNet synsets group III

Method	Accuracy	Precision	Recall	F1-Score
CCAE	0.7137	0.71	0.70	0.70
DenseNet	0.6735	0.69	0.69	0.69
ResNet	0.6149	0.65	0.63	0.64
Inception	0.5826	0.58	0.58	0.58
Xception	0.6538	0.66	0.65	0.65

with score 71.37%. DenseNet achieved accuracy of 67.35%, which is better than remaining methods.

Table 7 shows the classification accuracy, precision, recall and f1-score results of Fish4Knowledge dataset. All twenty three (23) classes are taken simultaneously for classification. It is evident that proposed method CCAE performed better than other methods and achieved accuracy of 99.28%. Among other methods DeepFish net with SVM classifier using data augmentation and scaling proposed by Hongwei achieved better accuracy with score 98.64%. Whereas, classification accuracy of 98.59% was achieved by DeepFish net with data augmentation. It is not worthy that our method achieved best accuracy among all methods without using any augmentation technique. Among other deep CNN methods Xception achieved better accuracy of 98.34%. Classification accuracy of Inception remained low among all used methods with scorer of 90.50%. CCAE achieved 0.99 precision score, the highest among all methods. Recall score of three methods such as CCAE, DenseNet and Xception remained same i.e. 0.98. Similarly CCAE, DenseNet and Xception achieved the same f1-score i.e. 0.98.

As observed, proposed method achieves best classification accuracy in all experiments for both datasets, which also highlights the robustness of our method. Whereas, it can also be observed that generally DenseNet performed better classification than ResNet, Inception and Xception in ImageNet underwater synsets classification. Whereas, DeepFish achieved better classification accuracy results as compared to state-of-the-art deep CNN methods. It is noteworthy that all five methods achieved best accuracy results for classification of

Table 7. Classification accuracy, precision, recall and f1-score comparison for Fish4Knowledge dataset

Method	Accuracy.	Precision	Recall	F1-Score
CCAE	0.9928	0.99	0.98	0.98
DenseNet	0.9758	0.98	0.98	0.98
ResNet	0.9480	0.96	0.95	0.95
Inception	0.9050	0.92	0.91	0.91
Xception	0.9834	0.98	0.98	0.98
DeepFish-SVM-aug-scale [3]	0.9864	...	...	...
DeepFish-SVM-aug [3]	0.9859	...	...	...
Deep-CNN [3]	0.9857	...	...	...
DeepFish-Softmax-aug-scale [3]	0.9849	...	...	...

fish4knowledge dataset as compared to ImageNet underwater synsets classification. Whereas, generally all five methods achieved worst accuracy results for classification of ImageNet underwater synsets group II.

5.2 Discussion

CCAE architecture takes advantage of both unsupervised learning as well as the localized spatial features enabled by convolutional filtering. It outperformed other state-of-the-art deep CNN architectures such as Deepfish, Inception, ResNet, DenseNet and Xception. The training of CCAE is carried out by ensemble of a softmax classifier during training time with the convolutional autoencoder architecture. This training strategy enabled the network to extract better feature extraction, which resulted in improved classification accuracy results.

Experiments are also conducted to assess the impact of number of layers and number of channels in each layer, on classification accuracy. Various experiments are conducted to determine the most appropriate number of layers and filters for best classification results. ImageNet underwater synsets of group I are taken for experiments. Table 8 shows the result of comparison of classification accuracy for different number of layers and filters.

Table 8. Impact of no. of layers and no. of filter on accuracy

Sr. no.	Layers	Filters in each layer	Accuracy
1	08	64-64-64-128-256-256-512-512	0.7375
2	08	64-128-256-256-512-512-512-512	0.7216
3	09	64-64-128-256-256-512-512-512-512	0.6269
4	10	x64-64-128-128-256-256-512-512-512-512	0.6114

Results as shown in Table 8 suggest that best classification accuracy with score 73.75% is achieved with CCAE encoder architecture which consists of 08 convolution layers (serial no 1 in Table 8). When number of channels in few layers changed by keeping the number of layers same, as shown at serial no 2 of Table 8, the classification accuracy of 72.16% achieved. This configuration of layers and channels in the encoder module is similar to VGG11 architecture. Classification accuracy keeps decreasing when one more convolution layer having 64 channels added to the serial 2 configuration. New configuration is shown at serial number 3 in Table 8. Configuration of layers and channels of the encoder at serial no 4 in Table 8 is similar to the VGG13 architecture. It consists of 10 convolution layers. Classification accuracy of 61.14% is achieved by using this configuration. From this experiment it can be inferred that increasing number of channels and number of layers in design of the encoder of CCAE does not result in increase in classification accuracy. And CCAE encoder architecture is most optimized to achieve best classification accuracy results.

Max pooling layer is used after every convolution layer of the encoder module. Avg pooling layer is used after last convolution layer of the encoder module. Two commonly used pooling layers in deep CNN are avg pooling layer and max pooling layer. We conducted the experiments to find better pooling layer to be used after last convolution layer of encoder. It was found during the experiments that classification accuracy of 73.75% was achieved by using Avg pooling layer. Whereas, classification accuracy dropped to 57.38% when max pooling layer is used after last convolution layer of the encoder module. It can be inferred that important information of features of the input image is retained by using avg pooling layer at the end of last convolution layer of the encoder module, which in result improves the classification accuracy.

6 Conclusion

In this paper, a novel feature extraction technique is proposed to improve the accuracy of image classifiers. Proposed model uses the strength of unsupervised deep convolutional autoencoder to learn useful features from large training data set to train a supervised softmax classifier. Experiments on underwater image data sets demonstrate that proposed model has remarkably improved classification accuracy. Classification accuracy, precision, recall and f1-score results showed that proposed model has out performed state-of-the-art deep CNN methods. The proposed model can easily be used to other image classification and object recognition tasks.

References

1. Schettini, R., Corchs, S.: Underwater image processing: state of the art of restoration and image enhancement methods. EURASIP J. Adv. Signal Process. **2010**(1), 1–14 (2010). 746052

2. Anwar, S., Li, C., Porikli, F.: Deep Underwater Image Enhancement. CoRR abs/1807.03528 (2018). arXiv:1807.03528
3. Qin, H., et al.: DeepFish: accurate underwater live fish recognition with a deep architecture. Neurocomputing **187**(C), 49–58 (2016)
4. Sun, X., et al.: Transferring deep knowledge for object recognition in low-quality underwater videos. Neurocomputing **275**(C), 897–908 (2018)
5. Xu, Y., et al.: Underwater image classification using deep convolutional neural networks and data augmentation. In: 2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), pp. 1–5 (2017)
6. Siddiqui, S.A., et al.: Automatic fish species classification in underwater videos: exploiting pre-trained deep neural network models to compensate for limited labelled data. ICES J. Mar. Sci. **75**(1), 374–389 (2017)
7. Jin, L., Liang, H.: Deep learning for underwater image recognition in small sample size situations. In: OCEANS 2017, Aberdeen, pp. 1–4 (2017)
8. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV 2015, pp. 1520–1528. IEEE Computer Society, Washington (2015)
9. Luo, W., et al.: Convolutional sparse autoencoders for image classification. IEEE Trans. Neural Netw. Learn. Syst. **29**(7), 3289–3294 (2018)
10. Guo, Y., et al.: Deep learning for visual understanding. Neurocomputing **187**(C), 27–48 (2016)
11. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) ICANN 2011, part I. LNCS, vol. 6791, pp. 52–59. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21735-7_7
12. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
13. Deng, J., et al.: ImageNet: a large-scale hierarchical image database. In: CVPR 2009 (2009)
14. Boom, B.J., et al.: Supporting ground-truth annotation of image datasets using clustering. In: 2012 21st International Conference on Pattern Recognition (ICPR 2012), November 2012, pp. 1542–1545. IEEE Computer Society, Los Alamitos (2012)
15. He, K., et al.: Deep residual learning for image recognition. CoRR abs/1512.03385 (2015). arXiv: 1512.03385
16. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. CoRR abs/1608.06993 (2016). arXiv:1608. 06993
17. Szegedy, C., et al.: Going deeper with convolutions. CoRR abs/1409.4842 (2014). arXiv: 1409.4842
18. Chollet, F.: Xception: deep learning with depthwise separable convolutions. CoRR abs/1610.02357 (2016). arXiv: 1610.02357
19. Zhang, C., et al.: Deep sparse autoencoder for feature extraction and diagnosis of locomotive adhesion status. J. Control. Sci. Eng. **2018**, 8676387:1–8676387:9 (2018)
20. Yanming, G., et al.: A review of semantic segmentation using deep neural networks. Int. J. Multimedia Inf. Retrieval **7**(2), 87–93 (2018)
21. Baldi, P.: Autoencoders, unsupervised learning and deep architectures. In: Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27, UTLW 2011, Washington, USA, pp. 37–50. JMLR.org (2011)

22. Szegedy, C., et al.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9 (2015)
23. Luo, W., et al.: Convolutional sparse autoencoders for image classification. IEEE Trans. Neural Netw. Learn. Syst. **29**, 3289–3294 (2018)
24. Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (2017)
25. Zhang, H., Patel, V.M.: Densely connected pyramid dehazing network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018
26. Guo, X., et al.: Deep clustering with convolutional autoencoders. In: ICONIP (2017)
27. Vincent, P., et al.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, ICML 2008, pp. 1096–1103. ACM (2008)
28. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: ICML Workshop on Deep Learning for Audio, Speech and Language Processing (2013)
29. Santurkar, S., et al.: How does batch normalization help optimization? In: Bengio, S., et al. (eds.) Advances in Neural Information Processing Systems 31, pp. 2483–2493. Curran Associates Inc., (2018)
30. Jiuxiang, G., et al.: Recent advances in convolutional neural networks. Pattern Recogn. **77**(C), 354–377 (2018)
31. Dumoulin, V., Visin, F.: A guide to convolution arithmetic for deep learning. CoRR abs/1603.07285 (2016)
32. Chen, L.-C., et al.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2018)
33. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Pereira, F., et al. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates Inc., (2012)
34. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2015)

A Comprehensive Analysis of MRI Based Brain Tumor Segmentation Using Conventional and Deep Learning Methods

Hikmat Khan[1], Syed Farhan Alam Zaidi[2], Asad Safi[3](✉), and Shahab Ud Din[3]

[1] Department of Computer Science, COMSATS University, Islamabad, Pakistan
[2] Department of Computer Science and Engineering, Chung-Ang University, Seoul, South Korea
[3] Faculty of Computer Information Science (CIS), Higher Colleges of Technology, Sharjah Women's Campus, Sharjah, UAE
asafi@hct.ac.ae

Abstract. Brain tumor segmentation plays an important role in clinical diagnosis for neurologists. Different imaging modalities have been used to diagnose and segment brain tumor. Among all modalities, MRI is preferred because of its non-invasive nature and better visualization of internal details of the brain. However, MRI also comes with certain challenges like random noise, various intensity levels, and in-homogeneity that makes detection and segmentation a difficult task. Manual segmentation is extremely laborious and time consuming for the physicians. Manual segmentation is also highly dependent on the physician's domain knowledge and practical experience. Also, the physician may not be able to see details at the pixel level and may only notice the tumor if it is more prominent and obvious. Therefore, there is a need for brain tumor segmentation techniques that play major role in perfect visualization to assist the physician in identifying different tumor regions. In this paper, we present recent advancements and comprehensive analysis of MRI-based brain tumor segmentation techniques that used conventional machine learning and deep learning methods. We analyze different proposed conventional and state-of-the-art methods in chronological order using Dice similarity, specificity, and sensitivity as performance measures.

Keywords: Brain tumor · Segmentation · Machine Learning · Deep Learning · Dice Similarity · 2D and 3D convolutional ANN

1 Introduction

The goal of automated or semi-automated brain tumor segmentation methods is to detect and accurately segment the abnormal regions within the brain. Brain tumors are mainly divided into two grades: Low Graded Glioma (LGG) and High Graded Glioma (HGG) [2]. LGG is less aggressive in nature while HGG

C. Brito-Loeza et al. (Eds.): ISICS 2020, CCIS 1187, pp. 92–104, 2020.
https://doi.org/10.1007/978-3-030-43364-2_9

is more aggressive in nature. Patients diagnosed with LGG have 5 years while patients diagnosed with HGG have 1 year of life expectancy on the average respectively [17]. The possible ways for treating the brain tumor are chemotherapy, radiations, and surgery. In clinician practice, neurologist manually segments the abnormal regions at each slice of MR imaging modalities [12,23,33]. This manual segmentation is subjective, prone to error, time consuming, costly and highly dependent on the neurologist's experience. Therefore, automated or semi-automated framework for accurate brain tumor segmentation is highly demanded. Such system could help in timely detection of tumor and also guides treatment planning. In this regards, different researchers attempted different Machine Learning (ML) techniques in order to develop system, which can address these demands, act as intelligent assistant and also enhance the neurologist efficiency [23].

In the light of literature, different ML techniques have been used for brain tumor segmentation task that can be divided into two categories: Conventional ML based techniques and Deep Learning (DL) based techniques. All conventional ML techniques follow below mentioned typical steps: pre-processing, feature extraction, feature selection, training classifier and post processing. At pre-processing step, different image processing techniques are applied in order to reduce the noise and improve the quality of the features, which directly affect the segmentation result. At features extraction step, domain pertinent handcrafted features are computed that directly map the domain knowledge. At feature selection step, different dimensionality reduction techniques are applied in order to reduce feature vector dimension that results in reduced training time, no overfitting, and increased classification accuracy. At training step, the selected classifier is trained using the selected feature vector. Finally, at post-processing step, the results of the classifier are further processed to achieve better results.

Recently, deep learning based techniques achieved record shattering state-of-the-art results on verity of computer vision and biomedical image segmentation tasks [1]. Taking inspiration from these results, different biomedical researchers suggest novel CNN architectures for brain tumor segmentation using MR images. These CNN architectures are based on 2D or 3D convolutional kernels and obtained state-of-the-art results [3,4,8,20,21].

In this paper, we have comprehensively summarized recent ML techniques that are employed specifically for brain tumor segmentation task from year 2010 to 2017. We have categorized ML techniques into two sub-categories: conventional and DL techniques, which can be seen in Fig. 1. Each sub-category is further divided based on the technique being used. We analyzed different proposed conventional and DL state-of-the-art methods that are using Dice similarity, accuracy, error rate, specificity, and sensitivity as performance measures.

The rest of the paper is organized as follows: Sect. 2 presents in detail different machine learning techniques used for brain tumor segmentation, which are further categorized into conventional and DL techniques. In Sect. 3, we evaluate and discuss the methods that are mentioned in Sect. 2. Lastly in Sect. 4, we conclude the discussion.

2 Conventional and Deep Learning Techniques

The number of research publications dedicated to brain tumor segmentation has been grown significantly in the last few decades. The observation captured the increasing demands for fully or semi-automated segmentation methods that could provide complementary information and play an insolent assistant role that aid the neurologist in enhancing quality of treatment. In order to address these requirements, different neuro-computing researchers proposed different ML techniques and achieved promising results.

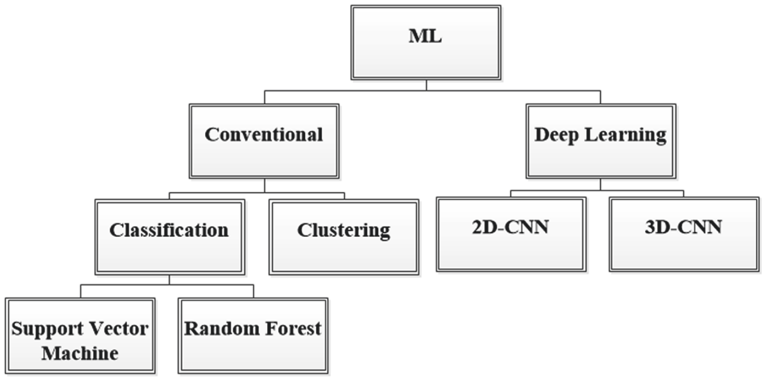


Fig. 1. Proposed Taxonomy: It is the proposed categorization that is used in this work.

2.1 Conventional Techniques

The conventional machine learning (ML) techniques employed for brain tumor segmentation consist of Support Vector Machine (SVM), Random Forest (RF), k-Mean and fuzzy C-Mean. These ML techniques are further classified into two categories: Classification and Clustering algorithms.

2.2 Classification

Most of the classification based ML techniques perform central pixel classification to segment brain tumor using MR images. Following are the classification based techniques that are used for brain tumor segmentation:

SVM Based Techniques

Arikan et al. [7] proposed semi-automated hybrid method of SVM and interactive seed selection based method for brain tumor segmentation. At pre-processing step, they used anisotropic diffusion filter for noise removal in the MR images. Then, random seeds selected from pre-processed MR images to train SVM classifier. For performance evaluation, they used publicly available Medical Image

Computing and Computer-Assisted Intervention (MICCAI) Brain Tumor Segmentation benchmark (BRATS) dataset. They selected four patients from MICCIA BRATS-2015 dataset to evaluate the performance of their proposed method. Their proposed method achieved average Dice Similarity (DS) of about 81% compared with ground truth.

Vaishnavee et al. [31] coped with brain tumor segmentation challenge using SVM and Self Organizing Map (SOM) techniques. At pre-processing step, histogram equalization was performed. The four features (i.e. mean, intensity, number of occurrences and variance) were computed to train classifier. At second step, they employed SOM clustering in order to localize and segment abnormal brain clusters. Moreover, they also performed the classification of brain MR images into different intra-tumor classes. In order to perform this classification in sub-tumor types, Gray Level Co-occurrence Matrix (GLCM) texture features were computed that were then followed by Principle Component Analysis (PCA) as dimensionality reduction step. Proximal Support Vector Machine (PSVM) classified the MR image in one of three classes: either normal, benign, or malignant.

Rathi et al. [25] addressed brain tumor segmentation as pixel classification task. Their approach computed three features: intensity (i.e. mean, variance, standard deviation, median intensity, skewness, and kurtosis), texture (i.e. contrast, correlation, entropy, energy, homogeneity, cluster shade, sum of square variance) and shape (i.e. circularity, irregularity, area, perimeter and shape index). PCA and Linear Discriminant Analysis (LDA) were used to reduce the dimensionality of the computed feature matrix. The SVM was trained to classify each pixel as white matter, gray matter, cerebrospinal fluid (CSF), tumor or non-tumor. The experiment was conducted on 140 brain MR images taken from Internet Brain Segmentation Repository (IBSR).

Zhang et al. [35] fused multi-spectral MR images and addressed the challenges in much efficient and effective way. Their work resulted in reduce computational cost, faster inference and less segmentation error. In their framework, they redefined feature extraction, feature selection and introduced additional adaptive training step. Three features (Intensities, texture information and wavelet transform) were computed in different kernel windows that propagate across multispectral images. During the training stage, SVM was retrained until the feature set got consistent. The SVM pixel classification was then forwarded to region growing algorithm, which further performed the refinement of the tumor boundaries. Their obtained result justified the fusion of multi-spectral MRI images and the two step classification.

Reddy et al. [26] computed three features (Local Binary Pattern (LBP), Histogram of Oriented Gradients (HOG) and Mean Intensity (MI)) in neighborhood pixels. These three features were concatenated to train SVM as pixel level discriminative classifier. Then, the result of SVM was smoothed with Gaussian filter in order to generate the confidence surface. The confidence surface contained the likelihood of each pixel as tumor or non-tumor. The incorporation of the confidence surface with level set and region growing segmentation algorithm achieved promising dice similarity score and outperformed the original version of both algorithms without confidence surface as prior.

Table 1. SVM based methods for brain tumor segmentation. All the results are shown in % unit. SG stands for Self-generated, where B- is used for BRATS in dataset field. The acc., R.call, and TE is used for accuracy, Recall, and Total Error respectively.

Ref.	Technique	Dataset	Dice	R.cal	Acc.	TE	Year
[7]	Random seed selection-SVM	B-2015	80	–	–	–	2016
[31]	PSVM	B-2015	–	90	–	–	2015
[25]	Multi-Kernel SVM	SG	–	–	–	7.6	2010
[35]	PCA based SVM	IBSR	–	–	97.8	–	2012
[26]	SVM & Guassian Filter SVM	SG	69	–	–	–	2012

RF Based Techniques

Following are the RF based techniques proposed for brain tumor segmentation.

