International Journal of Academic Multidisciplinary Research (IJAMR)
ISSN: 2643-9670
Vol. 3 Issue 12, December – 2019, Pages: 39-44

Handwritten Signature Verification using Deep Learning

Eman Alajrami[1], Belal A. M. Ashqar[2], Bassem S. Abu-Nasser[2], Ahmed J. Khalil[2], Musleh M. Musleh[2], Alaa M. Barhoom[2], Samy S. Abu-Naser[2]

[1] Faculty of Information Technology, University of Palestine, Gaza, Palestine
[2]Department of Information Technology, Faculty of Engineering and Information Technology, Al-Azhar University, Gaza, Palestine

***Abstract**: Every person has his/her own unique signature that is used mainly for the purposes of personal identification and verification of important documents or legal transactions. There are two kinds of signature verification: static and dynamic. Static(off-line) verification is the process of verifying an electronic or document signature after it has been made, while dynamic(on-line) verification takes place as a person creates his/her signature on a digital tablet or a similar device. Offline signature verification is not efficient and slow for a large number of documents. To overcome the drawbacks of offline signature verification, we have seen a growth in online biometric personal verification such as fingerprints, eye scan etc. In this paper we created CNN model using python for offline signature and after training and validating, the accuracy of testing was 99.70%.*

Keywords: Handwritten, Signature, Verification, Deep Learning

1. INTRODUCTION

Signature verification and forgery detection is the process of verifying signatures automatically and instantly to determine whether the signature is real or not. There are two main kinds of signature verification: static and dynamic. Static, or off-line verification is the process of verifying a document signature after it has been made, while dynamic or on-line verification takes place as a person creates his/her signature on a digital tablet or a similar device. The signature in question is then compared to previous samples of that person's signature, which set up the database. In the case handwritten signature on a document, the computer needs the samples to be scanned for investigation, whereas a digital signature which is already stored in a data format can be used for signature verification. Handwritten signature is one of the most generally accepted personal attributes for verification with identity whether it may for banking or business [1,2].

There are several verification methods for offline signature verification which most of the companies use nowadays. Offline signatures use the static features of the system which involves image processing techniques to analyses the accuracy of the signatures. These include the initial identification of a person through the password. There are other multimodal systems that use the two different biometric features in order to strictly authenticate a person's identity.

2. RELATED WORKS

In [3, 4, 5], and [6], various explanations of offline signature verification are explored. In offline signature verification, template matching and Hidden Markov model techniques are generally employed. These techniques based on the structure of the signature. Template matching is a technique in digital image processing for fine small parts of an image which match a template image. When it comes to template matching, metrics like n square error or structural similarity index, or a warping method can be used which warps one curve onto another that the original shape is maintained. Authors in [7] show the use of Deep CNNs to identify who the signature belongs to and whether it is a forgery. This is done in a two-phase approach: writer-independent feature learning, and writer dependent classification. Working this paper simplifies this approach by treating the signatures and their forgeries as separate classes. It completely writer independent approach.

3. PROPOSED METHODOLOGY

The handwritten signature is a behavioral biometric which is not based on any physiology characteristics of the individual signature but on the behavior that change over time. Since an individual's signature alters over time the verification and authentication for the signature may take a long period which includes the errors to be higher in

some cases. Inconsistent signature leads to higher false rejection rates for an individual· who did not sign in a consistent way.

3.1 Data Acquisition

Handwritten signatures are collected and some unique features are extracted to create knowledge base for each and every individual. A standard database of signatures for every individual is needed for evaluating performance of the signature verification system and also for comparing the result obtained using other techniques' on the same database. Fig. 1 and Fig. 2 show samples of indivisible forged and real signatures respectively.

