Handwritten Signature Verification Using Transfer Learning and Data Augmentation

Yash Gupta, Ankit, Sanchit Kulkarni, and Pooja Jain

1 Introduction

Signature is one of the most commonly accepted methods for personal verification and identification. Signature verification is important for banking, legal documents and still an important area of research in the field of machine learning and deep learning. Typically, signatures are of two types: (1) handwritten and (2) digital. Capturing handwritten signature needs a paper with pen or electronic pad with stylus. Apart from the ink impression on the paper, signature verification also requires to consider writing speed, pressure, etc.

In this paper, we focus on feature extraction and classification on the image dataset of handwritten signature stored in PNG format. We make several contributions for each feature extraction and classification. First, to obtain feature extraction for each algorithm and CNN architectures independently. Second, we present algorithms that are more suitable for classification famous as supervised learning algorithm. Data augmentation is another aspect of our paper where even the small dataset can be used to increase the dataset for performing the feature extraction task (Figs. 1 and 2).

The remainder of this paper is organized as follows. In Sect. 2, we have discussed the problem and literature survey. Section 3 introduces algorithms and architectures to efficiently extract features and discusses algorithms and methods of classification using those extracted features. In Sect. 4, perform a set of experiments on various architectures using concepts of transfer learning and data augmentation. Finally, we have a conclusion for our work and suggestions for future work in Sect. 5.

Y. Gupta · Ankit · S. Kulkarni · P. Jain (✉)
Computer Science and Engineering, Indian Institute of Information Technology, Nagpur, India
e-mail: pooja.jain@cse.iiitn.ac.in

B. Agarwal et al. (eds.), *Proceedings of International Conference on Intelligent Cyber-Physical Systems*, Algorithms for Intelligent Systems,
https://doi.org/10.1007/978-981-16-7136-4_19

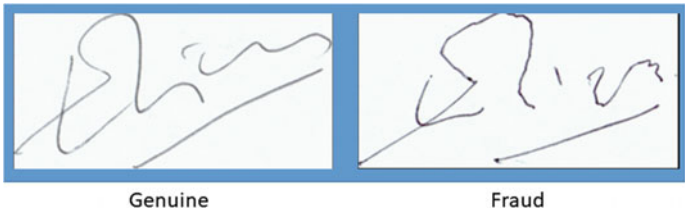


Fig. 1 Signature of the same subject left: real right: morphed

Fig. 2 Augmentation on same image rotation and zoom

2 Literature Survey

The objective is to develop the handwritten verification system using the latest advancement in deep learning. Input parameter to this system is paired of two signatures in portable network graphics images (PNG) format and outputs the Boolean value (1 or 0). This paper focuses on the experiment of convolution neural network architectures and different classification techniques. The deep learning-based method has emerged as successful tool for computer vision and pattern recognition-related applications [17]. It is a lot easier to verify the digital signature as compared with handwritten signature verification, this counts in the most challenging areas of pattern recognition. Although signature verification is a well-researched problem and there are many contributors of the same.

SVC2004 [1] "The first international signature certification competition". The competition has two competitions, competing first with 13 teams with ERR 2.84% and second with 8 teams with ERR 2.89%.

Many ways to get the right limit from the reference are being investigated. A positive result yields a false negative rejection rate of 2.8% and a false acceptance rate over 1.5%. A database test containing a total of signatures over 1200 of people greater than 100 shows that author-based thresholds provide better results than using the same limit [3]. The "Siamese" process of the neural network is used.

The authenticity of the test signature is established by aligning it with the reference user's reference signature, using a dynamic time. The authors [4] compared the test signature with the corresponding mean values found in the reference set, forming a three dimensional vector. This feature vector is then divided into one of two categories (real or fraud). The key component analysis received an error rate of 1.4% of the 619 test signatures and 94 people.

From Ref. [5], an online signature verification methodology has been introduced. The system uses a timeline set with Hidden Markov Models (HMMs). Development tests and tests are reported in the subcorpus of the MCYT bimodal biometric database containing more than 6500 signatures from a total of 145 studies [6]. The developers were familiar with the verification process and did their best to defraud the system. The acceptance rate for random forgeries, i.e., the accidental similarity of two different signatures, was 0.16%.

Classifiers based on neural net feeds are used. The factors used to distinguish are guessing times and symbols based on the upper and lower envelope. The output of the three separators is integrated using a connecting system. The integration of these separators based on signature verification is a distinctive feature of this work. Test results show that the combination of classifiers increases the reliability of visual results.