Ellwaa et al. [11] proposed a fully automated MRI-based brain tumor segmentation method, which used iterative random forest. The accuracy was improved by iteratively selecting the patient that possessed the detailed information from dataset, which was used to train random forest classifier. The method was validated on BRATS-2016 dataset. The criteria of selecting the patient with highest information helped them to achieve promising results.

Lefkovits et al. [16] proposed a RF based discriminative model for brain tumor segmentation using multi-modal MR images. The discriminative model was used to establish the optimal parameters. These determined optimum parameters were used to fine-tune RF. The fine-tuned RF obtained the optimized tumor segmentation. They used BRATS-2012 and BRATS-2013 datasets for validation of their framework. Their method showed comparable results to other proposed methods on BRATS 2012 and BRATS-2013 datasets.

Meier et al. [19] proposed a RF based discriminative model using highly discriminative features. At pre-processing step, MedPy was used for intensity standardization, which was used to harmonize the sequence intensity profile. The voxel wise features were extracted for training the random forest, which was based on intensities values. The classifier was trained easily and consistently performed well on a large range of parameters.

Reza et al. [27] proposed an improved brain tumor segmentation method that computed textural features on multi-modal MR images. They used BRATS-2013 and BRATS-2014 datasets to validate their method. Mean and median (statistical validation) were calculated to check the efficiency of the proposed method. They performed pixel level classification using random forest classifier. In order to obtain further improved results, they used binary morphological filters on obtained segmented MR images to get precise contours. Moreover, the smallest objects were removed using connected components algorithm during post-processing. The holes in the tumor regions were also detected and filled, using connected neighbor intensities.

Abbasi et al. [5] proposed an automated 3D model for brain tumor detection and segmentation. For preprocessing, bias field correction and histogram matching was used and then ROIs (Region of Interests) were identified and separated from the background of the FLAIR image. They used HOG and LBP for learning features. The RF was used to segment brain tumor on BRATS 2013 dataset.

Table 2. Random Forest (RF) based methods for brain tumour segmentation. Dice similarity of all the methods are shown in % unit. SG stands for self-generated, morph. stands for morphological filters, where B- is used for BRATS.

Ref.	Technique	Dataset	Complete	Core	Enhanced	Year
[11]	Iterative RF	B-2016	82	72	56	2016
[16]	RF	B-2013	82	-	-	2016
[19]	RF	SG	84	66	39	2015
[27]	RF with Morph.	B-2014	81	66	71	2012
[5]	RF	B-2013	83.8	76	76	2012

2.3 Clustering

Clustering based techniques are also used for the detection and localization of abnormalities within brain MR images. Following are most recent researches based on clustering technique for brain tumor segmentation using MR images.

Singh et al. [29] proposed an unified approach of fuzzy C-Mean and level-set segmentation algorithm to delineate the brain tumor and region of interest (ROI). Their proposed method comprised of two steps. At first step, fuzzy c-mean applied to divide the brain MR images into different clusters. At second step, level-set algorithm applied to initial contour in order to delineate the abnormal regions in MR images. Their cascaded strategy that comprised of distinct approaches obtained promising results.

Eman et al. [6] proposed a novel combination of k-Mean and fuzzy C-Mean (KIFCM) for brain tumor segmentation. Their proposed method comprised of five steps: pre-processing, clustering, extraction, contouring, and segmentation. At pre-processing step, median filter employed to reduce random noise and brain surfer was used to extract the brain area. At clustering step, KIFCM applied to obtain the clustering results. At extraction and contouring step, they first applied threshold and then smoothed the thresholded output with de-noising operation. Finally, level-set was employed to contour the tumor area in MR image. Their proposed method was computationally efficient and outperformed k-Mean, Expectation Maximization and fuzzy C-Mean individually.

Kaya et al. [14] proposed a PCA based technique for brain tumor segmentation using only T1-weighted MR images. Five common PCA based techniques were used to reduce the dimensionality of feature vector in k-Means and fuzzy

C-Mean clustering algorithm. The reconstruction error and Euclidean distance errors was used to evaluate their method. The Probabilistic PCA outperformed others four PCA method.

Verma et al. [32] proposed a method for brain tumor segmentation, which was based on mountain clustering technique. They improved the mountain clustering algorithm in their proposed research; their method was comparable with well-known k-Means and fuzzy C-Mean clustering techniques based on the cluster entropy. Their method performed well and presented minimum average entropy (0.000831) against k-Means (0.000840) and fuzzy C-Mean (0.000839) algorithm.

Selvakumar et al. [28] proposed a combined fuzzy C-Mean and k-Mean clustering method to segment the brain tumor pictured in MR images. In preprocessing phase, skull striping and de-nosing operations were performed. At the segmentation step, first brain tumor was segmented uing k-Mean, then the segmented result was proceeded to fuzzy C-Mean to perform final segmentation of tumor regions. Their proposed method extracted the features using these two clustering algorithm that was further used for approximate reasoning. Their method showed the shape and exact spatial location of the tumor, which enhanced to diagnosing of infected tumorous brain region. They used amount of the area as evaluation metric that was calculated from clusters to find the stage.

2.4 Deep Learning (DL) Techniques

We classified the CNN based techniques for brain tumor segmentation in two categories: 2D-CNN and 3D-CNN.Which are discussed below.

2D-CNN Techniques

Chang et al. [9] proposed a fully CNN based architecture for brain tumor segmentation with Hyperlocal Local Features Concatenation (HLFC). The fully convolutional network had five convolutional layers with nonlinear activation functions. Bilinear interpolation unsampled the output of last convolution layer to the original input sized image. Hyperlocal feature concatenation reintroduced the original data in the network by concatenation operation across channels, which was used to produce the segmentation map with two narrow size (3×3) convolutional filters after concatenating hyperlocal features and bilinear interpolated image. They validated the proposed method on BRATS-2016 dataset. Their method was computationally efficient and able to complete segmentation task on an optimized GPU in less than one second.

The patch wise learning of CNN model ignore the contextual information of the whole image. To overcome this issue, Lun et al. [18] presented a CNN based fully automated method for brain tumor segmentation using MR images. Their method incorporated with multi modalities MR images and used global features based CNN model. Local regions were in the form of central pixel labeled patch, which was extracted by cropping image with 30×30 kernel. These patches were used for training the model. They adopted a re-weighting scheme for loss layer in CNN to elevate the imbalance label problem. Their method was validated on

BRATS-2015 dataset. Their global features based re-weighting method outperformed the prior patch wise learning method.

Pereira et al. [23] proposed an automated CNN based method for brain tumor segmentation using four MR weighted modalities (T1, T1c, T2 and FLAIR). They used fixed size 3 × 3 convolutional kernels. The fixed size convolutional kernel enabled them to develop a deeper CNN with more non-linearities applied and fewer weightes. For intensity correction, they used N4ITK method. They proposed separated architectures for High Graded Glioma (HGG) and Low Graded Glioma(LGG). They validated their method on BRATS-2013 and BRATS-2015 dataset. The deeper CNN model with fixed smaller size convolutional kernels obtained highest dice similarity on all sub-tumoral compartments.

Xiao et al. [33] segmented brain tumor in MR images using DL. They integrated the stacked denoising auto-encoder into segmentation procedure. In preprocessing phase, the patches of each MRI were extracted and to obtained the gray level sequence of image patches. The deep learning based classification model use extracted gray level image patches as input and performed classification. Then classification results were mapped on to the binary image. In post-processing phase, they used morphological filters to smoothened the edges and filling the gaps in tumor region. After post processing, they got fined tumor segmentation results.

Havaei et al. [12] proposed a cascaded two-pathway CNN architecture by extracting smaller and larger sized patches at the same time respectively. The cascaded CNN processed smaller and larger contextual details of the central pixel simultaneously. Patches of size 33 × 33 and 65 × 65 were extracted for local and global pathway of CNN to classify the label of the central pixel respectively. Their novel two pathways CNN architecture processed the local and global detail of the central pixel and obtained near state-of-the-art results.

Davy et al. [10] proposed a fully automated brain tumor segmentation approach based on local structure prediction with CNN and k-Means. Similarly, Rao et al. [24] extracted multi plane patches around each pixel and trained four different CNNs each taking input patches from a separate MRI modality image. Outputs of the last hidden layers of those CNNs are then concatenated and used as feature maps to train a random forest (RF) classifier.

Table 3. 2D-CNN based techniques for brain tumor segmentation. All the results are presented in dice similarity (Complete, Core, and Enhanced tumor regions) and accuracy, shown in % unit. SG stands for self-generated, morph. stands for morphological filters, where B- is used for BRATS.

Ref.	Technique	Dataset	Comp.	Core	Enhan.	Acc.	Year
[9]	HLFC based CNN	B-2016	87	81	72	-	2016
[18]	SegNet	B-2015	75	77	76	-	2016
[23]	2D-CNN	B-2015	78	65	75	7.6	2016
[33]	CNN with morph.	SG	-	-	-	98.04	2016
[12]	Cascaded 2 pathway-CNN	B-2013	88	79	0.73	-	2017

3D-CNN Techniques

Kamnitsas et al. [13] proposed 3D-CNN architecture for brain tumor segmentation, named DeepMedic. DeepMedic was previously presented for lesion segmentation. Moreover, their 3D-CNN architecture was extended with residual connection to investigate the lesion effects. BRATS-2015 dataset was used for validating their method. Their results showed that the DeepMedic with residual connection performed well then the simple DeepMedic CNN architecture.

Kayalibay et al. [15] worked on segmentation of medical imaging data using 3D-CNN. Their proposed method was the combination of 3D convolutional filters and CNN. They applied their method to segment brain tumor in MR images. Moreover, they also employed their method to segment bones in hand MR images. Their method was trained and tested on each modality separately. The fused the modality wise CNN output which helped to obtained promising results. Their method obtained comparable dice similarity score on the BRATS-2013 and BRATS-2015.

Yi et al. [34] segmented glioblastoma using three-dimensional CNN. The convolutional layers in their architecture were combined with the Difference of Gaussian (DoG) filters to perform the 3-dimensional convolution operation. They used BRATS-2015 dataset that contained 274 tumor samples. They used dice similarity to measure the quality of the segmentation of their proposed method.

Urban et al. [30] proposed a 3D-CNN architecture for the multi-modal MR images for glioma segmentation. Multi-modality 3D patches, basically cubes of voxels, extracted from the different brain MRI modalities are used as inputs to a CNN to predict the tissue label of the center voxel of the cube. The proposed network achieved promising results across three tumors but due to the 3D-CNN that was based on 3D patches the overall network computational burden was more when compared to 2D patches based 2D-CNN methods.

Nie et al. [22] proposed fully automated 3D-CNN based technique for brain tumor segmentation using T1, Diffusion Tensor Imaging (DTI) and functional MRI (fMRI) MR modality. Different pre-processing technique applied on each MR modality i.e. intensity normalization were applied on T1, tensor modelling applied on DTI and for fMRI frequency specific BOLD (blood oxygen level-dependent) fluctuation power were calculated. They used 3D-CNN as features extractor and SVM performed the final prediction.

3 Performance Evaluation and Discussion

We created a separate table for each category that is proposed for brain tumor segmentation. Each table further presents the detailed results of techniques used under same category. For instance, Tables 1, 2, 3, and 4 presents the detailed results of techniques based on SVM, RF, 2D-CNN, and 3D-CNN respectively. Furthermore, the tables also summarize their research technique, dataset and performance measurement matrices that is used to quantify technique's effectiveness. Different technique are using different performance measure criteria. Some of these are dice similarity, precision, accuracy, total error, and entropy.

Table 4. 3D-CNN based techniques for brain tumor segmentation. All the results are presented in dice similarity (Complete, Core, and Enhanced tumor regions) and accuracy, shown in % unit. SG stands for self-generated, morph. Stands for morphological filters, where B- is used for BRATS.

Ref.	Technique	Dataset	Comp.	Core	Enhan.	Acc.	Year
[13]	DeepMedic with RC	B-2015	89.6	76.3	72.4	-	2016
[15]	3D-CNN	B-2015	85	72	61	-	2017
[34]	3D-CNN	B-2015	89	76	80		2016
[30]	CNN with DoG	B-2015	87	77	73	-	2016
[22]	3D-CNN with SVM	SG	-	-	-	89.9	2014

Conventional ML techniques relied heavily on the handcrafted features that attempt to model the domain knowledge. On the other hand, DL techniques have the unique capability to learn increasingly complex hierarchy of features in extremely unsupervised fashion, which were impossible to represent with handcrafted features. These additional features basis enabled the DL based technique to outperform conventional ML techniques, Particularly, Convolutional Neural Network (CNN) performed well on a verity of vision and biomedical image segmentation tasks.

Another ML technique that is extensively used for brain tumor segmentation are clustering based techniques.These technique are semi-automated and requires neurologist intervention in order to generate the fully clustered tumor regions. However, these technique are heavily dependent spatial information distribution of MR images (i.e. intensity, texture etc.). Such dependencies often cause to deviate the result of the clustering based techniques.

3D-CNN based techniques attempt to model three dimensional details of the three dimensional MR images using 3D convolutional kernels and obtained promising results. However, there is a disadvantage of using 3D-CNN for medical image segmentation task. Because in medical field, there is always a challenge of scarcity of labeled data. In order to address this challenge, 2D-CNN architectures that process 2D patches are introduced. The 2D-CNN based techniques obtained comparable results to 3D-CNN. However, 2D-CNN only process the local contextual information that result in decreased accuracy compared to 3D-CNN.

4 Conclusion

In this paper, we presented the comprehensive review of ML based techniques used for brain tumor segmentation. We classified the ML techniques into two broad categories: conventional and deep learning based techniques. The conventional techniques are further classified based on ML algorithms that are being used. While deep learning based technique are classified into 2D-CNN and 3D-CNN techniques. However, CNN based technique outperformed conventional

ML based technique and obtained state-of-the-art results on brain tumor segmentation. 3D-CNN based techniques are far demanding computationally and inappropriate for biomedical image segmentation task. On the other hand, 2D-CNN architectures are specifically designed for medical segmentation task that address the limited available data challenge and are less demanding computationally. Furthermore, 2D-CNN method also achieved comparable results to 3D-CNN.

References

1. The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3) - Adit Deshpande - CS Undergrad at UCLA ('19)
2. Tumor Types - National Brain Tumor Society
3. MICCAI 2014 challenge on multimodal brain tumor segmentation. In: Proceedings of the MICCAI-BRATS, Boston, Massachusetts (2014)
4. MICCAI challenge on multimodal brain tumor segmentation. Proceedings MICCAI-BRATS Workshop, 2016 (2016)
5. Abbasi, S., Tajeripour, F.: Detection of brain tumor in 3D MRI images using local binary patterns and histogram orientation gradient. Neurocomputing **219**, 526–535 (2017)
6. Abdel-Maksoud, E., Elmogy, M., Al-Awadi, R.: Brain tumor segmentation based on a hybrid clustering technique. Egypt. Inf. J. **16**(1), 71–81 (2015)
7. Arikan, M., Fröhler, B., Möller, T.: Semi-automatic brain tumor segmentation using support vector machines and interactive seed selection. In: Proceedings of the MICCAI-BRATS, Boston, Massachusetts, pp. 1–3 (2014)
8. Farahani, K., Kalpathy-Cramer, J., Kwon, D., Menze, B.H., Reyes, M.: MICCAI 2015 challenge on multimodal brain tumor segmentation. In: Proceedings of the MICCAI-BRATS, Munich, Germany (2015)
9. Chang, P.D.: Fully convolutional neural networks with hyperlocal features for brain tumor segmentation. In: Proceedings MICCAI-BRATS Workshop, pp. 4–9 (2016)
10. Havaei, M., et al.: Brain tumor segmentation with deep neural networks. In: proceedings of BraTS-MICCAI (2014)
11. Ellwaa, A., et al.: Brain tumor segmantation using random forest trained on iteratively selected patients. International Workshop on Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. Lecture Notes in Computer Science, pp. 129–137. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-55524-9_13
12. Havaei, M., et al.: Brain tumor segmentation with deep neural networks. Med. Image Anal. **35**, 18–31 (2017)
13. Kamnitsas, K., et al.: Deepmedic on brain tumor segmentation. BrainLes 2016. LNCS, vol. 10154. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-55524-9_14
14. Kaya, I.E., Pehlivanlı, A.Ç., Sekizkardeş, E.G., Ibrikci, T.: PCA based clustering for brain tumor segmentation of T1w MRI images. Comput. Methods Programs Biomed. **140**, 19–28 (2017)
15. Kayalibay, B., Jensen, G., van der Smagt, P.: CNN-based Segmentation of Medical Imaging Data. arXiv preprint arXiv:1701.03056 [cs.CV], January 2017
16. Szilagyi, L., Lefkovits, L., Lefkovits, S.: Brain tumor segmentation with optimized random forest. In: Proceedings MICCAI-BraTS Workshop, pp. 30–34 (2016)

17. Louis, D.N., et al.: The 2016 world health organization classification of tumors of the central nervous system: a summary. Acta Neuropathol. **131**(6), 803–820 (2016). https://doi.org/10.1007/s00401-016-1545-1
18. Lun, T.K., Hsu, W.: Brain tumor segmentation using deep convolutional neural network. In: Proceedings of BRATS-MICCAI (2016)
19. Wiest, R., Reyes, M., Meier, R., Knecht, U.: CRF-based brain tumor segmentation: alleviating the shrinking bias. In: Crimi, A., Menze, B., Maier, O., Reyes, M., Winzeck, S., Handels, H. (eds.) Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. LNCS. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-55524-9_10
20. Menze, B., Jakab, A., Bauer, S., Reyes, M., Prastawa, M., Van Leemput, K.: MICCAI 2012 challenge on multimodal brain tumor segmentation. In: Proceedings of the MICCAI-BRATS, Nice, France (2012)
21. Menze, B., Jakab, A., Reyes, M., Prastawa, M.: MICCAI 2013 challenge on multimodal brain tumor segmentation. In: Proceedings of the MICCAI-BRATS, Nagoya, Japan (2013)
22. Nie, D., Zhang, H., Adeli, E., Liu, L., Shen, D.: 3D deep learning for multi-modal imaging-guided survival time prediction of brain tumor patients. In: Ourselin, S., Joskowicz, L., Sabuncu, M.R., Unal, G., Wells, W. (eds.) MICCAI 2016. LNCS, vol. 9901, pp. 212–220. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46723-8_25
23. Pereira, S., Pinto, A., Alves, V., Silva, C.A.: Brain tumor segmentation using convolutional neural networks in MRI images. IEEE Trans. Med. Imaging **35**(5), 1240–1251 (2016)
24. Rao, V., Sarabi, M.S., Jaiswal, A.: Brain tumor segmentation with deep learning. In: Proceedings of MICCAI Multimodal Brain Tumor Segmentation Challenge (BraTS), pp. 56–59 (2015)
25. Rathi, V.P., Palani, S.: Brain tumor MRI image classification with feature selection and extraction using linear discriminant analysis. arXiv preprint arXiv:1208.2128 (2012)
26. Reddy, K.K., Solmaz, B., Yan, P., Avgeropoulos, N.G., Rippe, D.J., Shah, M.: Confidence guided enhancing brain tumor segmentation in multi-parametric MRI. In: 2012 9th IEEE International Symposium on Biomedical Imaging (ISBI), pp. 366–369. IEEE (2012)
27. Reza, S., Iftekharuddin, K.M.: Improved brain tumor tissue segmentation using texture features. In: Proceedings of MICCAI-BRATS Challenge on Multimodal Brain Tumor Segmentation (2014)
28. Selvakumar, J., Lakshmi, A., Arivoli, T.: Brain tumor segmentation and its area calculation in brain MR images using K-mean clustering and FUZZY C-mean algorithm. In: International Conference on Advances in Engineering, Science and Management (ICAESM), pp. 186–190. IEEE (2012)
29. Singh, P., Bhadauria, H.S., Singh, A.: Automatic brain MRI image segmentation using FCM and LSM. In: 3rd International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), pp. 1–6. IEEE (2014)
30. Urban, G., Bendszus, M., Hamprecht, F., Kleesiek, J.: Multi-modal brain tumor segmentation using deep convolutional neural networks. In: Proceedings of MICCAI BraTS (Brain Tumor Segmentation) Challenge, Winning Contribution, pp. 31–35 (2014)

31. Vaishnavee, K.B., Amshakala, K.: An automated MRI brain image segmentation and tumor detection using SOM-clustering and proximal support vector machine classifier. In: 2015 IEEE International Conference on Engineering and Technology (ICETECH), pp. 1–6. IEEE (2015)
32. Verma, N.K., Gupta, P., Agrawal, P., Cui, Y.: MRI brain image segmentation for spotting tumors using improved mountain clustering approach. In: Applied Imagery Pattern Recognition Workshop (AIPRW), pp. 1–8. IEEE (2009)
33. Xiao, Z., et al.: A deep learning-based segmentation method for brain tumor in MR images. In: 2016 IEEE 6th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS), pp. 1–6. IEEE, October 2016
34. Yi, D., Zhou, M., Chen, Z., Gevaert, O.: 3-D convolutional neural networks for glioblastoma segmentation. arXiv preprint arXiv:1611.04534 [cs.CV], November 2016
35. Zhang, N., Ruan, S., Lebonvallet, S., Liao, Q., Zhu, Y.: Kernel feature selection to fuse multi-spectral MRI images for brain tumor segmentation. Comput. Vis. Image Underst. **115**(2), 256–269 (2011)

Sentiment Analysis on Predicting Presidential Election: Twitter Used Case

Nedaa Baker Al Barghuthi[1](✉) and Huwida E. Said[2](✉)

[1] Higher Colleges of Technology, Sharjah, United Arab Emirates
Nedaa.Albarghuthi@hct.ac.ae

[2] Zayed University, Dubai, United Arab Emirates
Huwida.Said@zu.ac.ae

Abstract. Twitter is a popular tool for social interaction over the Internet. It allows users to share/post opinions, social media events, and interact with other political and ordinary people. According to Statista web site 2019 statistical report, it estimated that the number of users on Twitter had grown dramatically over the past couple of years to research 300 million users. Twitter has become the largest source of news and postings for key presidents and political figures. Referring to the Trackalytics 2019 report, the recent president of the USA had posted 4,000 tweets per year, which indicates an average of 11–12 tweets per day. Our research proposes a technique that extracts and analyzes tweets from blogs and predicts election results based on tweets analysis. It assessed the people's opinion and studied the impact that might predict the final results for the Turkey 2018 presidential election candidates. The final results were compared with the actual election results and had a high accuracy prediction percentage based on the collected 22,000 tweets.