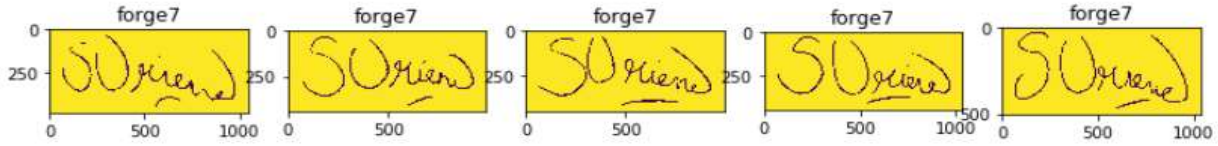

Fig.1. Example of forged Signature for user no. 7

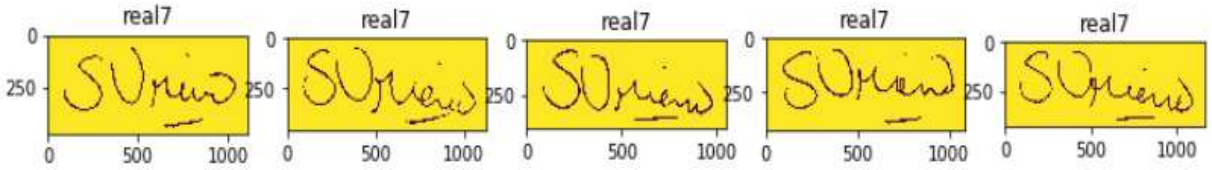

Fig.2. Examples of real Signature for user no. 7

3.2 Pre-processing

3.2.1 Resizing

The system must be able to maintain the high performance regardless of the size and slant given for the signature. It should be important that the system must be insensitive enough for the correction in the signature image. The image matrix is rescaled to standard resolution which is 512 X 512 in this case. The result of this operation is seen in Fig. 3 and Fig 4.

Fig. 3: Original Signature.

Fig. 4: Resized Signature.

3.2.2 File Management

Firstly, we take the name of the subject as an input. Then the directory of the raw images is loaded in to the file system. We have to include Real signatures as well as Forgeries in batches for bulk processing. The destination directory has to be included which consists of Training and Testing as sub-directories. This will make the system avoid directory error. Then, the images from the selected source directories, i.e., Real and Forgery are loaded as lists into the system. Each image from the list is loaded, processed and then final image file is created. The images are split into the respective ratio between the destination path sub-directories.

4. TRAINING THE MODEL

In this application, we use CNNs(or ConvNet) which is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. CNNs were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. In our work,

we use the Keras library with the TensorFlow backend to implement CNN. The directory of preprocessed images is loaded and then we train the model to evaluate the performance.

5. IMPLEMENTATION

In this work, the signature images are stored in a file directory structure which the Keras Python library can work with. Then the CNN has been implemented in python using the Keras with the TensorFlow backend to learn the patterns associated with the signatures. Then the model derived has been verified using accuracy and loss metrics to see how well the model has fit the data. Finally, the model has been tested by using a signature from a holdout set to see if the predictions are correct. Table1 contains a detailed architecture model of the implementation.

Table 1: Model architecture

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 510, 510, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 255, 255, 32)	0
conv2d_2 (Conv2D)	(None, 253, 253, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 126, 126, 64)	0
conv2d_3 (Conv2D)	(None, 124, 124, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 62, 62, 128)	0
conv2d_4 (Conv2D)	(None, 60, 60, 256)	295168
max_pooling2d_4 (MaxPooling2D)	(None, 30, 30, 256)	0
conv2d_5 (Conv2D)	(None, 28, 28, 256)	590080
max_pooling2d_5 (MaxPooling2 D)	(None, 14, 14, 256)	0
conv2d_6 (Conv2D)	(None, 12, 12, 512)	1180160
max_pooling2d_6 (MaxPooling2D)	(None, 6, 6, 512)	0
flatten_1 (Flatten)	(None, 18432)	0
dense_1 (Dense)	(None, 256)	4718848
dense_2 (Dense)	(None, 60)	15420

Table 2. List of formulas for the operations in the CNN

Operation	**Formula**
Convolution	$z^l = h^{l-1} * W^l$
Max Pooling	$h^l{}_{xy} = \max_{i=0..s,j=0..s} h^{l-1}(x+i)(y+j)$
Fully-connected layer	$z_l = W_l * h_{l-1}$
ReLu(Rectifier)	$ReLU(z_i) = \max(0, z_i)$
Softmax	$\text{softmax}(z_i) = e^{zi} / \sum_j e^{zj}$

In our implementation, an image goes through a series of convolution and max pooling layers which are in an alternating fashion. When the image goes through convolution process, a predefined number of feature maps are created which are fed into a max pooling layer, which creates pooled feature m aps from the feature maps received from the convolution layer which is before it. This pooled feature map is sent into the next convolution layer and this process continues until we reach the fourth max pooling layer. The pooled feature map from the last m ax pooling layer is flattened and sent into the fully connected layers. After several rounds of forward and backward propagation, the model is trained and a prediction can now be made. The formulae for all the steps of the CNN are mentioned in Table 2.