In Ref. [7], Datasets selected by CEDAR, MCYT and GPDS were performed. The performance of the algorithm proposed is based on three precision steps such as FAR, FRR and AER [9]. Compared with the standard system, the findings were found to be 20% error. The database SVC2004 was selected to verify the signature [10]. We tested our approach to GPSSynthetic, MCYT, SigComp11 and CEDAR databases that demonstrate the generality of our suggestion. The review [11] includes the implementation of state-of-the-art programs in selected subjects in five public databases.

The authors [8] used signal processing for the signature verification task. Vector representation of words is used for the analysis of sentiments [16]. The authors [12–15] used the Google Net, Inception-v1, Inception-v3, DAG-CNN and other architectures model for signature verification. We have in-depth studied in this paper about the VGG16, ResNet-50, Inception-V3 and Xception architectures.

3 The Proposed Methodology

A. Dataset

Data used are ICDAR 2011 Signature Dataset, which consists signature of 69 subjects and multiple genuine and forged signatures.

B. Proposed system

This paper is divided into two major steps (1) Feature extraction and (2) Classification. Following toward the tasks, CNN models are involved for feature extraction and supervised models for classification.

C. Feature extraction

Convolution neural network (CNN) is a popular neural network architecture for working on image dataset. It consists of certain number of layers such that output of previous layer is fed to next layer as input. These images are feed as 3D array than 1D or flattened array because CNN architectures are designed to treat images as human visual cortex. The architecture of CNN determines the function of each layer and connections in layers. There are four architectures of CNN used in the paper for experiment, VGG16, Inception-v3, ResNet-50 and Xception. Choosing a suitable architecture for the dataset is crucial to complete the first step, i.e. feature extraction of our handwritten signature verification system.

Convolution Layers: The convolution layer is the building block of the convolutional network that does a lot of complex computer-based lifting. CONV layer parameters contain a set of readable filters. Each filter is small in area (in terms of length and width) but expands to the full depth of input volume. During the progression, we slide (accurately, convince) each filter into the width and height of the input volume and calculate the dot products between the filter input. As we load the filter over the dimensions of the input, we will produce a map that provides the feedback for that filter to all areas below.

Max-pooling Layers: High integration, or greater cohesion, is a merging function that determines the maximum, or largest value in each section of the map for each feature. The results are sample or combined feature maps highlighting the feature that is most present in the piece, not the central presence of the feature in the case of a moderate combination. This has been found to be more effective in performance than standard integration of computer viewing functions such as image splitting. We can make concrete of the composite work by re-inserting it into the feature map of the active metal detector and manually calculating the first line of the composite map.

All the architectures used in the paper are modified by removing the fully connected layers with the output layer to fine-tune the already trained model to extract features.

Architecture 1: VGG16

VGGNet-16 has 16 layers of resolution and is very attractive due to its similar Architecture, similar to AlexNet, but has many filters. It is currently the most widely used way to remove elements from images. VGG16 weight loss is publicly available and used in many other programs and challenges as a first feature release. However, VGG16 has 138 million parameters, which is a challenge to train. When the model is specified in the database and the parameters are changed and updated for increased accuracy, we can use the parameter values.

Architecture 2: Inception-v3

In 2014, Google researchers introduced the first network that stood first in the competition, which has ImageNet dataset for discovery challenges.

The model is made up of a basic unit called the "Inception Cell" in which we perform a series of interaction. Inception-v3 has 24 M parameters.

Architecture 3: ResNet50

ResNet50 is a variety of ResNet model with 48 layers. It is the most popular and used ResNet model, and we have the design of ResNet50 in depth. ResNets was originally included in the image recognition function but as stated in the paper that the framework can be used for non-computer activities and for better accuracy. ResNet50 has 23 M parameters.

Architecture 4: Xception

Xception is known as Extreme Inception based entirely on divisively divisive structures. The construction of Xception has 36 disclosure layers that form the basis of network outsourcing. The 36 layers of convolutional are organized into 14 modules, all with residual connections around it, with the exception of the first and last modules. The Xception architecture is a series of deep dividing layers with remaining connections. Xception is an adaptation from Inception model and has 23 M parameters.

Optimizers used are (i) Stochastic gradient descent (SGD), (ii) Root Mean Square Propagation (RMSprop) (iii) Adaptive Gradient Algorithm (Adagrad), (iv) Active Design and Analysis Modeling (Adam).