Keywords: Twitter API · Virtualization · Data mining · Sentiment analysis · Tweets · Election · Positive polarity · Negative polarity

1 Introduction

At present, social media provides a massive amount of data about users and their social interaction. This data plays a useful role in policy, health, finance, and many other sectors as well as predicting future events and actions. In a relatively short period, social media has gained popularity as a tool for mass communications and public participation when it comes to political purposes and governance. Obtaining a successful data forecast helps to understand the limitations of predictability in social media and avoid false expectations, misinformation, or unintended consequences. Rapid dissemination of information through social networking platforms, such as Twitter, enables politicians and activists to broadcast their messages to broad audiences immediately and directly outside traditional media channels.

During the 2008 US election, Twitter had used as an essential tool to influence the results of Barack Obama's campaign [1]. Obama's campaign succeeded in using Twitter as a campaign and gaining more followers. Accordingly, more voters had elected during the 17 months of the election period. The campaign published 262 tweets and gained about 120,000 new followers. As a result of the Twitter campaign, all major candidates and political parties nowadays use social media as an essential tool to convey their messages [1]. This paper aims to study the expected patterns of political activity and Twitter campaigns in this context. The Turkish presidential election is used as a case scenario to extract and analyze the final campaign results. The second section of the paper highlights the related work on election prediction around the Asia region.

Furthermore, the third section addresses the work approach and methods adopted. The results section was organized in Sect. 4 of the paper. A final discussion and conclusion section was listed in Sect. 5.

2 Related Work on Election Prediction Based on Twitter

The Twitter tool is used as a vital component to assess the needs of social media among users [2]. It has many benefits, such as social and political sentiments, measuring the popularity of political and social policies, marketing products, extracting positive and negative opinions, discovering recent trends, and detecting product popularity. Besides all, sarcastic and non-sarcastic tweets are also crucial for detecting sentiments. There have been many studies examining the characteristics and features of the political discourse of Twitter during the US 2016 election. The objective of these studies is to develop a robust approach that can be applied to extract data from Twitter to predict the results of the upcoming elections. According to [1, 3, 4] Twitter is a popular application on social media. During the US presidential election of 2016, people used social media to express their admiration or dissatisfaction for a particular presidential candidate. The authors measure how these tweets expressed and compared to the poll data. The authors used an Opinion Finder Lexicon dictionary and the Naive Bayes Machine Learning Algorithm to measure the sense of tweets related to politics. In another research paper [3], there were 3,068,000 tweets contains 'Donald Trump' text, and 4,603,000 contains 'Hillary Clinton' text collected within 100 days before the election period (see Fig. 1). Also, [1] has collected 3 million tweets within the same period. In both types of research, authors have manually labeled the collected tweets. Also, another technique was used to automate label based on the hashtag content/address. The authors concluded that Twitter had become a more reliable environment. By observing the tweets during 43 days before

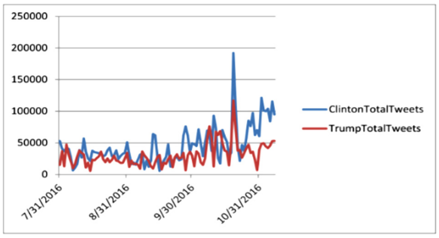


Fig. 1. Tween volume chart

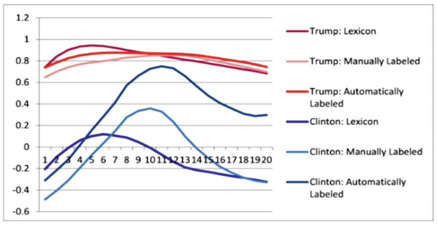


Fig. 2. Correlation Coefficient for Trum and Clinton Tweets 43 days before election using a Moving Average of k days

the election period, when using "moving average technique," it was noticed that 94% correlation was recorded with poll data used as illustrated (see Fig. 2).

The lexicon dictionary [3] contains 1600 positive and 1200 negative words. The result of applying this analysis was shown in Figs. 3 and 4.

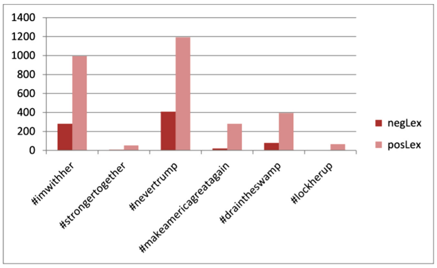


Fig. 3. Lexicon sentiment for Trump hashtags

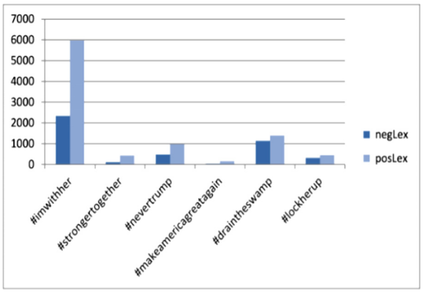


Fig. 4. Lexicon sentiment for Clinton hashtags

National Language Toolkit (NLTK) was used in the Naïve Bayes algorithm. There were five hundred negative and five hundred positive tweets labeled in this algorithm. Two key search words ('Hillary Clinton') and ('Donald Trump') were used. The criteria of this algorithm and the analysis results are shown in Figs. 5 and 6. Authors in [1] have used a machine learning approach for automatic tweet labeling. Convolution Neural Network (CNN) was performed using Python and Tensorflow. The collected tweets are trained by 140 sentiment data set. After that, tweets were labeled as a positive or negative sentiment. The election dataset is used for evaluation and predicting the votes by using two algorithms and sentiment analysis. The predicted votes were compared with the polling statistics collected during the election period, and it shows that the expected result was matched with the actual voting result [5].

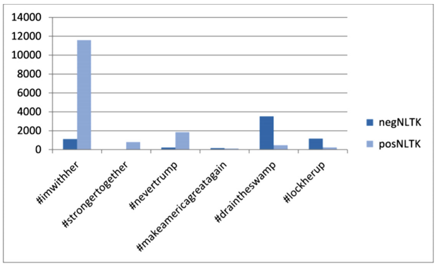


Fig. 5. NLTK sentiment for Clinton hashtags

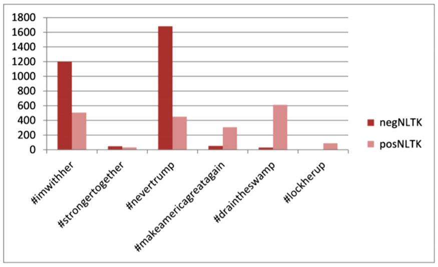


Fig. 6. NLTK sentiment for Trump hashtags

Moreover, [6] produced an analytical text on the American Elections in 2016. In this research, three million tweets were collected within 21 days before and after Election Day. Emotion analysis was used to examine the user's sentiment behavior regarding his Twitter profile and its associated features. Both research papers have studied the topic of

Twitter and discuss its relevance to news and events. More often the SentiStrength is used for sentiment score calculation. The collected tweets were cleaned based on messages referring to the keyword search terms ‘election 2016’, ‘Hillary Clinton’ or ‘Donald Trump.’ Furthermore, [6] has conducted several hypotheses to study and discuss the characteristics of Twitter’s political user behavior, views, and other discussions. The result of the analysis showed that the majority of feelings of the collected tweets were negative for both candidates.

There is a sign of unpleasant sensations of the latest elections. Also, it was discovered from the analytic results that few numbers of tweets posted during debate sessions and were mostly re-tweeted. Authors in [7] provided a novel method to facilitate data mining related to the participant’s opinion with the support of linguistic analysis and sentiment ratings. It was used to identify the sentiment score level of political involvers in Pakistan. According to the literature, this original paper was recognized as the first one that studied the sentiment analysis on social media. The authors have introduced a new technique for mining opinion in a political context. Two classifications analyses were applied among the collected data using the Bayes Naïve probability distribution and Support Vector Machine (SVM) algorithms. The result of applying this approach was shown in Fig. 8. The sentiment analysis was performed on the Sentiment viz web-interface that provides a graphical analysis of the different levels of sentiments categories, as shown below (see Fig. 7). Figure 8 also shows that SVM performed better than Naïve Bayes algorithm and gave higher accuracy.

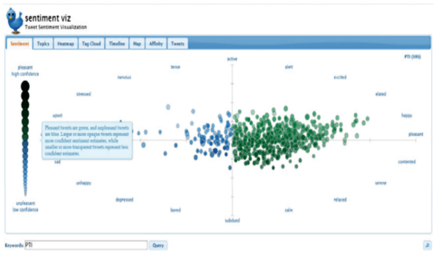


Fig. 7. Sentiment classification defined over the keyword “PTI”

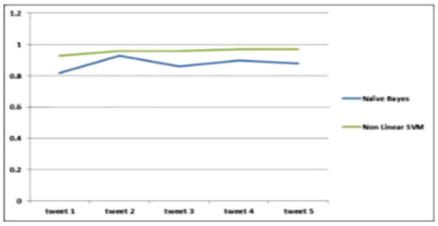


Fig. 8. Comparisons of SVM and Naive Bayes

Figure 9 shows an output display of the invented model. This model has an option to display the data in a visual environment. Authors of [7] have used this model to represent the negative opinion for Pakistan People’s Party (PPP) among different places and cities. From the display of Fig. 9, it was noticed that Lahore city has the highest negative opinion level about PPP. The highlighted information leads to many facts.

Furthermore, other parties can see this fact from a different angle. This leads to an easy comparison and clarity concerning PPP. The proposed model was tested on the Pakistan Election. Sentiment analysis was performed and shows that Lahore has the highest negativity percentage among the country, as it is listed in Fig. 9. SVM was also used for determining the polarity of the sentiment. SVM is based on features of the data, and label polarity of the tweet’s sentiment, whether it is positive or negative, otherwise,

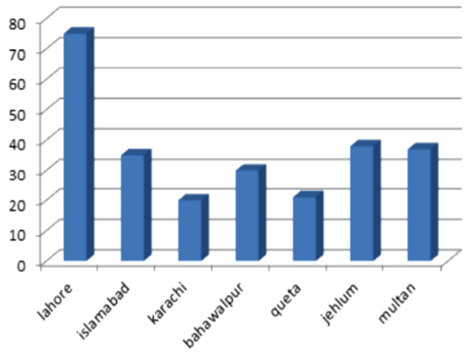


Fig. 9. The magnitude of negativity about PPP in different cities

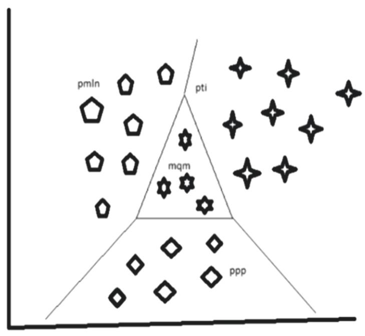


Fig. 10. Working of SVM Model

it will be labeled as neutral, along with the political party name for the posted tweets [9] (see Fig. 10).

The author in [8] has improved the political sentiment analysis on Twitter using a dynamic keyword method. This paper presents a new method for collecting data to predict election results and a way for subject models to extract topics [10]. The RSS aggregator uses news articles and dynamic keywords shared on Twitter at the same time, creating an intelligent prediction method that mainly depends on the size of Twitter. It also attempts to enhance electoral predictions on social media using a dynamic word-based methodology for Delhi Assembly Election 2015. There were two channels for data collected using RSS (see Fig. 11) and Twitter API. Two types of analysis were applied to the collected data (volume and sentiment-based analysis). This analysis is useful to detect the voting activity from the tweets by using keywording searches related to ‘election’ using the LDA-based approach, as shown in (see Fig. 12). This analysis is based on simple opinion and comparison-based opinion. It is evident from the graph (see Fig. 10) that SVM techniques give good results.

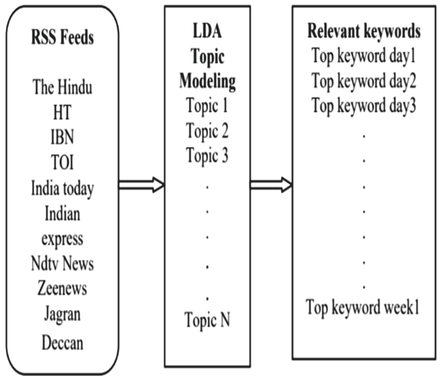


Fig. 11. Selection procedure for topics and keywords from RSS feeds

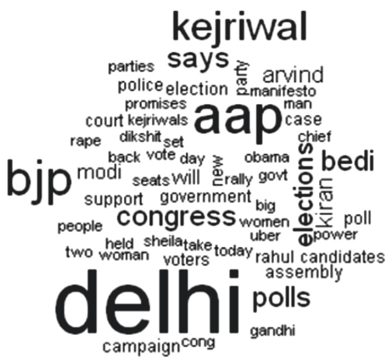


Fig. 12. Tag cloud showing the most relevant keywords in RSS

3 Methodology

Two datasets were generated using the Twitter application for data collection. Each dataset has 22000 tweets. The collected tweets were extracted before the Turkish Presidential Election period. The election took place during the period of 22nd and 24th June 2018. Spyder Python development environment was used for the data mining and sentiment analysis process (see Table 1). Multiple stages were performed on this dataset during the analysis processes. Step 1 of the process involves translating the text portion in the datasets to the English Language text format. We selected the python spyder programming script and the translator parser.

Table 1. Tools and Software packages used in the Tweet Analysis

Tools used	Software package used
Twitter API	Tweepy package
Spyder Python Development Environment	Json and CSV packages
Anaconda Navigator Environment	Textblob package
SentiStrength Classifier Toolbox	Naive Bayes Analyzer package
Twitter API	Wordcloud package

Furthermore, step 2 investigates the dataset's text conversion into a lower case text format. A filtering process was applied to eliminate the unwanted text. This filter was used to remove redundant data for better extraction. A custom stopwords lexicon dictionary that contained English and Turkish words were used to filter the dataset's text. The output text from this filter was extracted from any punctuations symbols, URLs, and hashtags. Besides, all the re-tweeted texts were excluded from the filtered text. After that, word and sentence tokenize process was completed. Then, each tweet was divided into two parts; discrete words and sentences. The output of the text was further extracted for better sentiment analysis performed on the output text using a Naïve Bayes Classifier. It classifies the sentiment polarity level using three levels of polarity, positive, negative, and neutral.

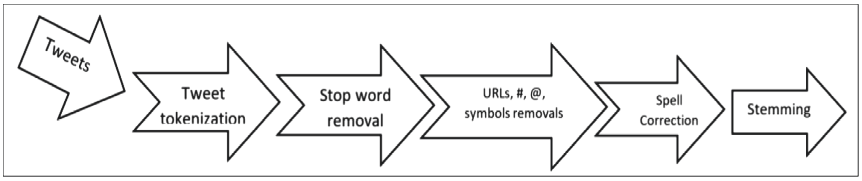


Fig. 13. Typical scenarios of pre-processing on standards text and Tweets

Finally, several data processes were implemented on the dataset text using the word cloud and word frequency algorithms. Multiple visual presentation and measurements were generated and displayed. Finally, the actual election results were compared with the Twitter sentiment analysis results and evaluated the precision percentage of the overall effect. Figures 13 and 14 show the processes performed on the data sets.