6. DATASET

The dataset which has been used in this research work is a collection of 10 signatures, with 5 Real and 5 forged signatures per individual. We have 30 person real signatures and this dataset was collected from Kaggle Dataset. The total images are 300 Images (150 Real and 150 forged). All the images are in RGB format.

7. BUILDING THE CONVOLUTIONAL NEURAL NETWORK

We import the necessary modules from the Keras API, which acts as a wrapper for the Tensorflow and Theano backends. We used the Tensorflow backend to build the model. The first python script trains the CNN with the Real and the forgery signatures as separate classes. We train the model with a 80-20 split(can be changed based on requirement) on the dataset and store the model as a .json file. We also store the weights of the model in a .h5 file. In the second script, we load the model from the .json file and the weights from the .h5 file, recompile the model, and then make out prediction on a test specimen from the holdout set. A third script plots the model's accuracy and loss with each epoch on the training and test sets.

8. RESULTS AND INTERACTIVE IMPLEMENTATION

In our work, We binarize the images and store them appropriately. We then do the required file handling and management operations to split the batches of images based on a required split ratio. After the models are built, plots of accuracy and loss are made, formulae of which are listed in Table 2.

We selected a signature from the holdout set (one Real and the other forged) as shown in Fig. 5.

Real signature forgery signature

Fig.5 Two sample signatures from test dataset

We create models for different splits of data and plot the training and validation accuracies to get an idea of the presence of any overfitting or underfitting.

On splitting the dataset in the ratio of 6:4 we obtained a maximum accuracy of 97% percent on the validation set as shown in Fig 6. There is very little overfitting as the training and testing accuracies are almost equal to each other.

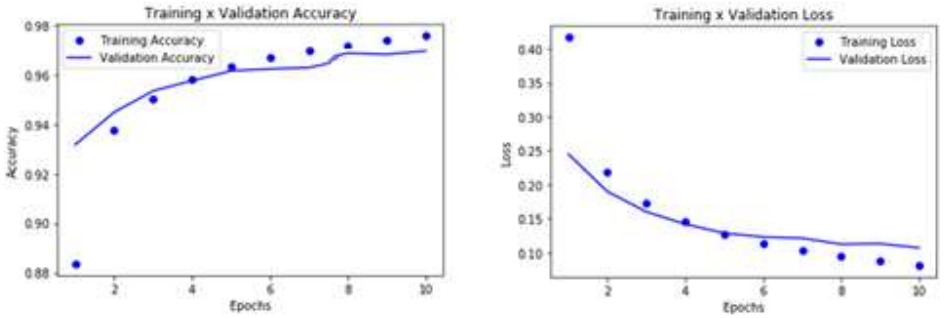

Fig.6 Model accuracy and loss for the split ratio of 6:4

On splitting our dataset in the ratio 7:3, we get a maximum accuracy of 98 percent on the validation set as shown in Fig. 7 at the third epoch. There is some underfitting here, and further epochs show overfitting.

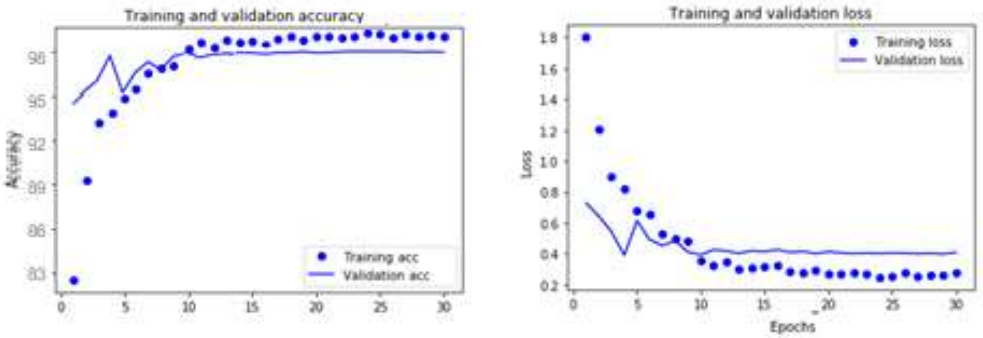

Fig. 7 Model accuracy and loss for the split ratio 7:3

On splitting the dataset in the ratio of 8:2 we obtained a maximum accuracy of 99.7 percent on validation set as shown in Fig. 8 at the fourth epoch.