D. Classification

After the feature extraction, they are stored in the comma separated value (CSV). We have obtained a different number of features as described in Table 1. Explaining back our dataset, we have prepared the pairwise data for each subject. The first column is genuine signature, and the second column is either genuine signature or fraud signature associated with the same subject. Labels used are 0 if both signatures are genuine and 1 otherwise.

File format used for genuine signature is PNG and naming schema used is XXX/YY_XXX for Genuine signature and XXX_forg/YY_ZZZXXX for forged signature. XXX denotes the person ID, YY denotes the signature number, ZZZ is the person ID signed the signature.

In the classification, we have used (i) Euclidian Distance, (ii) Cosine Similarity, (iii) Linear SVM, (iv) RBF SVM, (v) Sigmoid SVM, (vi) Poly SVM, (vii) Logistic

Table 1 Basic info about architecture

Architecture	VGG16	Inception-v3	ResNet-50	Xception
Parameters	138 M	24 M	23 M	23 M
Features Extracted	512	2048	2048	2048

Table 2 The random sample of five points

Column 1	Column 2	Target
065/06_065	065_forg/01_0118065	1
042/09_042	042_forg/03_0118042	1
029/05_029	029/02_029	0
001/001_12	001/001_15	0
065/01_065	065/05_065	0

Regression and (viii) Random Forest. We first created pairwise similarity between column 1 and column 2 using Euclidian distance and Cosine similarity. Pairs with the similarity greater than trainable hyper-parameter are not forged.

$$\textbf{Euclidian Distance} : d(p,q) = \sqrt{\sum_{i-1}^{n}(q_i - p_i)}$$

$$\textbf{Cosine Similarity} : sim(p,q) = \frac{p \cdot q}{\|p\| * \|q\|} = \frac{\sum_{i=1}^{n}(p_i * q_i)}{\sqrt{\sum_{i-1}^{n} pi^2} * \sqrt{\sum_{i-1}^{n} qi^2}}$$

∴ p, q are Euclidean points;p_i, q_i are feature vectors; n is dimension of vector.

Support Vector Machine, Logistic Regression, Random Forest: K(Xi, Xj) = (Xi,Xj + 1)d.

Features of image 1 and image 2 are concatenated to make total features of 2 * features of 1 image. The reason to choose SVM is that our previous experience and its results on high dimensional data (Table 2).

4 Results

The results obtained with feature extraction are presented in Tables 3, 4, 5, 6 and Graphs 1 and 2. The results obtained with classification are presented in Tables 7, 8, 9, 10. Bold value represents the best results.

Table 3 Training accuracy (3-fold)

	Optimizers			
	SGD	RMSprop	Adagrad	Adam
VGG16	0.8648	**0.9645**	0.8821	**0.9584**
Inception-v3	0.8042	**0.9827**	**0.9567**	**0.9922**
ResNet50	**0.9515**	**0.9991**	**0.9991**	**0.9974**
Xception	0.7730	**0.9835**	0.8215	**0.9939**

Table 4 Training loss (3-fold)

	Optimizers			
	SGD	RMSprop	Adagrad	Adam
VGG16	**0.4497**	**0.0918**	0.3716	**0.1069**
Inception-v3	0.4485	**0.0448**	0.2218	**0.0176**
ResNet50	0.1561	**0.0050**	**0.0324**	**0.0084**
Xception	0.5424	**0.0642**	0.4889	**0.0221**

Table 5 Validation accuracy (3-fold)

	Optimizers			
	SGD	RMSprop	Adagrad	Adam
VGG16	**0.7091**	**0.9717**	0.5111	**0.9556**
Inception-v3	0.5818	0.4202	**0.6020**	**0.6323**
ResNet50	0.4182	0.5879	0.5818	0.4182
Xception	0.5697	0.5818	0.5657	0.5899

Table 6 Validation loss (three-fold)

	Optimizers			
	SGD	RMSprop	Adagrad	Adam
VGG16	**0.5971**	**0.0793**	**0.9206**	**0.1127**
Inception-v3	**0.7371**	8.5688	**0.7872**	2.3959
ResNet50	1.2646	**0.6738**	1.4782	**0.7494**
Xception	**0.7339**	7.0186	**0.7754**	3.2455