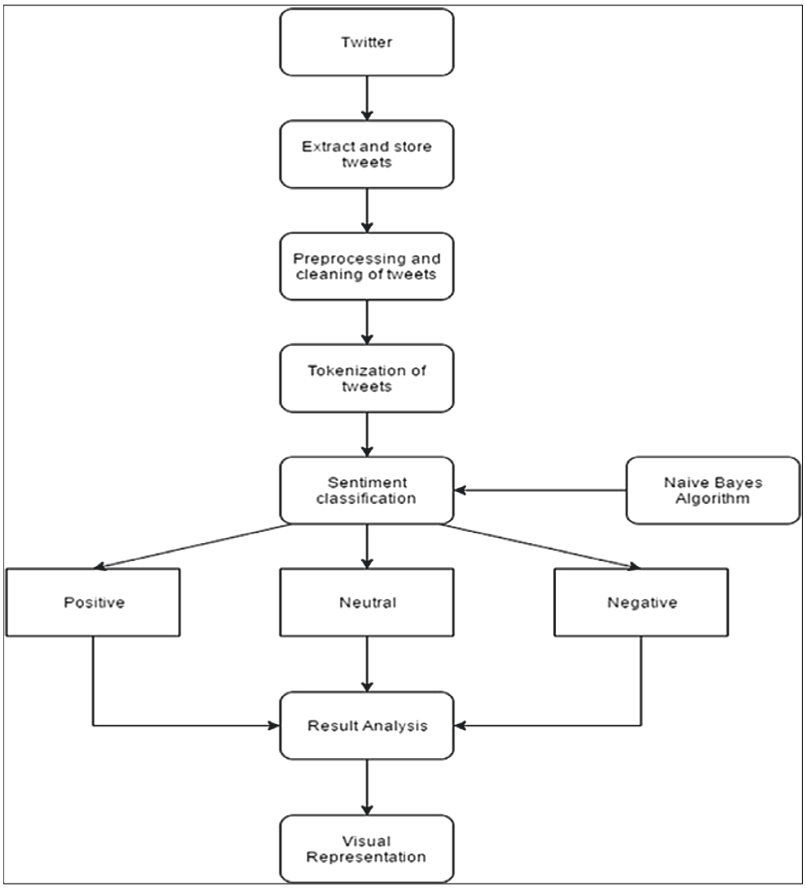


Fig. 14. Flowchart sentiment analysis algorithm

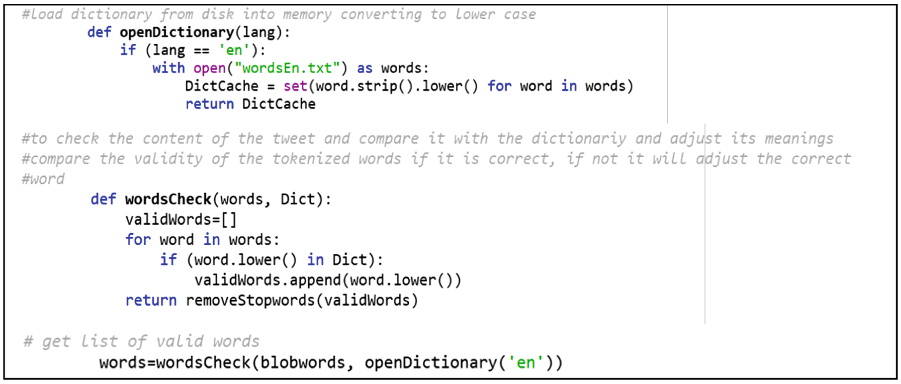
4 Sentiment Analysis Process

a. **Tokenize each text into words**
Each tweet will be tokenized and split into separated words as showed in Fig. 15.

```
#tokenize words
        blobwords=[lstlst for sublist in blobs for lstlst in sublist]
```

Fig. 15. Tokenize each text into words

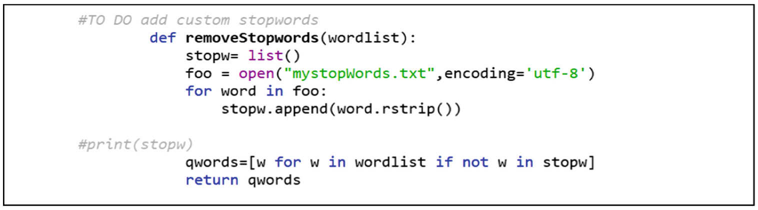
b. **Converting text to lower case and spell check process**
Each tokenized word is converted into a lower case. After that it will be validated by comparing it with an English word list dictionary. If there is a typo mistake, it will be corrected and returned back to its dataset as shown in Fig. 16.



```
#Load dictionary from disk into memory converting to lower case
        def openDictionary(lang):
            if (lang == 'en'):
                with open("wordsEn.txt") as words:
                    DictCache = set(word.strip().lower() for word in words)
                    return DictCache

#to check the content of the tweet and compare it with the dictionariy and adjust its meanings
#compare the validity of the tokenized words if it is correct, if not it will adjust the correct
#word
        def wordsCheck(words, Dict):
            validWords=[]
            for word in words:
                if (word.lower() in Dict):
                    validWords.append(word.lower())
            return removeStopwords(validWords)

# get list of valid words
          words=wordsCheck(blobwords, openDictionary('en'))
```

Fig. 16. Converting text to lower case and spell check process

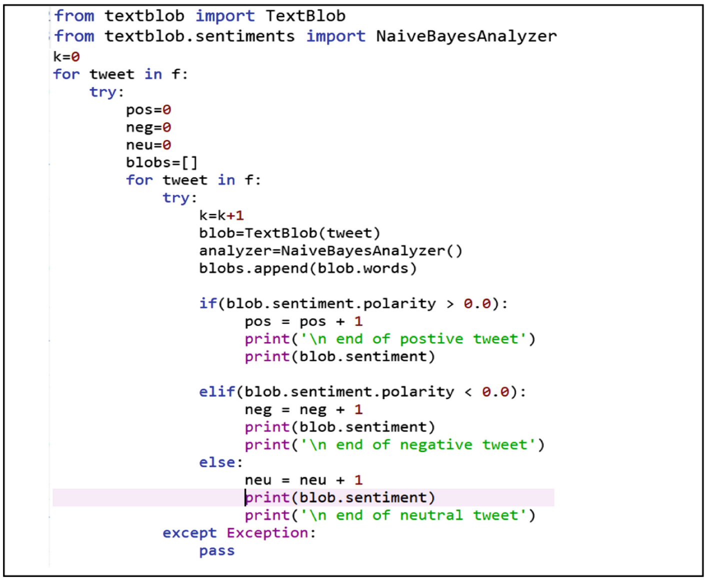
c. **Create a custom stop words and remove the unwanted text from the main data set**
Two custom stopwords dictionaries were created using *utf-8* encoding. Stopwords list for English and another one for Turkish. These stopwords lists have most frequent words that people usually use. It should be removed from the dataset before and after the data processing as shown in Fig. 17.



```
#TO DO add custom stopwords
        def removeStopwords(wordlist):
            stopw= list()
            foo = open("mystopWords.txt",encoding='utf-8')
            for word in foo:
                stopw.append(word.rstrip())

#print(stopw)
            qwords=[w for w in wordlist if not w in stopw]
            return qwords
```

Fig. 17. Create a custom stop words and remove the unwanted text from the main dataset

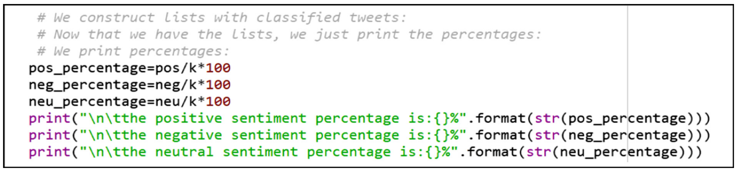
d. **Sentiment Polarity Analysis Process**
TextBlob and NaiveBayesAnalyzer packages were installed and imported. Naïve Bayes Sentiment classifier is used to word sentiment classification. This function returns a score between [1-1 and 1]. *If sentiment score* > 1, this means that the tweet has a positive sentiment. *Else if sentiment score* < 1, this means that the tweet has a negative sentiment. Else if sentiment score is equal to zero, it is considered that this tweet has a neutral sentiment as shown in Fig. 18.



```
from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer
k=0
for tweet in f:
    try:
        pos=0
        neg=0
        neu=0
        blobs=[]
        for tweet in f:
            try:
                k=k+1
                blob=TextBlob(tweet)
                analyzer=NaiveBayesAnalyzer()
                blobs.append(blob.words)

                if(blob.sentiment.polarity > 0.0):
                     pos = pos + 1
                     print('\n end of postive tweet')
                     print(blob.sentiment)

                elif(blob.sentiment.polarity < 0.0):
                     neg = neg + 1
                     print(blob.sentiment)
                     print('\n end of negative tweet')
                else:
                     neu = neu + 1
                     print(blob.sentiment)
                     print('\n end of neutral tweet')
            except Exception:
                pass
```

Fig. 18. Sentiment polarity analysis process

e. **Measure the overall polarity sentiment level for each elector**
The overall polarity level percentage (positive, neutral and negative) is calculated for each elector (see Fig. 19).



```
 # We construct lists with classified tweets:
 # Now that we have the lists, we just print the percentages:
 # We print percentages:
pos_percentage=pos/k*100
neg_percentage=neg/k*100
neu_percentage=neu/k*100
print("\n\tthe positive sentiment percentage is:{}%".format(str(pos_percentage)))
print("\n\tthe negative sentiment percentage is:{}%".format(str(neg_percentage)))
print("\n\tthe neutral sentiment percentage is:{}%".format(str(neu_percentage)))
```

Fig. 19. Measure the overall polarity sentiment level for each elector

5 Observation

The datasets were examined using the above-mentioned approach. Exciting results were generated. The testing was introduced using two election candidates. Those are *Mr. Regep Tayyip Erdogan* (*RT Erdogan* - candidate #1) and his competitor *Mr. Muharrem Inci (Inci* - candidate #2*).* Similar experimental procedures were applied to these two candidates' datasets. Table 2 shows samples of positive, negative and neutral opinions for candidate #1 using the Naïve Bayes Classifier.

The results showed that candidate #1 has more positive sentiment votes. A "48%" positives sentiments reflect that more people are pleased to elect him. The competitor has

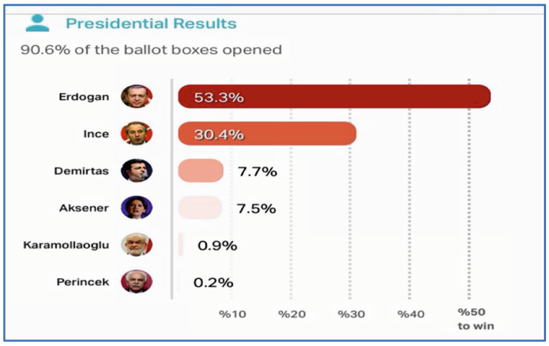
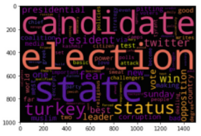
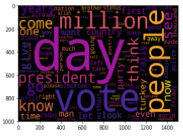
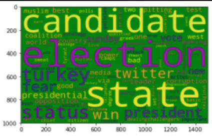
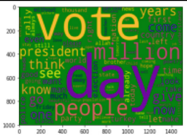


Fig. 20. Actual Turkey presidential elections 2018 results [11]

Table 2. Comparison between actual Turkey presidential final result vs. Twitter predicted result

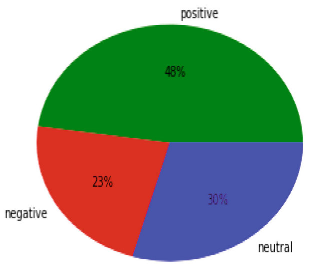
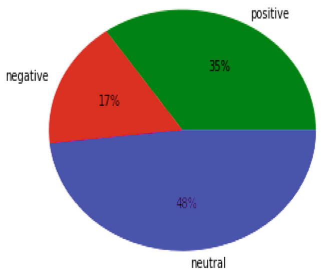
Turkey presidential elections 2018	Candidate 1	Candidate 2
Actual final election results on June 25, 018	53.3%	30.4%
Predicted election results through the proposed Twitter sentiment analysis on June 20–24, 2018	48.0%	35.0%
Accuracy error percentage	**9.94%**	**11.84%**

Table 3. The word cloud results from the collected dataset during June 22nd–24th, 2018, for both electors. It shows the top 100 words that are frequently repeated and the most 100 positive words about both electors

Related Tweets to:	Relevant tweets words - Candidate 1	Related tweeted words - Candidate 2
Top 100 frequent words on Twitter related to each elector		
Top 100 positive words related to elector		

“35%” positive sentiment votes. Moreover, 23% of tweets have negative feelings votes for candidate one whereas 30% recorded tweets were neutral votes. As for candidate 2, 35% of the tweets were identified as positive sentiments; on the other hand, only 17% of the tweets listed as negative feelings. For this candidate, the neutral votes were recorded as 48% of the overall tweets. These results are comparable to the official election results listed, as shown in Fig. 20 and Tables 2, 3 and 4.

Table 4. Positive, negative and neutral sentiment

Sample	Candidate 1		Candidate 2	
Positive	**Sentiment :** polarity=0.42023809523809524, subjectivity=0.559077380952381	Zeeshan Haider, iamshanichadhar, 2018-06-22 03: 55,0,0," ""I do not know what he say in this video but every word is important for every Muslim he is real Hero as a leader	**Sentiment:** polarity=0.4, subjectivity=0.625	Now look, where are the citizens of **a great country** when the president is in the palaces? how is this country represented from this profession? just open your mind and look at the state of the country. how does it look?
Negative	**Sentiment** polarity=-0.6999999999999998, subjectivity=0.6666666666666666	Vicki Andrada, BlindNewsGirl, 2018-06-22 03: 27,0,1," ""I fear if # Erdogan realizes by doing it the democratic way is not working," he will go really violent on the Turkish people, worse than he I have already had it .. Perhaps then he'd go too far for Turks," but it's a very scary situation	**Sentiment**: polarity=-0.46875, subjectivity=0.8	No trolls! It is no? Peace, theft, contradiction with Allah
Neutral	Sentiment (polarity=0.0, subjectivity=0.0)	Helmi Deris, helmi_deris, 2018-06-22 03: 37,0,0," ""In Turkey's pious heartland"," faith in Erdogan trumps economy worries - World | The Star Online	**Sentiment:** polarity=0.0, subjectivity=0.0	The AKP lie beneath the party feet that it enlists.
Sentimental Polarity Percentage				

6 Analysis and Discussion

The actual Turkey Presidential results are shown in Fig. 14. It indicates that candidate 1 received 53.3% from the overall election votes. In contrast, candidate 2 received 30.4% of the total electoral votes. As shown in Table 3, the proposed sentiment analysis model was successful in predicting the final results of the presidential election votes. This model predicted that candidate 1 has positive support from the community with 48% of the predicted twitter votes. When the actual voting result was received during the election period, it shows that there is 9.94% accuracy. Nowadays, social media plays an essential role in sentiment analysis, mainly if it is used in predicted votes before running the actual election results [12, 13]. This proposed model can also be useful for politicians, especially if they need to understand their followers and supporters. As shown in Table 4, the word cloud has a critical impact, in collecting the negative sentiments from the community and visualize them. By doing so, the voters can have a useful resource from these data to improve their proposed strategy seeking more supporters. On another hand, this model can be used in several cases, such as to study the activities of the elector's followers on social media to understand their thought, opinion, and subjectivity.

7 Conclusion and Future Work

In this research, a systematic sentimental analysis of collected tweets towards predicting presidential election results was introduced. The data was collected from tweets blogs using Twitter API aggregator. A polarity sentimental analysis and word cloud counts were applied to the collected tweets. Text files were created to compare the tweets. In this context the classification (i.e. Naïve Bayes Classifier) of the tweets was based on three levels; positive, negative and neutral. When the experiment was executed we were able to generate a word cloud for the most repetitive words found in the tweets retriever. The classified data was visualized by the pie chart and word cloud based on the sentiment scores. As future work, the authors would like to develop an in-depth analysis of the emotional behavior for election candidates. Besides we would like to enhance our prediction approach by adapting multiple classifiers and compare the outcome with the real results.

References

1. Heredia, B., Prusa, J., Khoshgoftaar, T.: Exploring the effectiveness of Twitter at polling the United States 2016 presidential election. In: IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), pp. 283–290. IEEE (2017). https://doi.org/10.1109/cic.2017.00045
2. Abuaiadah, D., Rajendran, D., Jarrar, M.: Clustering Arabic tweets for sentiment analysis. In: IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), pp. 449–456. IEEE (2017). https://doi.org/10.1109/aiccsa.2017.162
3. Joyce, B., Deng, J.: Sentiment analysis of tweets for the 2016 US presidential election. In: Undergraduate Research Technology Conference (URTC), IEEE MIT, pp. 1–4. IEEE (2017)
4. Huberty, M.: Can we vote with our tweet? On the perennial difficulty of election forecasting with social media. Int. J. Forecast. **31**(3), 992–1007 (2015). https://doi.org/10.1016/J.IJFORECAST.2014.08.005
5. Yaqub, U., Atluri, V., Chun, S.A., Vaidya, J.: Sentiment based analysis of tweets during the US presidential elections. In: Proceedings of the 18th Annual International Conference on Digital Government Research, pp. 1–10 (2017)
6. Ussama, Y., Chun, S.A., Atluri, V., Vaidya, J.: Analysis of political discourse on Twitter in the context of the 2016 US presidential elections. Gov. Inf. Q. **34**(4), 613–626 (2017). https://doi.org/10.1016/J.GIQ.2017.11.001
7. Gull, R., Shoaib, U., Rasheed, S., Abid, W., Zahoor, B.: Pre processing of Twitter's data for opinion mining in political context. In: Proceedings of 20th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, KES 2016 (2016)
8. Jain, S., Sharma, V., Kaushal, R.: PoliticAlly: finding political friends on Twitter. In: IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1–3 (2015). https://doi.org/10.1109/ANTS.2015.7413659
9. Sharma, U., Yaqub, R., Pabreja, S., Chun, A., Atluri, V., Vaidya, J.: Analysis and visualization of subjectivity and polarity of Twitter location data. In: Proceedings of the 19th Annual International Conference on Digital Government Research Governance in the Data Age - DGO 2018, pp. 1–10 (2018)
10. Ceron, A., Curini, L., Iacus, S.M., Porro, G.: Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. New Media Soc. **16**(2), 340–358 (2014)

11. Hurriyet Daily News, As it happened: Erdoğan re-elected president, “People’s Alliance” wins majority at parliament (2018). http://www.hurriyetdailynews.com/turkey-election-live-updates-vote-counting-starts-as-polls-close-across-turkey-133726. Accessed 16 August 2018
12. Burnap, P., Gibson, R., Sloan, L., Southern, R., Williams, M.: 140 characters to victory?: Using Twitter to predict the UK 2015 general election. Electoral. Stud. **41**, 230–233 (2016)
13. Vinay, J., Shishir, K.: Towards prediction of electronic outcomes using social media. Int. J. Intell. Syst. Appl. **12**, 20–28 (2017)

Convolutional Neural Network U-Net for Trypanosoma cruzi Segmentation

Allan Ojeda-Pat[1](✉), Anabel Martin-Gonzalez[1], and Roger Soberanis-Mukul[2]

[1] Computational Learning and Imaging Research, Facultad de Matematicas, Universidad Autonoma de Yucatan, Merida, Mexico
allan.ojedaa@gmail.com, amarting@correo.uady.mx

[2] Computer Aided Medical Procedures, Technical University Munich, Munich, Germany
rodsom22@gmail.com

Abstract. Chagas disease is a mortal silent illness caused by the parasite *Trypanosoma cruzi* that affects many people worldwide. A blood test is one of the preferred methods to get an accurate diagnosis of the disease but takes a long time and requires too much effort from the experts to analyze blood samples in the search of the parasites presence. Therefore, it is very useful to have an automatic system to detect the parasite in blood sample microscopic images. In this paper we present a deep learning method to segment the *T. cruzi* parasite on blood sample images. We implemented a convolutional neural network based on the U-Net model and we trained it with different loss functions to get accurate results. We report an F2 value of 0.8013, a recall value of 0.8702, a precision value of 0.6304 and a Dice score value of 0.6825.

Keywords: Chagas disease · Deep learning · U-Net segmentation

1 Introduction

According to the World Health Organization, Chagas disease is a potentially dangerous disease caused by the parasite *Trypanosoma cruzi* [1]. This parasite resides mainly in Latin America and is transmitted to humans by the feces of triatomine insects, colloquially known as kissing bugs or insect ‘pic’. It is estimated that there are about 7 to 8 million people infected with Chagas disease in the world, most of which are in Latin America [1, 2], and more than 10 thousand deaths per year are attributed [2]. It is estimated that more than 30% of infected patients suffer from heart problems, and above 10% suffer from digestive, neurological or mixed problems [1]. Many of these symptoms do not occur until years after infection with the parasite. It is important to mention that during the initial phase of Chagas disease, which occurs in the first two months after infection, many parasites circulate in the blood. After the initial phase, the chronic phase continues. In this phase the parasites lodge mainly in the heart and in the digestive muscles, making it difficult to locate them in a common blood sample through a microscope [3]. It is important to carry out the detection studies during the initial phase for effective diagnosis and prompt treatment of the disease [4]. A blood test is essential to detect and treat Chagas disease in time. The most common tests for the diagnosis of Chagas disease include ELISA test and by blood smear inspection. A blood smear involves placing a drop of blood on a slide. The sample is stained for later analysis under the microscope. Detection of parasites through microscopic inspection of blood smears is a very common and convenient technique, but sometimes it is a process that requires a lot of time and effort when many blood samples have to be analyzed [5–9]. In recent years, Machine learning and Deep learning techniques have been rapidly used in the medical imaging research area to diagnose a disease [10], and these techniques could help to reduce the effort of getting a quick diagnosis of the Chagas disease. With an automated segmentation system, experts could easily confirm if it is a proper identification/diagnosis of the *T. cruzi* parasite and experts can get relevant data about sizes and shapes for further use, all of this with saving time. The purpose of this article is to present an automated method based on deep learning to segment the *T. cruzi* parasite on blood samples images as an alternative to a tedious common manual inspection. With this segmentation method, experts can easily confirm and verify if the parasite is present in the blood sample images and collect information from the segmented parasites.

Following in Sect. 2 we describe the state of the art of clinical and machine learning works for Chagas disease detection. Section 3 explains the image dataset and manual work. Section 4 details the method developed with the U-Net model and the details of the training. Section 5 details the results of the training and the conclusions are presented in Sect. 6.

2 Previous Work

In the state of the art for Chagas disease diagnosis, we can separate the works on two groups: the clinical methods and the Machine learning-based methods.

2.1 Clinical Methods

A clinical method requires the use of a specialized portable test for the diagnosis of the Chagas disease. One of them is called Chagas Stat-Pak presented in [11] which has 99.6% sensitivity and 99.9% specificity; this performance is comparable with the obtained with an ELISA test [3]. Another method that requires a portable test is called Chagas Detect Plus [12]. Although these portable tests for Chagas detection are quick and precise, it is mandatory to confirm with a manual or serological method to get an accurate diagnosis [12]. An automated system based on Machine learning and/or Deep learning could replace or assist the manual confirmation method, and in case of a positive result, the patient can begin the appropriate treatment.