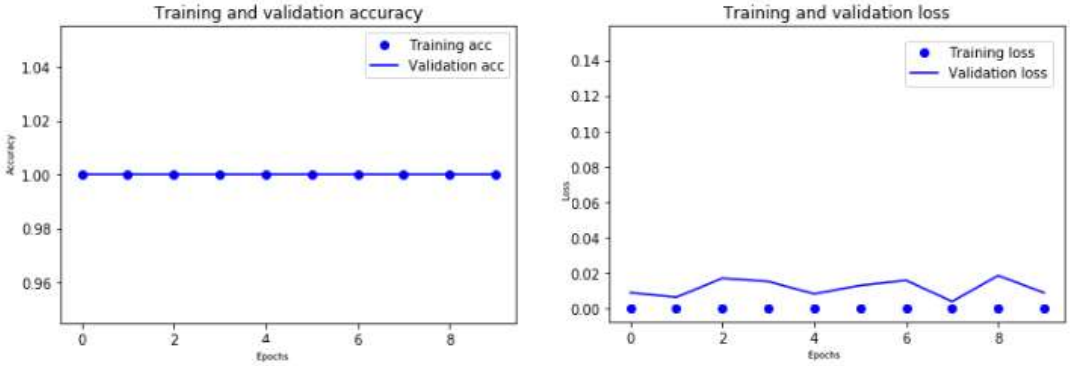

Fig. 8 Model accuracy and loss for the split ratio 8:2

The accuracies corresponding to the three splits of the dataset used is shown in Table 3. The criterion used to select the best model is the lowest difference between the training and validation accuracies.

Table 3: accuracies obtained from the splitting of the dataset

Split Ratio	Training Accuracy	Validation Accuracy
6:4	98.2%	97.0%
7:3	99.0%	98.0%
8:2	99.9%	99.7%

Based on the criterion set, the third model with the 80-20 data split offered the best result.

9. CONCLUSION

A model that can learn from signatures and make predictions as to whether the signature in question is a forgery or not, has been successfully implemented. This model can be deployed at various government offices where handwritten signatures are used as a means of approval or authentication. While this method uses CNNs to learn the signatures, the structure of our fully connected layer is not optimal. This implementation may be considered extreme. In the model created in this work, two classes are created for each user (Real and forgery). We have 30 users and thus we have a model with 60 classes to predict. The best accuracy we got was 99.7%.

10. REFERENCES

1. A. Karpathy ”Backpropagation, Intuitions” from Stanford’s CS231n, http://cs231n.github.io/optimization-2/, 2015
2. K. Simonyan, A. Zisserman ”Very Deep Convolutional Networks for Large-Scale Image Recognition” in ICLR 2015
3. S. Zagoruyko, N. Komodakis ”Learning to Compare Image Patches via Convolutional Neural Networks” CVPR 2015, arXiv:1504.03641v1
4. E. zgndz, T. entrk, and M. E. Karslgil, Off-line signature verification and recognition by support vector machine, in European signal processing conference, EUSIPCO, 2005.
5. E. J. Justino, A. El Yacoubi, F. Bortolozzi, and R. Sabourin, An off-line signature verification system using HMM and graphometric features, in Fourth IAPR International Workshop on Document Analysis Systems (DAS), Rio de. Citeseer, 2000, pp. 211222.
6. Yeung, D.-Y., Chang, H., Xiong, Y., George, S.E., Kashi, R.S., Matsumoto, T., Rigoll, G.: SVC2004: ”First International Signature Verification Competition” in ICBA 2004. LNCS, vol. 3072, pp. 16-22. Springer, Heidelberg.
7. Bailing Zhang. Off-line signature verification and identification by pyramid histogram of oriented gradients. International Journal of Intelligent Computing and Cybernetics, 3(4):611–630, 2010.
8. Z. Zhang, X. Liu, and Y. Cui. Multi-phase offline signature verification system using deep convolutional generative adversarial networks. In 2016 9th International Symposium on Computational Intelligence and Design (ISCID), volume 02, pages 103–107, 2016.
9. R. Zouari, R. Mokni, and M. Kherallah. Identification and verification system of offline handwritten signature using fractal approach. In Image Processing, Applications and Systems Conference (IPAS), 2014 First International, pages 1–4, November 2014.