Table 7 Model i—VGG16-Adam

	Accuracy	Precision	Recall	Time
Logistic Regression	0.9852	0.9789	0.9880	7.2 s
Random Forest	**0.9900**	**0.9851**	**0.9926**	**12.5 s**
Linear SVM	**0.9902**	**0.9909**	**0.9956**	**66 s**
RBF SVM	0.9353	0.9361	0.9146	75 s
Sigmoid SVM	0.5017	0.4322	0.4349	82 s
Poly SVM	0.8624	0.8977	0.7962	67 s

The first observation from the above tables VGG16 architecture outperformed all other architectures and features from the models that can be used for classification are with at least 95% training accuracy and 60% validation accuracy. Four models that we choose to test our classification algorithms are (i) VGG16—Adam, (ii) VGG16—RMSprop, (iii) Inception-v3—Adam and (iv) Inception-v3—Adagrad.

Table 8 Model ii—VGG16-RMSprop

	Accuracy	Precision	Recall	Time
Logistic Regression	0.9887	0.9836	0.9911	7.22 s
Random Forest	**0.9911**	**0.9868**	**0.9934**	**12.1 s**
Linear SVM	**0.9915**	**0.9860**	**0.9932**	**38.1 s**
RBF SVM	0.9625	0.9558	0.9534	56.8 s
Sigmoid SVM	0.5608	0.4982	0.4982	80 s
poly SVM	0.8699	0.9040	0.8060	62 s

Table 9 Model iii—Inception-v3—Adam

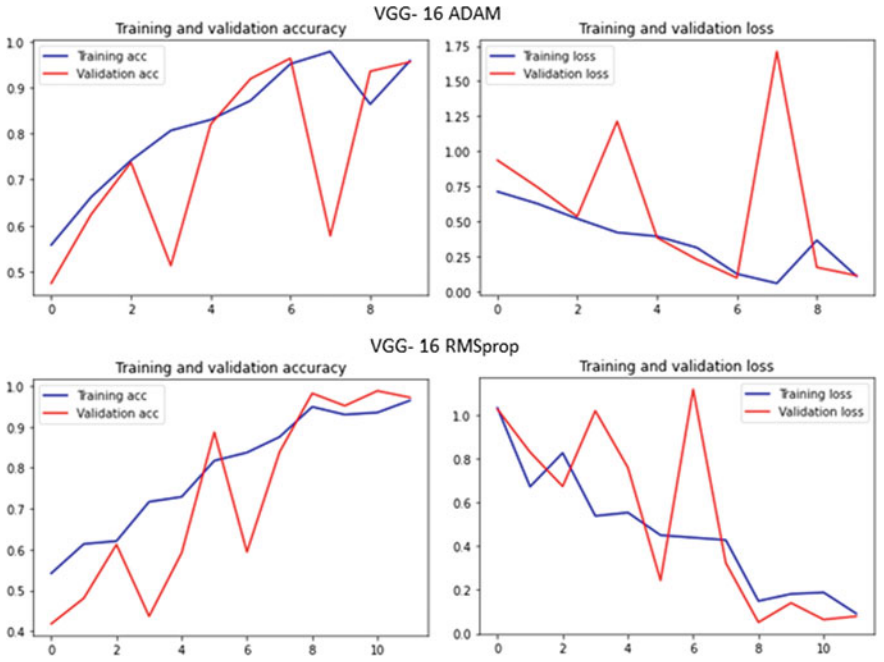
	Accuracy	Precision	Recall	Time
Logistic Regression	0.8897	0.8698	0.893	8.48 s
Random Forest	**0.9914**	**0.9872**	**0.993**	**11.8 s**
Linear SVM	0.9347	0.9100	0.946	406 s
RBF SVM	0.8955	0.8790	0.894	93 s
Sigmoid SVM	0.6680	0.6267	0.625	64 s
poly SVM	0.9066	0.8907	0.907	74 s

Table 10 Model iv—Inception-v3—Adagrad

	Accuracy	Precision	Recall	Time
Logistic Regression	**0.9920**	**0.9883**	**0.9939**	**7.39 s**
Random Forest	0.9891	0.9839	0.9919	14.8 s
Linear SVM	0.9894	0.9857	0.9923	19.2 s
RBF SVM	**0.9915**	**0.9882**	**0.9942**	**32.3 s**
Sigmoid SVM	0.9178	0.9101	0.9139	31.4 s
poly SVM	**0.9928**	**0.9892**	**0.9945**	**19.9 s**

We now, for the classification, refer to the selected architecture for feature extraction with the assigned roman number above. i, ii, iii and iv for VGG16—Adam, VGG16—RMSprop, Inception-v3—Adam and Inception-v3—Adagrad, respectively.