2.2 Machine Learning Methods

In the machine learning and computer vision area, some studies have been reported for the detection of the *Trypanosoma cruzi* parasite in blood sample images [3, 4, 8, 13].

Detection Methods. In 2013, Uc-Cetina et al. [4], propose the detection of the *T. cruzi* parasite in blood samples images using an algorithm based on the analysis of the Gaussian discriminant. In the classification of a new set of input pixels, they calculate the probabilities of the Bayesian model, and the highest probability will indicate the dominant class of the sample. As the final step, they implemented a search algorithm to find possible candidates of parasites to extract and classify the input vector. The performance rates they reported are false negatives 0.0167, false positives 0.1563, true negatives 0.8437 and true positives 0.9833.

In 2013, in the work of Soberanis-Mukul et al. [8], an automated algorithm is proposed to detect Chagas disease using different segmentation and classification techniques. In the first step, a binary segmentation is applied using a Gaussian classifier. By calculating the probabilities of the Bayesian model, it is segmented groups of pixels that could represent a parasite. Later, the segmented images go through a K-Nearest-Neighbors classifier [16] previously trained to get a binary classification: it is a *T. cruzi* parasite or not. The authors reported a sensitivity value of 98% and a specificity value of 85%.

In 2015, in the work of Uc-Cetina et al. [3], it is compared two classify algorithms: AdaBoost and Support Vector Machine (SVM) for the detection of *T. cruzi* parasite on blood sample images. The proposed algorithm consists of the detection of possible parasites through training of a binary AdaBoost classifier feed up with features extracted by using specific Haar templates designed for the morphology of the *T. cruzi* parasite. A post-processing step is applied using an SVM classifier to assist in discard false positives. The authors compared the proposed AdaBoost + SVM algorithm with a single SVM algorithm. The classification using AdaBoost + SVM obtained the best results reporting a sensitivity value of 100% and a specificity value of 93.25%.

Segmentation Methods. In 2014, Soberanis-Mukul [13] made a comparison of three classifiers: SVM, AdaBoost and Artificial Neural Networks (ANN) for detection and segmentation of the *T. cruzi* parasite. His methodology consists of three steps: computing of superpixels, extraction of optimal features and the training of the classifiers (ANN with backpropagation algorithm, AdaBoost with perceptron assembly and SVM). Superpixels are formed by sets of pixels that describe continuous regions delimited throughout the image. The proposal to use groups of pixels allows features to be extracted considering the neighborhood and its relationship within a group. In the experimentation stage, the author compares his three classifiers with three classifiers of the state of the art (Gaussian and Bayes Classifier and the algorithm proposed in [14]). In the end, the author reports that the Gaussian classifier had the lowest mean square error with 0.18568, followed by the SVM classifier proposed by the author who obtained an error of 0.22635, and finally, the ANN classifier with an error of 0.361.

3 Dataset

The database consists of 974 color images on RGB format of size 2560 × 1920 pixels.

The images were taken from blood samples of mice infected with the *T. cruzi* parasite. The blood samples show blood cells and *T. cruzi* parasites highlighted with special ink

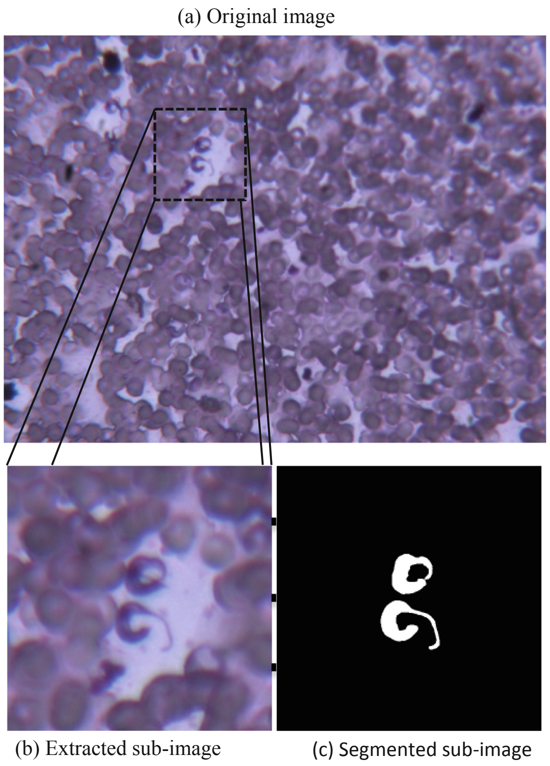


Fig. 1. Ground truth generation of a sub-image. In (a) we extract (b), and (c) is the manual segmentation of (b).

for easy recognition. Some of the original images contain multiple parasites, and other images do not contain any parasites. As a first step to create our dataset, we cropped 940 sub-images of size 512 × 512 pixels with contain parasites presence, and 60 sub-images of size 512 × 512 pixels without parasites taken from the original database.

To train the segmentation network with ground truth examples, it is necessary to get a manual binary segmentation of every sub-images. An example of the manual work can be observed in Fig. 1. In the segmented sub-images, white pixels represent a parasite and black pixels represent the background.

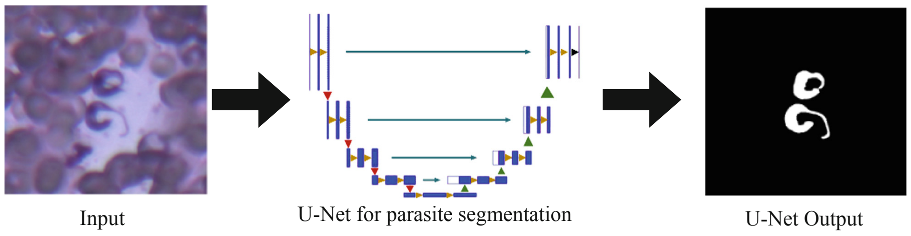


Fig. 2. The proposed method for *T. cruzi* parasite segmentation

4 Method

This section introduces the method used to segment the *T. cruzi* parasite.

In computer vision, the goal of semantic segmentation is to label each pixel of an image with its corresponding class. The desired output is an image where all the pixels are classified using, for example, a convolutional neural network (CNN). We can define the above as

$$p = g(x) \tag{1}$$

where x is an RGB input image, p is a segmentation output and *g()* is a CNN. Figure 2 shows the complete visual method.

4.1 U-Net Model

The U-Net model is a fully convolutional neural network, i.e., an architecture with only convolutional layers, to perform semantic segmentation of mainly biomedical images. The U-Net was developed by Ronneberger et al. [15]. The architecture has two paths. The first path is the contraction or the encoder path. The encoder captures the context in the image through a stack of convolutional and down-sampling layers. The second path is the symmetric expanding path or the decoder. The decoder enables precise location of the classified pixels through up-convolutions. More technically, the contraction path consists of the repeated application of two 3×3 convolutions, each one followed by a Rectified Linear Unit (ReLU) activation function, and one 2×2 max-pooling operation after two convolutional layers. The previous process is repeated five times increasing the number of convolutional filters (64, 128, 256, 512 and 1024). To get fine details in the segmentation map, multi-scale features are combined by connecting corresponding resolutions in the contracting and expanding path. The expanding path consists of the repeated application of an up-sampling operation (through one 2×2 transposed convolution) followed by a concatenation operation (with the corresponding feature map from the contracting path), and two 3×3 convolutions with a ReLu activation function. The previous process is repeated four times decreasing the number of convolutional filters (512, 256, 128 and 64), and at the end, a 1×1 convolution (with two filters) is applied to get the final segmentation map. The entire network has 23 convolutional layers. For the purpose of this work, we used a variation of the U-Net architecture by applying zero-padding to every convolution operation to maintain the input image dimensions; and a 1×1 convolution (with one filter) applied to the last layer to get the binary segmentation map, obtaining a CNN with 24 layers. Figure 3 shows the proposed U-Net model.

4.2 Training

The complete dataset consists of 1000 images, each one with its corresponding ground truth image. The training set consists of 600 images of which 564 images contain *T. cruzi* parasites and 36 images do not contain any parasite. The validation set consists of 200 images of which 188 images contain *T. cruzi* parasites and 12 images do not contain

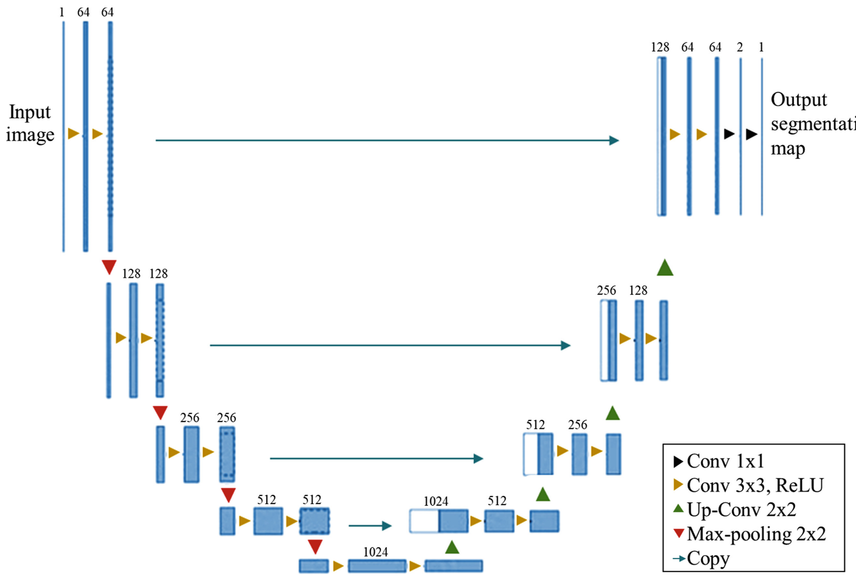


Fig. 3. U-Net model proposed with an RGB input image and a segmentation map as output.

any parasite. The final test set consists of 200 images with the same distribution as the validation set.

As shown in Table 1, the dataset has a very high-class imbalance, i.e., most of the pixels of each image correspond to the background and only a small number of pixels correspond to *T. cruzi* parasite class. During training, if there is a high-class imbalance on the dataset, the network tends to predict the most common class [9]. The output image may contain most pixels of the predominant background class.

Table 1. Pixels distribution per class of the dataset

Class	*T. cruzi* parasite	Background
Pixels	1.8%	98.2%

To mitigate this problem, we train the network with a custom loss function called Weighted Binary Cross-Entropy Loss (WBCE) [16]. This loss function is defined in terms of the Binary Cross-Entropy Loss (BCE):

$$BCE(p, \hat{p}) = -\left(plog(\hat{p}) + (1 - p)\log(1 - \hat{p})\right) \tag{2}$$

where $Y = 1$ is the *T. cruzi* parasite class, $Y = 0$ is the background class, $P(Y = 1) = p$ and $P(Y = 0) = 1 - p$ are the values of the ground truth for both classes. The predicted probability for class 1 is given by the sigmoid function $P\left(\hat{Y}=1\right)=\frac{1}{1+e^{-x}}=\hat{p}$ and for class 0 is $P\left(\hat{Y}=0\right)=1-\hat{p}$.

The Weighted Binary Cross-Entropy Loss [16] is defined as:

$$WBCE(p,\hat{p})=-(\beta plog(\hat{p})+\alpha(1-p)\log(1-\hat{p})) \tag{3}$$

where β represents the proportion of the *T. cruzi* parasite class and α the proportion of the background class.

We used data augmentation to increase the number of training samples by applying random transformations such as zooming, rotations, vertical and horizontal flips to both the input and ground truth images. Thus, we increase 3 times the training dataset size.

We conduct two experiments. The first experiment consists on training the U-Net model for 48 epochs with the Weighted Binary Cross-Entropy Loss function, steps per epoch value of 600, He initialization [17], Adam Optimizer as an optimizer algorithm with a learning rate of 1e−4, batch size of 2, $\beta=9.0$ and $\alpha=1.0$ (given more weight to *T. cruzi* parasite class). It takes approximately 14 h to train on the Google CoLab platform [18] using Keras framework.

The second experiment consists on training the U-Net model for 62 epochs with the regular Binary Cross-Entropy Loss, steps per epoch value of 300, He initialization, Adam Optimizer with a learning rate of 1e-4 and bach size of 2. It takes approximately 15 h to train on the same platform.

We use the Dice Coefficient value as a metric to monitor how good the predictions are compared with the ground truth. The Dice Coefficient is defined in [15] as:

$$Dice\,Coefficient=\frac{2*|P\cap Y|}{|P|+|Y|} \tag{4}$$

where P is the segmentation map returned by the U-Net and Y is the ground truth image. The Dice Coefficient compares the pixel-wise agreement between P and Y. The metric returns a value close to one if the predictions of the network are close enough.

5 Results

The learning curves of the WBCE loss function for the first experiment is shown in Fig. 4. Figure 5 shows its mean Dice Coefficient values of every epoch during training, where on epoch 41, we get the higher Dice Coefficient value of 0.6735 for the validation set.

As we can see in Fig. 4, the model reaches a local minimum in the validation loss and it cannot decrease its validation error in contrast with the training error. Thus, we can conclude that there is a slight overfitting (Fig. 4) that does not affect the segmentation performance as we can see a higher Dice Score value on epoch 41 (Fig. 5).

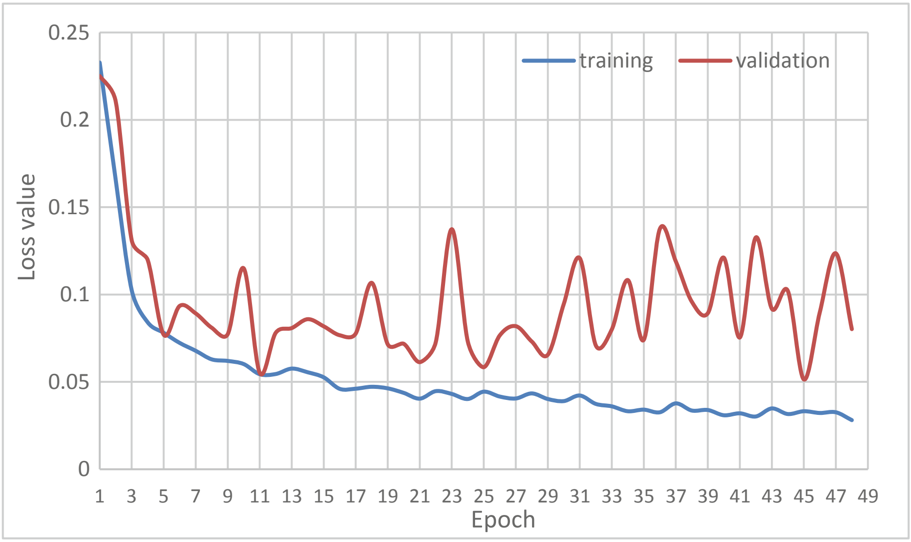


Fig. 4. Training curves of U-Net with WBCE Loss

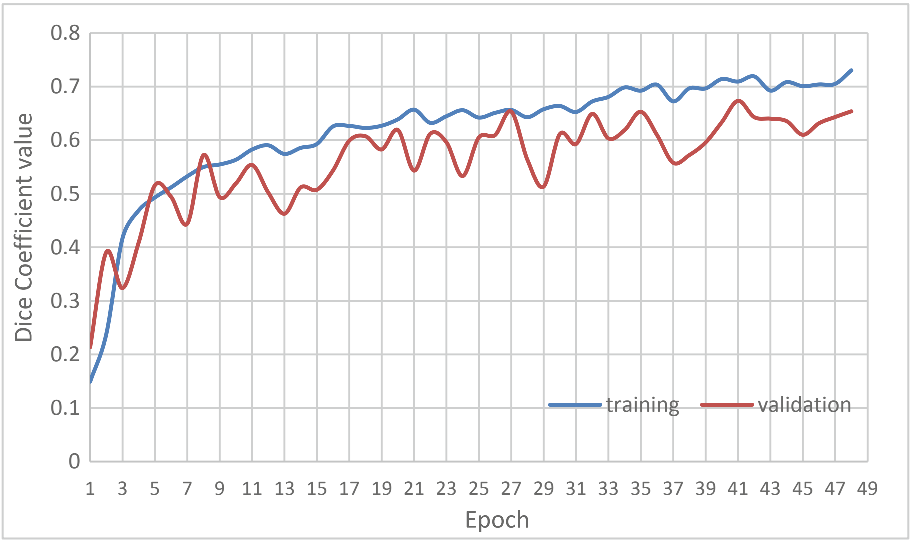


Fig. 5. Dice Coefficient values of U-Net with WBCE Loss

Figure 6 shows the learning curves of the BCE loss function for the second experiment and Fig. 7 shows its mean Dice Coefficient values.

On epoch 54, we get the higher Dice Coefficient value of 0.733 for the validation set. In Fig. 6, we can see that the model cannot decrease the validation loss after 62 epochs and tends to a horizontal asymptote, we conclude that it has to be result of the model reaching a local minimum.

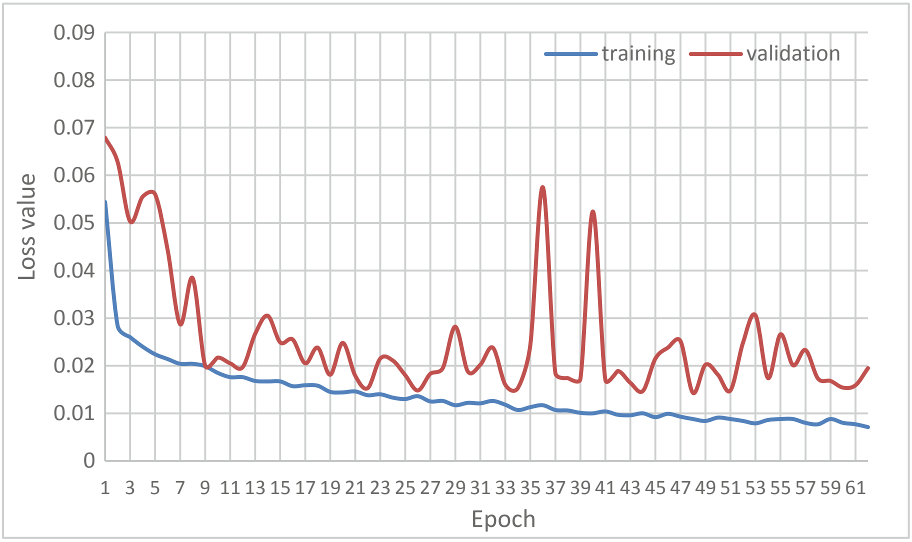


Fig. 6. Training curves of U-Net with BCE Loss

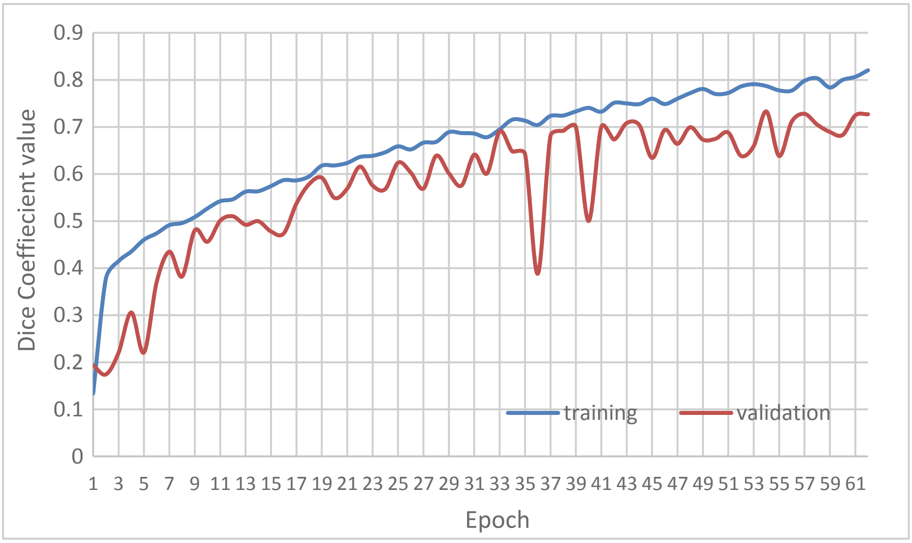


Fig. 7. Dice Coefficient values of U-Net with BCE Loss

We compare the results obtained with the F2 score. The F2 score is a good choice as we need to measure how good the final binary segmentation is in terms of how many correct positive classifications (regions of pixels of *T. cruzi* parasite) are made. The F2 score measures the accuracy of a test in terms of the precision and recall values [19].

The F2 score is defined as:

$$F2\,score = 5 \cdot \frac{precision \cdot recall}{(4 \cdot precison) + recall} \tag{4}$$

The precision is the ratio of true positives (TP) to all predicted positives, true positives and false positives (TP + FP), as in:

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

The recall is the ratio of true positives (TP) to all actual positives, true positives and false negatives (TP + FN), as in:

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

Table 2. Results obtained from test set

Model	Loss function	Loss value	Dice Coefficient value	Recall	Precision	F2 score
1	WBCE	0.0588	0.6825	0.8702	0.6304	0.8013
2	BCE	0.0148	0.7072	0.7808	0.7671	0.7714

As we are focus on the *T. cruzi* parasite segmentation, we are interested in the correct classification of class 1 pixels. Analyzing the results presented in Table 2 for the final test set, we can assume that model 1 trained with the WBCE performs better on classifying positive pixels based on the F2 score. A high recall value means that the model is focused on predicting as many correct positive pixels as possible to keep the shape of the parasite, although this could result on more false positives, as we can see in the lower precision value. Even though the Dice Coefficient value is slightly better on the model 2, it only means that the pixel-wise agreement is good to both classes of pixels (*T. cruzi* and background pixels), but its lower F2 score means that it does not predict the positive pixels as well as the model 1 does.

In Fig. 8, we can observe some of the qualitative results for both experiments. As we can see, the predictions are very closed to the ground truth, although there could be a room of improvement for tricky images.

In Fig. 9, we can see five examples of wrong predictions or not-complete segmentations returned by the U-Net. Analyzing these resulting images, we conclude that shape orientation and morphology of the *T. cruzi* parasites presented in Fig. 9c to e are not the most common on the training set, and as a result the U-Net does not learn how to properly segment them. In Fig. 9b, the reason for a black segmentation output is because the input image poor focus, producing a low parasite staining; so, the U-Net model cannot segment it correctly.

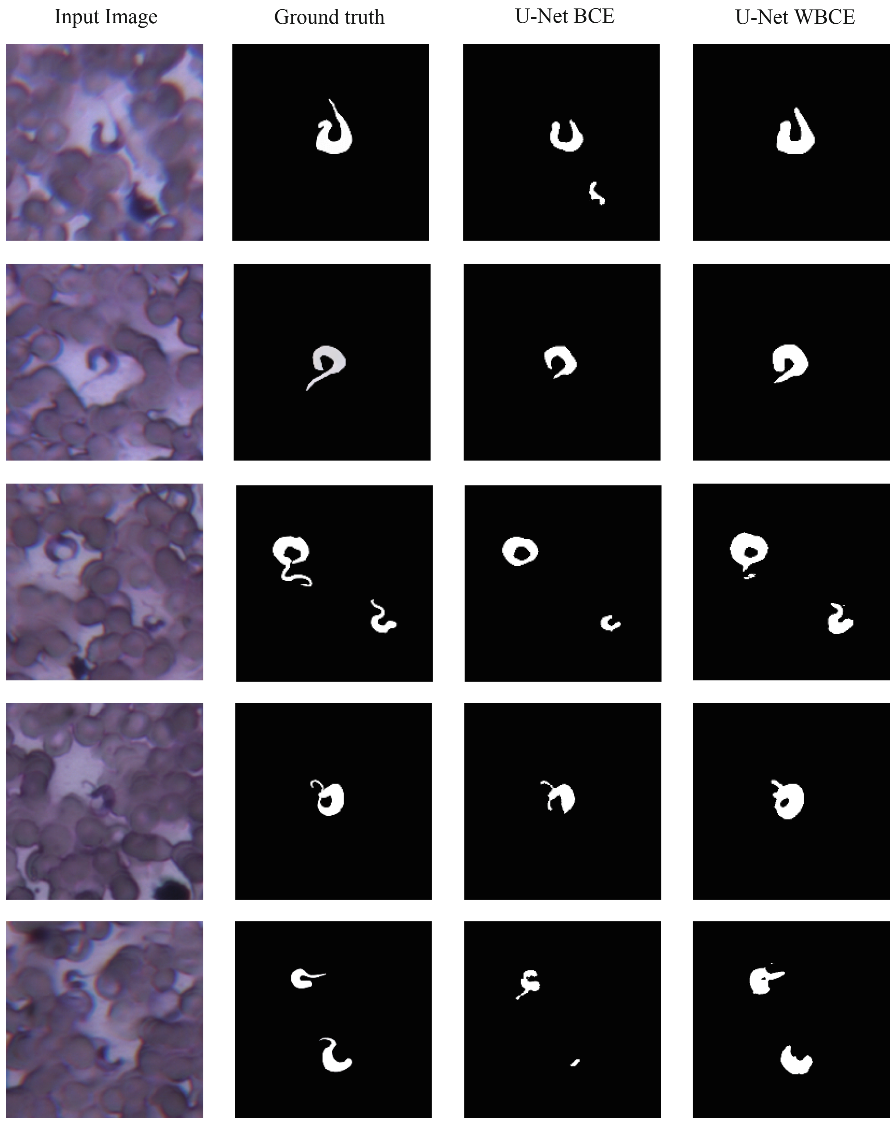


Fig. 8. Qualitative results of predictions returned by the U-Net

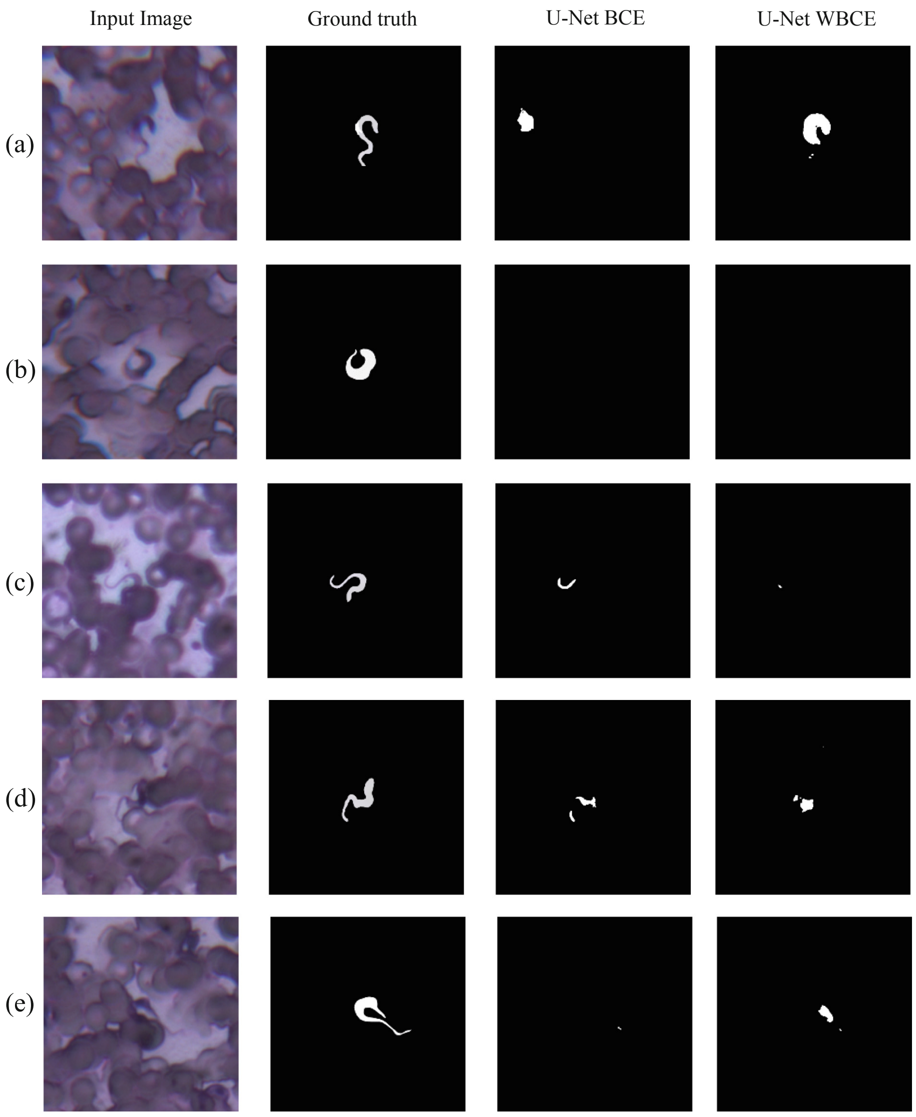


Fig. 9. Qualitative results of wrong predictions returned by the U-Net

6 Conclusions

We proposed an automated method based on deep learning to segment the *T. cruzi* parasite on blood samples images as an alternative to a tired manual visual inspection with a microscope. A proper segmentation can assist experts to easily confirm if the *T. cruzi* parasite is present on the images to get a quick diagnosis of the Chagas disease, saving time and getting relevant information for further use. We chose the U-Net model because it is commonly used for biomedical image segmentation showing satisfactory results. We conduct two experiments with the U-Net model. The first experiment consists on training the U-Net model with the Weighted Binary Cross-Entropy Loss function to reduce the class imbalance between *T. cruzi* parasite class and background class. The second experiment consists on training with the regular Binary Cross-Entropy Loss. We get the best F2 score and recall value on the first experiment. We report an F2 value of 0.8013, a recall value of 0.8702, a precision value of 0.6304 and a Dice score value of 0.6825. In comparison with the state of the art methods, our proposal obtains good results without executing a pre-processing of the input data (as the definition and feature extraction) or a post-processing of the results, obtaining a quick and reliable model to segment the *T. cruzi* parasite. Qualitative results show a good segmentation performance for *T. cruzi* parasite. With these promising results, we have shown that a deep learning model based on the U-Net architecture can achieve a proper *T. cruzi* parasite segmentation in blood sample images.

Acknowledgments. We thank Dr. Hugo Ruiz-Piña from the Centro de Investigaciones Regionales "Dr. Hideyo Noguchi" for providing us with the *T. cruzi* parasite image database.

References

1. W. H. Organization Homepage: Chagas disease American Trypanosomiasis. https://www.who.int/en/news-room/fact-sheets/detail/chagas-disease-(american-trypanosomiasis). Accessed 02 Apr 2019
2. W. H. Organization: Chagas disease American trypanosomiasis. https://www.who.int/chagas/disease/en/. Accessed 22 May 2019
3. Uc-Cetina, V., Brito-Loeza, C., Ruiz-Piña, H.: Chagas parasite detection in blood images using AdaBoost. Comput. Math. Methods Med. **2015** (2015). 13 pages. https://doi.org/10.1155/2015/139681
4. Uc-Cetina, V., Brito-Loeza, C., Ruiz-Piña, H.: Chagas parasites detection through Gaussian discriminant analysis. Abstraction Appl. **8**, 6–17 (2013)
5. Delas Peñas, K., Rivera, P., Naval, P.: Malaria parasite detection and species identification on thin blood smears using a convolutional neural network, pp. 1–6 (2017). https://doi.org/10.1109/chase.2017.51
6. Poostchi, M., Silamut, K., Maude, R., Jaeger, S., Thoma, G.: Image analysis and machine learning for detecting malaria. Transl. Res. **194**, 36–55 (2018)
7. Mehanian, C., et al.: Computer-automated malaria diagnosis and quantitation using convolutional neural networks. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 116–125 (2017). https://doi.org/10.1109/ICCVW.2017.22
8. Soberanis-Mukul, R., Uc-Cetina, V., Brito-Loeza, C., Ruiz-Piña, H.: An automatic algorithm for the detection of Trypanosoma cruzi parasites in blood sample images. Comput. Methods Prog. Biomed. **112**, 633–639 (2013)
9. Górriz, M., Aparicio, A., Raventós, B., Vilaplana, V., Sayrol, E., López-Codina, D.: Leishmaniasis parasite segmentation and classification using deep learning. In: Perales, F.J., Kittler, J. (eds.) AMDO 2018. LNCS, vol. 10945, pp. 53–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94544-6_6
10. Latif, J., Xiao, C., Imran, A., Tu, S.: Medical imaging using machine learning and deep learning algorithms: a review. In: 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1–5 (2019). https://doi.org/10.1109/icomet.2019.8673502

11. Ponce, C., et al.: Validation of a rapid and reliable test for diagnosis of Chagas' disease by detection of Trypanosoma cruzi-specific antibodies in blood of donors and patients in Central America. J. Clin. Microbiol. **43**, 5065–5068 (2005)
12. Egüez, K., et al.: Rapid diagnostic tests duo as alternative to conventional serological assays for conclusive Chagas disease diagnosis. PLoS Neglect. Trop. Dis. **11**(4), (2017). https://doi.org/10.1371/journal.pntd.0005501
13. Soberanis-Mukul, R.: Algoritmos de segmentación de trypanosoma cruzi en imagenes de muestras sanguineas. Master's thesis, Universidad Autónoma de Yucatán (2014)
14. Soberanis-Mukul, R.: Detección de trypanosoma cruzi en imágenes obtenidas a partir de muestras sanguíneas. Bachelor's thesis, Universidad Autónoma de Yucatán (2012)
15. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, William M., Frangi, Alejandro F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
16. Losses for segmentation. https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/. Accessed 03 Sep 2019
17. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: IEEE International Conference on Computer Vision (ICCV 2015), vol. 1502 (2015). https://doi.org/10.1109/iccv.2015.123
18. Google Colaboratory. https://colab.research.google.com. Accessed 15 Oct 2019
19. What is the F2 score in machine learning? https://www.quora.com/What-is-the-F2-score-in-machine-learning. Accessed 16 Sep 2019

Segmentation of Echocardiographic Images in Murine Model of Chagas Disease

Rafael Viana-Camara[1(✉)], Carlos Brito-Loeza[1], Anabel Martin-Gonzalez[1], and Nidiyare Hevia-Montiel[2]

[1] Computational Learning and Imaging Research Group, Universidad Autónoma de Yucatán, Mérida, Mexico
leafar1314@gmail.com

[2] Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Mexico City, Mexico

Abstract. In this work, we present a methodology for the semiautomatic segmentation of left ventricle of the hearth of mice in echocardiographic images of the murine model for Chagas disease. The methodology presented is based on the active contour model with shape prior. We will show through experimental results the good performance of the model and discuss pros and cons of the methodology.

Keywords: Image segmentation · Chagas disease · Active contours · Shape prior

1 Introduction

Chagas disease is an illness caused by the protozoan parasite Trypanosoma Cruzi. Within the last years, Chagas disease has spread around the globe very rapidly threatening the life of thousands of people. Once infected, the T. Cruzi parasites remain hidden mainly in the heart muscle of the patient, yielding an infection that over the years may cause sudden death from heart failure.

This disease consists of two different phases: initially, the acute phase that lasts for a period of about two months after the infection has been contracted [2], and the second phase also known as chronic phase that may last years.

In the acute phase, a large number of parasites may be observed circulating in the bloodstream and usually, there are no symptoms or they are mild in this phase. Then again, in the chronic phase the parasites are hidden mainly in the heart and digestive muscle causing patients to suffer from cardiac and digestive disorders. Over the years, the infection can cause sudden death from cardiac arrhythmias or progressive heart failure from the destruction of the heart muscle and its innervations [6].

Supported by organization CONACYT, National Council of Sciences and Technology, México.

In order to study and analyze the reactions of the disease, the murine model (use of strains of special mice to study a human disease or condition, as well as the way to prevent and treat it) turns out to be very useful in Chagas disease research [9]. Murine model presents much of the immunological, pathological and physiological characteristics of Chagas disease in humans and the acquisition of samples is relatively easy compared to that of other animals. It is also a low cost procedure and therefore highly attractive for research.

In the present echocardiographic study, for the following up of the strains of infected mice, ultrasound videos of the heart were obtained. Due to hearth anatomy, quantifying the damages caused by the disease is quite difficult due to the very condition of the ultrasonic image. The ultrasound technique used to capture the images, uses high frequency sound waves to provide cross-sectional images of the region under study. This very same technique is popular in almost all medical fields and a variety of clinical situations [7]. This imaging mode is rapidly evolving with significant advances in transducer technology and more sophisticated imaging routines [1].

In spite of recent advances, segmentation of ultrasound images is strongly influenced by the quality of the data and by no means an easy task. There are characteristic artifacts, such as attenuation, mottling, shadows and signal loss that makes difficult to obtain reasonable partitions of the images. Typical complications also arise because the contrast between the areas of interest is often low [5].

In this work a semiautomatic segmentation technique for the left ventricle in echocardiographic images of the murine model for Chagas disease will be shown.

The outline of the paper is as follows. In Sect. 2, we review the foundations of the active contour model without edges, the level set method and shape-prior based model. In Sect. 3, we present how to construct the shape prior for the left ventricle of the hearth of mice, how to select proper parameters for the model and the morphological post processing applied to the results, finally in Sect. 4 we present our conclusions.

2 Active Contour Model with for Image Segmentation with Shape Prior

2.1 Active Contours

An active contour (commonly called Snake) consists of a elastic curve that, placed on an image, starts deforming from an initial form in order to delimit the regions of interest in the scene. Curve deformation is achieved by applying internal forces, intrinsic to the Snake that control the smoothness of the curve, as well as external forces, that pushes the Snake towards the salient characteristics of the image. The active contour property makes deformable models an effective tool in multiple tasks, such as in the analysis of medical images, where the low signal-to-noise ratio makes the results obtained through classical techniques insufficient.

Geometrically, a Snake is a parametric contour $c(s,t) = (x(s,t), y(s,t))$, variable in time and defined in the image plane $(x,y) \in R^2$, where the $x(s,t), y(s,t)$ coordinates of the contour are functions of the parametric variable $s \in [0,1]$, and time t. The contour is supposed to be closed, through boundary conditions. The shape of the contour is expressed by the following energy function E_{total}, which must be minimized in order to determine the shape and final position of the Snake:

$$E_{total} = \int_0^1 \frac{\alpha|c'(s)|^2 + \beta|c''(s)|^2}{2} ds + \int_0^1 g(u(c(s)))ds \tag{1}$$

where and E_{ext} correspond to the terms of internal and external energy, respectively. E_{int} gives the deformation characteristics of the elastic contour and the functions $\alpha(s)$ and $\beta(s)$ determine the degree to which the Snake can be stretched or curved. These functions are useful for manipulating the physical behavior and local continuity of the model. Thus, for example, an increment in the magnitude of $\alpha(s)$ results in increments in the tension of the curve, which tends to eliminate loops or curls by reducing the contour length. On the other hand, an increment in $\beta(s)$ increases the rigidity of the Snake, making it softer in shape and less flexible.

These functions, $\alpha(s)$ and $\beta(s)$, may be dependent of s, curve length, and by adjusting them it is possible to change the characteristics of the Snake in each parametric coordinate. However, most applications specify constant values along the contour for α and β. The external energy function E_{ext}, is derived from the image in such a way that it takes in its smallest values the characteristics that are of interest to us, such as the edges or borders. For this case, $g(u(c(s)))$, denotes a potential scalar function that is defined in the image plane.

2.2 Level Sets

A simple but clever way to track evolving active contours was designed Osher and Sethian [8]. The main idea behind this method is the representation of curves or surfaces as the zero level set of a hyper-surface of higher dimension. For instance, sharp curves on a 2-dimensional space are considered as the level sets of a continuous surface of in the 3-dimensional space. This is, a smooth function $\phi(x,y,t)$ can be defined to represent the surface while the set of definitions $\phi(x,y,t) = 0$ for some values of t may represent the evolving contours. By doing so, the evolution of a curve can be transformed into the evolution of a function of level sets in 3D space.

To obtain such a representation, let $\phi(x,y,t=0)$ be a function of level sets, where the zero level set corresponds to the curve being tracked. By making the curve as the boundary surface, it is possible to define two separated regions, the region inside the curve and region outside the curve. A signed distance function (SDF) is then defined within the surface. This is

$$\phi(x,y,t=0) = d \tag{2}$$

where, d represents the shortest distance from the point x on the surface and the curve. Throughout the process of evolution of the curve its points are adjusted by the following equation:

$$\phi_t + F|\nabla\phi| = 0 \tag{3}$$

known as the Eikonal equation, where F represents the velocity related to the evolution of the surface properties such as the normal direction and its curvature.