For the Classification Tasks, models selected for feature extraction are used with supervised learning algorithms and supporting performance metrics such as Accuracy for each model, Confusion Matrix whenever necessary. All the models are trained on CPU i5-7200U with 8 GB of RAM. We ran the tests for three cross-fold validation.

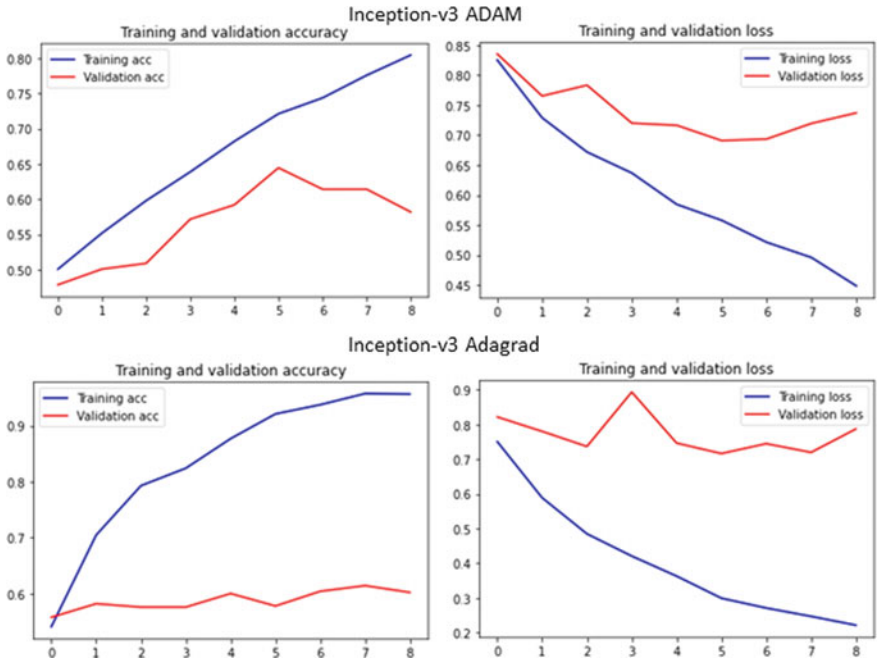


Graph 1 Feature extraction results for a selected model of VGG16

From our observation, Euclidean distance and Cosine similarity didn't perform well with our model while SVM outperformed all the models. Euclidean distance and Cosine similarity-based classification methods tend to overfit with our features. Average training time for Inception-v3-Adagrad architecture is significantly lower than other architectures. Bold marked model is with accuracy greater than 99% in Tables 7, 8, 9, 10. Best performing model is poly SVM with an accuracy of 99.28%. Other metrics to evaluate the model are also in Fig. 3. Figures 4, 5, 6, 7 are sample examples of our final handwritten signature verification system.

5 Conclusion

In this paper, we have presented methods for feature extraction and classification on signature dataset. This paper does not focus on manual crafted features, inspire feature extraction is done using CNN architectures. Experiment conducted on the ICDAR 2011 Signature Dataset showed features extracted from VGG16 outperformed all other architectures with small margins. Other architectures may perform better than other datasets because experiments are random for feature extraction. In addition, classification best results are obtained with poly SVM on feature extracted from Inception v3trained on Adagrad optimizer.