When used for the task of image segmentation, F may depend on image information with an ideal value of zero at the edge of the object, that is, the largest value of the image intensity gradient.

Some advantages of using the level set method for curve evolution are the stability of the method and its capacity of handling changes of curve topology very easily.

2.3 Active Contours Without Edges

The active contours model without edges, also known as the Chan-Vese model, was introduced some years ago by Chan and Vese [3]. This model assumes that the gray level image can be split in regions of almost constant brightness. The mathematical formulation of the model is the minimization of a functional with the following components:

$$F(c_1, c_2, C) = \mu \cdot long(C) + v \cdot Area(In(C)) + \lambda_1 F_1(c_1, C) + \lambda_2 F_2(c_2, C). \tag{4}$$

with

$$F_1(c_1, C) = \int_{In(C)} |u_0(x,y) - c_1|^2 d\Omega \tag{5}$$

$$F_2(c_2, C) = \int_{Out(C)} |u_0(x,y) - c_2|^2 d\Omega, \tag{6}$$

where the given image (allowed to be noisy) is denoted by u_0 and defined on $\Omega \subset R^2$, the initial curve will be C and can take any form, size and location in the image, c_1, c_2 are constants that represent the average of the image inside and outside the curve respectively. $Area(In(C))$ and $long(C)$ are terms for keeping bounded the longitude of the curve and the area inside the curve. Finally, $\mu, v \geq 0$ and $\lambda_1, \lambda_2 > 0$ are parameters chosen in advance.

The problem is therefore to compute the minimum of F with respect to (c_1, c_2, C). The active contours model without edges can be rewritten using the level set formulation as follows:

$$\begin{aligned}\min_{c_1,c_2,\phi} F_{CV}(c_1, c_2, \phi) = \mu \int_\Omega \delta(\phi)|\nabla\phi| d\Omega + \nu \int_\Omega H(\phi) d\Omega \\ + \lambda_1 \int_\Omega |u_0 - c_1|^2 H(\phi) d\Omega + \lambda_2 \int_\Omega |u_0 - c_2|^2 (1 - H(\phi)) d\Omega\end{aligned} \tag{7}$$

where $\delta(\cdot)$ is the delta function and $H(\cdot)$ the Heaviside function.

2.4 Shape-Prior Based Model

A strong limitation of the Chan-Vese model is that it does not consider the shape of the region or object to be segmented. Therefore, artifacts with similar gray level to the one of the object of interest (OOI) are also assigned to the same region completely ruining the segmentation result. To remedy this, in [4] a shape-prior was introduced in the mathematical formulation. The main idea is to construct a shape model from a set of different views of the OOI and use this a priori information somehow in the model to force the contour to move in that direction. The model is as follows:

$$F_{PS}(c_1, c_2, \phi) = F_{CV}(c_1, c_2, \phi) + \alpha F_{shape}(\phi), \tag{8}$$

where

$$F_{shape}(\phi) = \int_{\Omega} (\phi(x) - \phi_0(x))^2 dx, \tag{9}$$

and ϕ_0 represents the shape prior constructed by averaging the level set functions of a set of images containing the OOI. Here $\alpha > 0$ is a weight.

3 Experiments

In this section, we present experimental results of using the active contour model with shape prior on ultrasound images of the heart of mice. In particular, the objective is to segment the left ventricle when it is in any of two states: diastole or systole. To this aim, shape priors were constructed for each state using a set of manually segmented images (the training set). To validate the results obtained from the semi-automatic segmentation, a python script was coded that generates a confusion matrix returning the true positive (TP), true negative (TN), false positive (FP) and false negative values (FN). With the values obtained from the confusion matrix we can calculate the accuracy, F1 score and the Mathews correlation coefficient (MCC).

3.1 Shape Prior Construction

The quality of construction of the shape prior is of paramount importance for the performance of the model. In this case, a set of manually segmented ultrasound images was used for this purpose. The manual segmentation focused in segmenting the left ventricle of the hearth in images from mice infected with the T. Cruzi parasite. It was decided to analyze only two states of the ventricle: the diastole state and the systole state.

Once the manual segmentation of the image set was finished, the binary images were processed with the level set algorithm to yield signed distance functions. This process even though slow, is relevant since it brings stability to the whole algorithm.

In Fig. 1(a), some samples of different ventricles, in diastole state, are shown. Although they are similar in shape, it is possible to observe some differences.

In Fig. 1(b) the resultant level set prior is presented. The shape prior looks as an out of focus image. This is due to the averaging process of all the level sets. The blurred regions may be interpreted as regions of uncertainty for the algorithm.

In Fig. 2(a), samples for the left ventricle in systole state are presented. The correspondent level set prior is shown in Fig. 2(b).

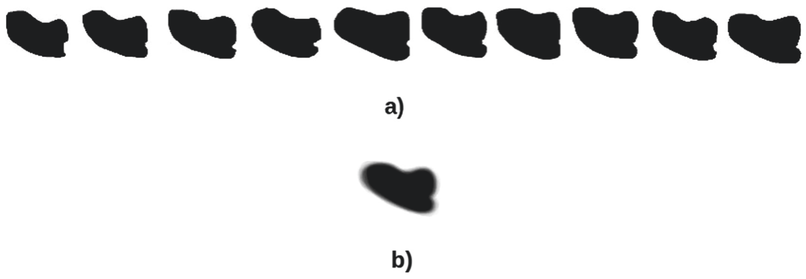


Fig. 1. (a) Diastole masks from the training set. (b) Average diastole mask.

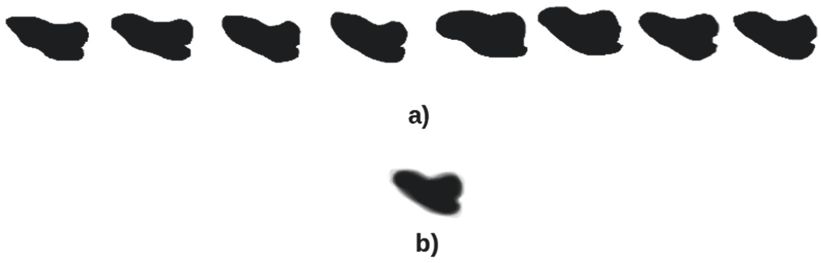


Fig. 2. (a) Systole masks from the training set. (b) Average systole mask.

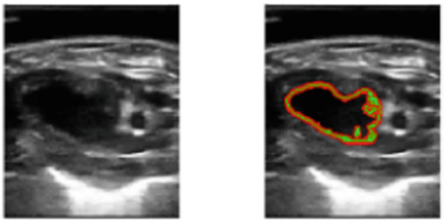
3.2 Tuning the Model and Results

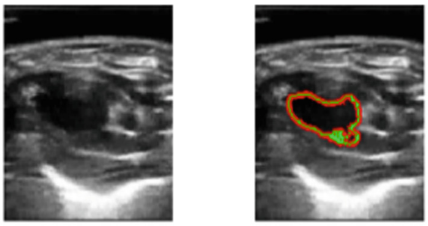
The selection of the parameters α, dt and μ is critical for the performance of the method. In this section, we show how to compute them and illustrate with some Figures the results for different combinations of them.

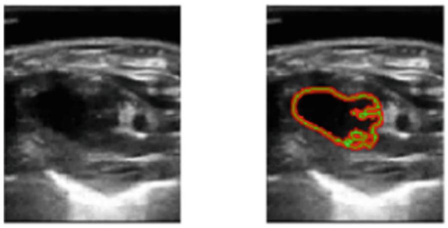
The first experimental tests were carried out over the ultrasonic images of the mice control group in the acute phase of the illness. Images of the hearth in diastole and systole states were used.

In Fig. 3, from (a) to (d), a good segmentation of the left ventricle, in both states, can be observed. The best parameters were obtained by trial and error by running the algorithm with different sets of values and evaluating accuracy, F1 score and Mathews coefficient correlation (MCC) at each iteration.

The parameters used here were $\alpha = 0.01$, a step size $dt = 0.1$ which allows a rapid convergence of the algorithm and a fixed $\mu = 0.2$. The algorithm converged on average after $1,485$ iterations. In the second experimental tests, the value of the weight parameter was increased to $\alpha = 15$, forcing contour to keep very close to the shape prior, the step size remained at $dt = 0.1$ and the value of $\mu = 0.2$ fixed. With this set of values, convergence was acquired after $1,458$ iterations on average.


(a) Diastole ventricle segmentation with accuracy of 97.3154%, F1 score of 98.5493% and MCC of 80.5765%


(b) Systole ventricle segmentation, resulting in an accuracy of 98.0501%, F1 score of 98.9745% and MCC of 79.1784%


(c) Diastole ventricle segmentation with accuracy of 97.3906%, F1 score of 98.5933% and MCC of 80.7767%

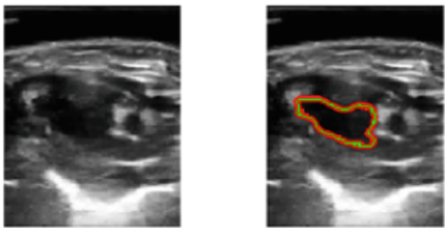
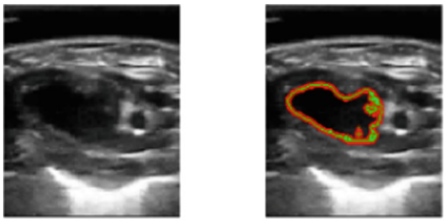
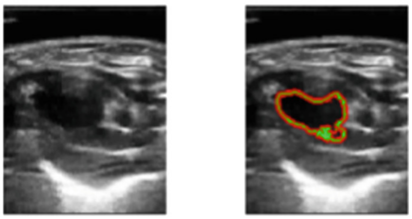
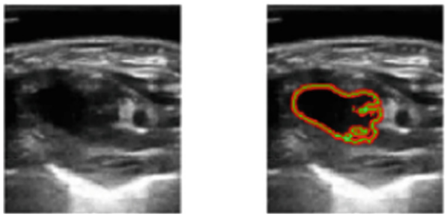

(d) Systole ventricle segmentation with accuracy of 98.8602%, F1 score of 99.4016% and MCC of 87.4044%

Fig. 3. Segmentation results with $\alpha = 0.01$, a step size $dt = 0.1$ and a fixed $\mu = 0.2$

The results of this experiment can be seen in Fig. 4 from (a) to (d). There it can be appreciated that a very small reduction on the quality of segmentation was obtained for the diastolic images while an improvement in accuracy of 0.025%, F1 score of 0.014% and MCC of 0.24% were obtained in Fig. 4(b) and also increases of 0.03% in accuracy, 0.01% in the F1 score and 0.4% in MCC for Fig. 4(d) can be observed. Finally, the third set of experimental tests were carried out with a value of $\alpha = 50$, a step size $dt = 0.5$ in order to accelerate the convergence of the algorithm and a $\mu = 0.2$. This time, it was observed that the algorithm converged on average in 364 iterations. The results are presented in Figs. 5 from (a) to (d) where, compared to the previous experiments, an overall increase in the accuracy values, F1 score and MCC, can be seen in both diastole and systole states.


(a) Diastole ventricle segmentation, resulting in an accuracy of 97.3154%, F1 score of 98.5480% and MCC of 80.7796%


(b) Systole ventricle segmentation with accuracy of 98.0752%, F1 score of 98.9878% and MCC of 79.4107%


(c) Diastole ventricle segmentation with accuracy of 97.3763%, F1 score of 98.5849% and MCC of 80.7796%

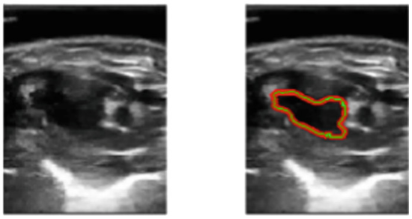
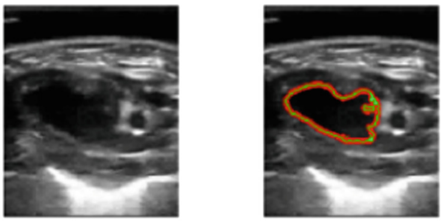
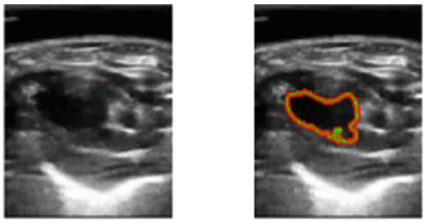
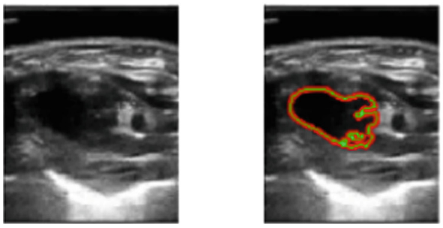

(d) Systole ventricle segmentation with accuracy of 98.8924%, F1 score of 99.4184% and MCC of 87.8153 %

Fig. 4. Segmentation results with $\alpha = 15$, a step size $dt = 0.1$ and $\mu = 0.2$ fixed.


(a) Diastole ventricle segmentation with accuracy of 97.7419%, F1 score of 98.7755% and MCC of 84.3559%


(b) Systole ventricle segmentation with accuracy of 98.1075%, F1 score of 99.0033% and MCC of 80.3443%


(c) Diastole ventricle segmentation with accuracy of 97.3763%, F1 score of 98.5849% and MCC of 80.7527%

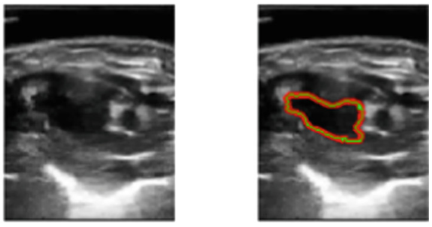

(d) Systole ventricle segmentation with accuracy of 98.8853%, F1 score of 99.4146% and MCC of 87.7450%

Fig. 5. Segmentation result with $\alpha = 50$, a step size $dt = 0.5$ and $\mu = 0.2$ fixed.

3.3 Morphological Post-processing

The output of the active contour model with shape prior very often presents some artifacts, either due to the inherent instability of the algorithm or due to noise present in the region of interest. Therefore, to smooth out the result and improve precision of the segmentation, we applied morphological filters such as bridge and fill to the output of the active contour model.

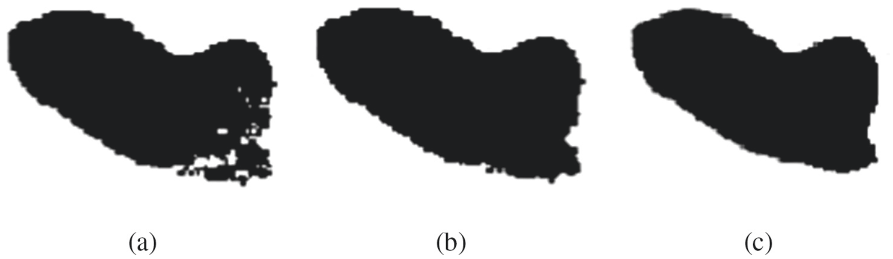


Fig. 6. Segmentation result for image problem in Fig. 3(a) for the diastole state. (a) Output from the active contour method (b) After morphological processing (c) Ground truth

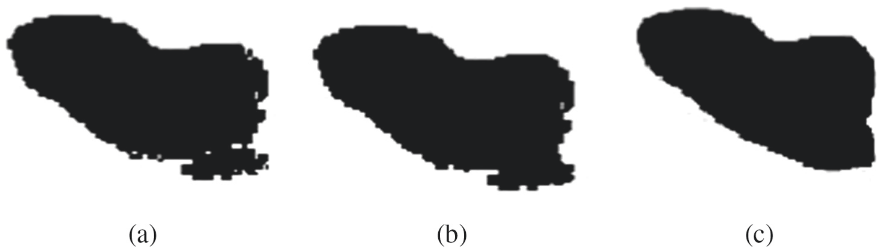


Fig. 7. Segmentation result for image problem in Fig. 3(c) for the diastole state. (a) Output from the active contour method (b) After morphological processing (c) Ground truth

In Figs. 6 and 7 we compare the results for the diastole state from the outcome of the active contours model, after morphological processing, and the ground truth (manual segmentation by the expert). In Figs. 8 and 9, we do the same for the systole state.

It can be appreciated by simple observation that morphological processing removes artifacts yielding a cleaner segmentation.

Finally, in Table 1, we present a summary of the different metrics we computed for each experiment.

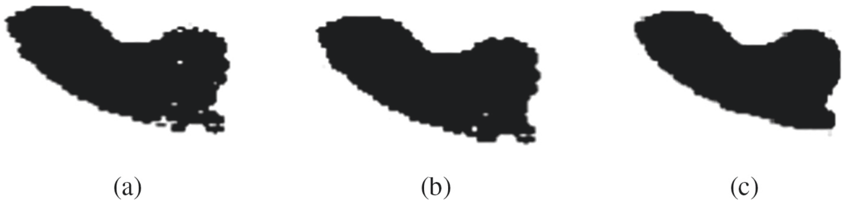


Fig. 8. Segmentation result for image problem in Fig. 4(b) for the systole state. (a) Output from the active contour method (b) After morphological processing (c) Ground truth

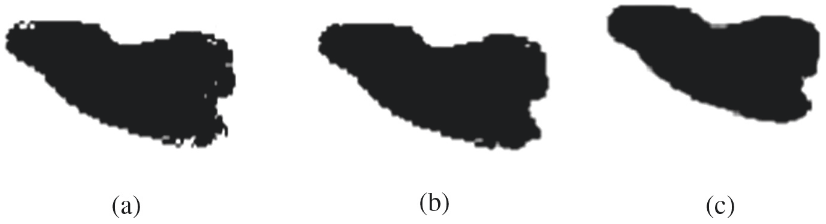


Fig. 9. Segmentation result for image problem in Fig. 4(d) for the systole state. (a) Output from the active contour method (b) After morphological processing (c) Ground truth

Table 1. Results of segmentation for the left ventricle in diastole state (first two rows) and systole state (last two rows).

Image problem	Accuracy (%)	F1 score (%)	MCC (%)	TP	TN	FP	FN
Figure 6	97.6416	98.7307	82.1177	25591	1651	313	345
Figure 6 with morphological operation	97.9104	98.8731	84.5971	25575	1742	222	361
Figure 7	97.8423	98.8447	82.5822	25752	1546	350	252
Figure 7 with morphological operation	97.9570	98.9051	83.6814	25744	1586	310	260
Figure 8	98.3011	99.1183	75.9173	26642	784	254	220
Figure 8 with morphological operation	98.3333	99.1345	76.6819	26632	803	235	230
Figure 9	98.4409	99.1832	82.1062	26412	1053	179	256
Figure 9 with morphological operation	98.4337	99.1791	82.2135	26399	1064	168	269

4 Conclusion

In this paper we presented a methodology to segment the left ventricle of the hearth of mice in ultrasound images by using the active contours model with shape prior. We shortly reviewed the theory of the active contours model and level set method and discussed the way to construct the shape prior for the two states of the ventricle: diastole and systole. The obtained results are encouraging since the quality and precision of the segmentation was pretty much in accordance with the manual segmentation from medical experts. Some metrics such as precision, F1 score and MCC were computed on the results and overall they achieved high values. We also presented how to further improve the quality of the segmentation by applying morphological post-processing to the binary images.

There is however still much to do in order to take this methodology to the next level. For instance, constructing the shape prior is very time consuming and tedious. There is also the need to align the shape prior to the region of interest in the image. This is an important step that now has to be carried out manually. The implementation of a simultaneous image registration and segmentation with shape prior algorithm would improve results by facilitating the alignment. This will be part of our future work.

Acknowledgements. Special thanks to Dr. Paulina Haro from the regional research center Dr. Hideyo Noguchi where part of this project was carried out. We also thank CONACYT for the funding provided in the form of a student scholarship.

References

1. Bridal, S.L., Correas, J.M., Saied, A., Laugier, P.: Milestones on the road to higher resolution, quantitative, and functional ultrasonic imaging. Proc. IEEE **91**(10), 1543–1561 (2003)
2. Uc-Cetina, V., Brito-Loeza, C., Ruiz-Piña, H.: Chagas Parasites Detection through Gaussian Discriminant Analysis. 1998 ACM Computing Classication System 8, 6–17, 2013 (1998)
3. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Trans. Image Process. **10**(2), 266–277 (2001)
4. Chan, T., Zhu, W.: Level set based shape prior segmentation. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), June 2005, vol. 2, pp. 1164–1170 (2005)
5. Noble, J.A., Boukerroui, D.: Ultrasound image segmentation: a survey. IEEE Trans. Med. Imaging **25**(8), 987–1010 (2006)
6. World Health Organization: Chagas disease (American trypanosomiasis) (2018)
7. World Health Organization: Diagnostic imaging (2018)
8. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. J. Comput. Phys. **79**(1), 12–49 (1988)
9. Zhou, Y.-Q., Foster, F.S., Nieman, B.J., Davidson, L., Chen, X.J., Henkelman, R.M.: Comprehensive transthoracic cardiac imaging in mice using ultrasound biomicroscopy with anatomical confirmation by magnetic resonance imaging. Physiol. Genomics **18**(2), 232–244 (2004)

Lung Cancer Patient's Survival Prediction Using GRNN-CP

Kefaya Qaddoum(✉)

Higher Colleges of Technology, Sharjah, UAE
kqaddoum@hct.ac.ae

Abstract. Published results for cancer patients have been previously estimated by applying various machine learning techniques to large. especially, for lung cancer, it is not well known to the time, which sorts of techniques would generate more imminent information, and which data attributes should be employed in order to prepare this information. In this study, a supervised learning technique is implemented to analyze lung cancer patients in terms of survival, the purpose of this study is to predict lung cancer and to compose an aiding model that will help form a more reliable prediction as a factor that is vital for advancing survival time evaluation. We utilize general regression neural networks (GRNN) for replacing the regular predictions with prediction periods to achieve a moderate percentage of confidence. The mechanism applied here employs a machine learning system called conformal prediction (CP), to assign consistent confidence measures to predictions, which are combined with GRNN. We apply the resulting algorithm to the problem of lung cancer diagnosis of supervised learning techniques is applied to the NCI database to classify lung cancer patients. Experimental results confirm that the prediction formed by this method is feasible and could be useful in clinical institutions.

Keywords: Neural network · Conformal prediction · Lung cancer classification · Biomedical big data

1 Introduction

CP is an original method, which can complement the predictions of conventional machine learning algorithms by measuring their confidence [4] in order to help to determine how accurate the prediction is, and to suggest good decision-making process consequently. References [4] and [5] proposed ICP to solve the computational ineffectiveness problem of CP.

This work uses a regression CP [1] built on neural networks (NNs). An adjusted CP was needed so as to apply CP to NNs, which is called generalized regression neural network conformal prediction (GRNN-CP). In the case of regression, CPs give a sufficient level of confidence compared to conventional techniques We used the National Cancer Institute (NCI) at the National Institutes of Health (NIH). As the largest publicly available cancer dataset [3], this database provides de-identified information on cancer statistics of the United States population, thus facilitating large-scale outcome analysis.

C. Brito-Loeza et al. (Eds.): ISICS 2020, CCIS 1187, pp. 143–150, 2020.
https://doi.org/10.1007/978-3-030-43364-2_13

We apply machine learning techniques to this dataset to analyze data specific to lung cancer, with the goal of evaluating the predictive power of these techniques. Lung cancer was chosen as it ranks as a leading cause of cancer-related death, with dismal 5-year survival rates. The goal of identifying survivability given a specific medical diagnosis is of great importance in improving care and providing information to patients and clinicians. Given a dataset of lung cancer patients with certain information such as age, tumor size, Radiation, and Surgery applied, the question is whether patient survival can be computationally predicted with any accuracy. Although survival time analysis may be considered clinically relevant to evaluate patient prognosis, doctors have struggled to estimate the diagnosis of lung cancer patients. In a recent study, physician consultants predicted a survival time median of 25.7 months, while physician registrars and residents predicted survival times of 21.4 and 21.5 months, respectively, for patients on average with 11.7 months actual survival [6]. The study found that only $\sim$60% of patients whose physicians estimated survival time $>$3 months survived this long. Another study found that physicians correctly predicted survival time to the month 10% of the time, to 3 months 59% of the time, and to 4 months 71% of the time, and tended to overestimate short term survival times but underestimate long term survival times [7, 10, 11]. Applying a correlational methodology via machine learning to predict survivability could help to improve such predictions.

In this study, patients diagnosed with lung cancer during the years 2006–2011 were selected in order to be able to predict their survival time. Some supervised learning methods were employed to classify patients based on survival time as a function of crucial attributes and, thus, help illustrate the predictive value of the several methods. The dataset in this study emphasizes on dimensions available at or near the time of diagnosis, which represents a more positive set of survival predictors

1.1 Producing Confidence Information

Machine learning may be used to produce accepted confidence of information, e.g., the Bayesian framework and 'probably approximately correct' (PAC theory) [5, 8]. This experiment will focus on the robustness of prediction intervals for a future independent observation to examine the dilemma of constructing prediction intervals in a regression phase. An improvement of a prediction interval over a point estimate is that it takes into account the variation of the future observation around the point estimate [6].

An expected failure might occur for the confidence levels to attribute the percentage of expected errors. The next section explains the framework then investigates, via a simulation study, the performance of these prediction intervals in terms of the prediction intervals robustness and their possible uses

1.2 The CP Framework

In this section, we describe the idea behind CP, and a more detailed description is provided by [1]. The interest here is in predicting the label of an example $xl + g$, based on a set of training examples $\{(x1, y1), \ldots, (xl, yl)\}$, where each $xi \in$ qd is the vector of attributes: for example, i and $yi \in R$ is the label of that example. The only assumption made is that all (xi, yi), $i = 1, 2, \ldots, n$ have been produced from the probability distribution.

The main aim of CP [6, 2] is to presume that each probable label $\hat{y}$ is presented in the form of the example *xl* + *g*, to check the possibility to generate the prediction rule:

$$\{(x1, y1), \ldots, (xl, yl), (xl + g, \hat{y})\} \tag{1}$$

This rule maps every input pattern *xi* to a predicted label *yi:*

$$D\{(x1, y1), \ldots, (xl, yl), (xl + g, \hat{y})\} \tag{2}$$

The nonconformity total of every set *(xi, yi)*: $y = 1, \ldots, l, l+g$ later estimated as the degree of contention between the prediction and the actual label *yi;* it may be noted that, in the case of the pair *(xl + g, y)*, the actual label is replaced by the assumed label y. The function used for measuring this degree of contention is referred to as the nonconformity measure of the CP. A change in the assumed label ŷ affects all predictions. Following this, the nonconformity score x*l* + *g* is compared to the nonconformity results of remaining examples to ascertain how rare the pair *(xl + g, y)* is, regarding the nonconformity measure used by the following function:

$$\hat{y}i = D\{x1, y1), \ldots, (xl, yl), (xl + g, \hat{y})\}) \tag{3}$$

The main weakness of the prime CP technique is that, given its inspirational quality, all its computations require repeating each new test example for every assumed label. This makes it computationally incompetent. CP is tightly efficient [6], and maybe merged with any traditional regression technique.

CP splits the training set (of size *l*) into two smaller sets; the convenient training set with $m < l$ examples, and the calibration set with $q{:} = l - m$ examples. Then, it uses the convenient training set for training, and the calibration set for calculating the probability distribution of each possible label *y* for *(x1, y1), . . ., (xm, ym)* to generate the prediction rule, where the nonconformity of each example in the calibration set is *i* = *1, …, q*, and the confidence level to be calculated as *1* − o which provide the minimum and maximum $\hat{y}$

CP algorithm requires a critical parameter that is the number *q*, of training examples to be allocated to the calibration set, while the nonconformity scores used by the CP to create its prediction intervals. This number is critical and should only relate to a small portion of the training set, where removing these examples causes a significant decrease in the predictive capability of the NN, and accordingly, to broader prediction intervals.

1.3 GRNN-CP Framework

The GRNN created by [5] is an estimation method for function regression that has been applied to engineering and science applications. GRNN is useful since it could employ a few training samples to converge to the underlying function of the data available. GRNN also is a useful tool to perform predictions and comparisons of system performance in practice. The standard GRNN in Fig. 1 can be used with a rapid training procedure due to the single training parameter σ. Finally, it does not require an exact topology definition such as the MLP, or basis function centers and weights, such as the RBF.

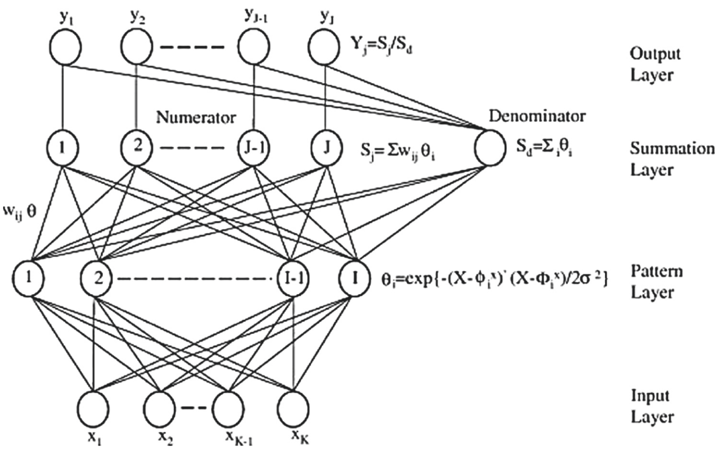


Fig. 1. General GRNN architecture (adapted from [9])

Employing CP with GRNN has the advantage of enabling much better control of the smoothness of the approximation so that the regression surface adapts to the local properties of the occurring in that area concedes a predicted output value data. In order to use CP in conjunction with some traditional algorithms, a nonconformity measure first needs to be defined.

As previously discussed, a nonconformity measure is a function measuring the contention between the actual label *yi* and prediction *yi*' produced by the prediction rule described by [6] of the underlying algorithm for the example *xi*. Regression meanwhile, can be readily defined as the absolute difference Between the two. This section describes the GRNN in CP shown in (4)

$$D(x, x) = \sum^{P} (x - x/\sigma)^2 \tag{4}$$

where yi is the ith case actual output value, D(x, xi) is calculated from (5), and n is the total number of cases in the dataset j = 1

$$D(x, x) = \sum^{P} (x - x/\sigma)^2 \tag{5}$$

(GRNN CP) algorithm, and defines a normalized nonconformity measure, which has the effect of producing tighter prediction intervals by taking into account the expected accuracy of GRNN.

The GRNN predicts continuous outputs. GRNN nodes require two main functions to calculate the difference between all sets of input pattern vectors and estimate the probability density function of the input variables. Euclidean distance is used to calculate the difference between input vectors between data values in attribute space. Weighting the calculated distance of any point by the probability of other points, where x is the input vector, xi is the i^{th} case vector, xj is the j^{th} data value in the input vector, xij is the j^{th} data value in the ith case vector, and σj is the smoothing factor (Parzen's window) for the j^{th} variable [6]. The error measurement of the mean square error (MSE) used in this work.

The MSE measures the average of the square amount by which the estimator differs from the quantity to be estimated. While finding the error, the calculation mentioned earlier will frequently be running with different smoothing factors (sigmas) [8]. Training stops when either a threshold minimum square error value reached, or the test set square error concluded. Since the aim is to produce a level of confidence information, we employ GRNN here to complement predictions with probabilistic texture. The purpose of the global parameter σ is to regulate the smoothness of the regression surface. However, as discussed previously, because the data density can vary in different regions, different values of σ may be needed for different patterns xi. Allocating an individual σ i for each i^{th} pattern in (5) and combining with (6) produces the standard GRNN as follows:

The smoothness parameter was arbitrarily chosen to $\sigma = 0.1$. As explained earlier in Sect. 3, CP splits the training set $\{(x1, y1), \ldots, (xl, yl)\}$ into two subsets: the convenient training set: $\{(x1, y1), \ldots, (xm, ym)\}$, and the calibration set:

$$\{(xm + 1, ym + 1), \ldots, (xm + q, ym + q)\}.$$

$$\mathrm{\alpha i} = \left|\mathrm{yi} - \mathrm{\hat{y}i}\right| \tag{6}$$

The GRNN-CP continues as follows:

Sort the nonconformity scores in descending order achieving the following order

$$\mathrm{\alpha(m + 1), \ldots, \alpha(m + q)} \tag{7}$$

For each new test example $xl + g$: supply the input pattern $xl + g$ to the trained GRNN to get the prediction $\hat{y}l + g$ and output the prediction interval

$$\left(\mathrm{\hat{y}l + g - \alpha(m + s), \hat{y}l + g + \alpha(m + s)}\right) \tag{8}$$

where $\mathrm{s = o(q + 1)}$.

2 Experimental Evaluation

The suggested approach has been examined on the NCI [9] dataset as Table 1 shows, dataset contains 683 instances with nine integers valued attributes for each instance. Prior to conducting the experiments in this section, datasets were normalized to the range between $[-1, 1]$. A random split has been conducted into k folds, and the trials were repeated k epochs, each using one k fold was tested, and the other $\mathrm{k} - 1$ folds to be the training set. Trial and error decided the number of hidden neurons, through a fold cross-validation process, with the GRNN predictor on stochastic sequences, which were different from those that evaluated the GRNN- CP. The GRNN was applied to both calibration and test samples.

The performance of the point predictions of the method used in this section, comparing its predictions to the estimated values, can estimate a model trained on the training set. These values are determined by periodically adjusted various model parameters. The fulfillment of the model was evaluated in terms of its root mean squared error (RMSE), see Fig. 2.

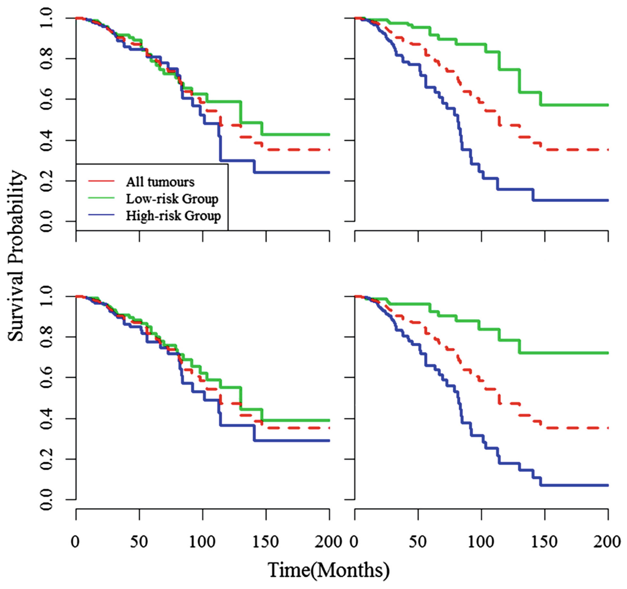


Fig. 2. Survival rate prediction with GRNN

Table 1. Data attributes. AJCC [9]:

Number	Attribute	Description	Type
1	Age:	Age at time of diagnosis	Discrete
2	Grade	Appearance of cancer cells and how fast they may grow	Numeric
3	Radiation	Sequence with Surgery Order of surgery and radiation therapy administered for patients who received both	Numeric
4	The number of Primaries	Number of malignant tumors other than the lung	Discrete
5	T	AJCC component describing tumor size	Numeric
6	N	AJCC component describing lymph node involvement	Numeric
7	M	AJCC component describing tumor dissemination to other organs	Numeric
8	Radiation	Indication of whether patient has received radiation	Numeric
9	Stage:	Stage of tumor - based on T, N, and M	Numeric
10	Primary Site Location of the tumor	within the lungs	Numeric

Measures, since we would like to have as many precise predictions as possible, given high confidence levels. The validation data across the Age, Stage, Grade, and Tumor Size groupings. Each frame has multiple lines showing the decrease in survival probability for each value, e.g., Stage, etc. The bulk of the curves is below the 50% survivability rate.

Since 50% of the validation patients survive less than 15 months, the standard deviation of residuals is higher than the survival time of half the population; any guess $\sim$15 months would have a similar result. Most of this deviation originates from the most extended surviving patients, which turned out to be the most difficult to predict. In contrast, the RMSE for patients in the validation set with survival time $\leq$35 months, compared to the ensemble prediction, is 11 months.

The comparison of the results to those in previous work or to clinical estimates is non-trivial. Much of the previous work differs from the approach here primarily through logistic regression into categorical survival times. 81% of those the model predicted results to live longer than 3 months.

The RMSE value is not the only factor to consider, however, favorable RMSE value, does not certainly monitor that RMSE correlates to relevance. The resulted values show that the prediction intervals produced by the method developed in this chapter are quite tight. The median widths obtained using nonconformity measures are 76.4% and 49.2% of the label range of the two datasets correspondingly, while the best widths achieved using the nonconformity degree are 72.5%, and 42.1% of the label range.

3 Conclusion

A new prediction system has been constructed in this paper. The proposed algorithm is based on using CP to find the most reliable prediction regressions using GRNN, to achieve low errors and more reliable predictions as the results show. The tests performed on the proposed training algorithm show that the right level of accuracy may be achieved when compared to other models.

A moderately considerable correlation was recognized between the measured and predicted values using the hybrid GRNN-CP method. The proposed algorithm produces prediction intervals to achieve a fitting confidence level. In terms of point predictions, the performed correlation coefficient between the predicted and the actual values was convenient; For example, 89% confidence level covers 21.5% of the data, while for the 91% confidence level, it covers 16.9%. It is worth mentioning that the prediction intervals produced by the proposed method are not only well-calibrated, and therefore highly stable, but they are also tight enough to be useful in patients' trials. Besides, GRNN-CP made progress in terms of prediction interval tightness over the average regression measure, but it still could be developed by reaching more tightness when it comes to prediction regression. Also, other regression methods could be implemented and examined with CP, taking into attention that adding extended datasets with more records could enhance prediction confidence. The models excel when dealing with low to moderate survival time instances, which is the large majority of the data, although there were challenges with both the data and the models, such as the non-linearity of outcomes. As the models struggle to predict patient survival time exceeding 35+ months, logistic regression may be preferred. The cause could be having too many less weighty criteria or too few rules, or that the inexperienced volume of data is needed. Moreover, the more complex models may be insignificantly more precise than the linear regression but maybe more difficult to decipher. Whether or not the increment in performance is worth the extended complexity should be investigated in future.

Future work could also reassess the data optimization and inputs.

References

1. Papadopoulos, H.: Regression conformal prediction with nearest neighbours. J. Artif. Intell. Res. **40**, 815–840 (2011)
2. Specht, D.F.: A general regression neural network. IEEE Trans. Neural Netw. **2**(6), 568–576 (1991)
3. Umesh, D.R., Ramachandra, B.: Association rule mining based predicting lung cancer recurrence on SEER lung cancer data. In: 2015 International Conference on Emerging Research in Electronics Computer Science and Technology (ICERECT), pp. 376–380 (2015)
4. Holst, H., Ohlsson, M., Peterson, C., Edenbrandt, L.: Intelligent computer reporting 'lack of experience': a confidence measure for decision support systems. Clin. Physiol. **18**, 139–147 (1998)
5. Papadopoulos, H., Gammerman, A., Vovk, V.: Confidence predictions for the diagnosis of acute abdominal pain. In: Proceedings of the 5th IFIP Conference on Artificial Intelligence Applications & Innovations, pp. 175–184 (2009)
6. Gammerman, A., Nouretdinov, I., Burford, B., Chervonenkis, A., Vovk, V, Luo, Z.: Clinical mass spectrometry proteomic diagnosis by conformal predictors. Stat. Appl. Genet. Mol. Biol. **7**(2) (2008)
7. Mangasarian, O.L., Wolberg, W.H.: Cancer diagnosis via linear programming. SIAM News **23**(5), 1–18 (1990)
8. Qaddoum, K., Hines, E., Iliescu, D.: Yield prediction technique using hybrid adaptive neural genetic network. Int. J. Comp. Intel. Appl. **11**, 1250021 (2012). http://dx.doi.org/10.1142/S1469026812500216. (15 pages)
9. NCI_Lung_Cancer_Overview, Lung Cancer, National Cancer Institute (2015). http://www.cancer.gov/cancertopics/types/lung/
10. Identifying hotspots in lung cancer data using association rule mining. In: Agrawal, A., Choudhary, A. (eds.), 11th International Conference on Data Mining Work shops (ICDMW). IEEE (2011)
11. Yu, X.Q., Luo, Q., Hughes, S., et al.: Statistical projection methods for lung cancer incidence and mortality: a systematic review. BMJ Open **9**, e028497 (2019). https://doi.org/10.1136/bmjopen-2018-028497

Author Index