Graph 2 Feature extraction results for a selected model for Inception-v3

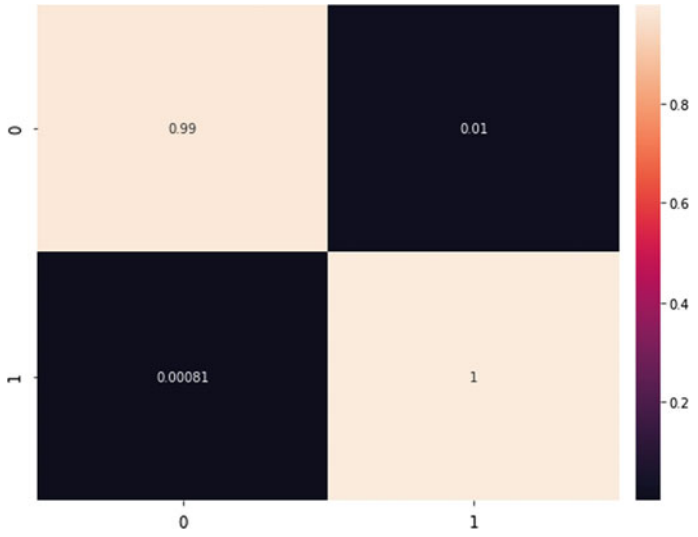


Fig. 3 Confusion matrix for best performing model

Fig. 4 Sample Example 1 after classification

Fig. 5 Sample Example 2 after classification

Fig. 6 Sample Example 3 after classification

Fig. 7 Sample Example 4 after classification

Though studies till now have proved its best results on the recognition of handwritten digits (MNIST), its performance is not significant in the verification of signatures. As future work, we will focus on the development of a more purpose-specific neural network model. Furthermore, other different classification techniques specific to signature data can be explored.

Acknowledgements We thank the Indian Institute of Information Technology for constant support and providing opportunity.

References

1. Yeung, D.Y., Chang, H., Xiong, Y., George, S., Kashi, R., Matsumoto, T., Rigoll, G.: SVC2004: First international signature verification competition. In International conference on biometric authentication, pp. 16–22. Springer, Berlin (2004)
2. Jain, A.K., Griess, F.D., Connell, S.D.: On-line signature verification. Pattern Recogn. **35**(12), 2963–2972 (2002)
3. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a" siamese" time delay neural network. Adv. Neural. Inf. Process. Syst. **6**, 737–744 (1993)
4. Yildiz, M., Yanikoğlu, B., Kholmatov, A., Kanak, A., Uludağ, U., Erdoğan, H.: Biometric layering with fingerprints: template security and privacy through multi-biometric template fusion. Comput. J. **60**(4), 573–587 (2017)
5. Fierrez, J., Galbally, J., Ortega-Garcia, J., Freire, M.R., Alonso-Fernandez, F., Ramos, D., Gracia-Roche, J.J.: BiosecurID: a multimodal biometric database. Pattern Anal. Appl. **13**(2), 235–246 (2010)
6. Praino, A.P., Treinish, L.A.: IBM Thomas J Watson Research Center, Yorktown Heights, New York
7. Sharif, M., Khan, M.A., Faisal, M., Yasmin, M., Fernandes, S.L.: A framework for offline signature verification system: Best features selection approach. Pattern Recogn. Lett. (2018)
8. Ghosh, R.: A recurrent neural network based deep learning model for offline signature verification and recognition system. Expert Syst. Appl. **168**, 114249 (2021)
9. Tamilarasi, K.: Design and implementation of deep learning strategy based smart signature verification system. Microprocess. Microsyst. **77**, 103119 (2020)
10. Ruiz, V., Linares, I., Sanchez, A., Velez, J.F.: Off-line handwritten signature verification using compositional synthetic generation of signatures and Siamese neural networks. Neurocomputing **374**, 30–41 (2020)
11. Hameed, M.M., Ahmad, R., Kiah, M.L.M., Murtaza, G.: Machine learning-based offline signature verification systems: a systematic review. Signal Process.: Image Commun. **116139** (2021)
12. Khalajzadeh, H., Mansouri, M., Teshnehlab, M.: Persian signature verification using convolutional neural networks. Int. J. Eng. Res. Technol. **1**(2), 7–12 (2012)
13. Jagtap, A.B., Hegadi, R.S., Santosh, K.C.: Feature learning for offline handwritten signature verification using convolutional neural network. Int. J. Technol. Human Interact. (IJTHI) **15**(4), 54–62 (2019)
14. Alajrami, E., Ashqar, B.A., Abu-Nasser, B.S., Khalil, A.J., Musleh, M.M., Barhoom, A.M., Abu-Naser, S.S.: Handwritten signature verification using deep learning (2020).
15. Shabbir, S., Malik, M.I., Siddiqi, I.: Offline Signature Verification Using Feature Learning and One-Class Classification. Pattern Recogn Artif Intell **1322**, 242 (2021)

16. Sharma, Y., Agrawal, G., Jain, P., Kumar, T.: Vector representation of words for sentiment analysis using GloVe. In: 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT), pp. 279–284. IEEE (2017)
17. Ram, S., Gupta, S., Agarwal, B.: Devanagri character recognition model using deep convolution neural network. J. Stat. Manag. Syst. **21**(4), 593–599 (2018). https://doi.org/10.1080/09720510.2018.1471264