Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 143 (2018) 978–987

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Conference on Advances in Computing and Communication (ICACC-2018)

Handwritten Signature Forgery Detection using Convolutional Neural Networks

Jerome Gideon S[a,*], Anurag Kandulna[b], Aron Abhishek Kujur[c], Diana A[d], Kumudha Raimond[e]

[a,b,c,d,e] *Department of Computer Sciences Technology, Karunya Institute of Technology and Sciences, Coimbatore, 641114, India*

Abstract

Handwritten signatures are very important in our social and legal life for verification and authentication. A signature can be accepted only if it is from the intended person. The probability of two signatures made by the same person being the same is very less. Many properties of the signature may vary even when two signatures are made by the same person. So, detecting a forgery becomes a challenging task. In this paper, a solution based on Convolutional Neural Network (CNN) is presented where the model is trained with a dataset of signatures, and predictions are made as to whether a provided signature is genuine or forged.

Keywords: Signature Forgery Detection; Convolutional Neural Networks; Machine Learning; Deep Learning

1. Introduction

A handwritten signature can be defined as the scripted name or legal mark of an individual, executed by hand for the purpose of authenticating writing in a permanent form. The acts of signing with a writing or marking instrument such as a pen or stylus is sealed on the paper. Ordway Hilton (who introduced A Review of Dynamic Handwritten Signature Verification) from an article in [1], introduced that the signature has at least three attributes, form, movement and variation. Since the signatures are produced by moving a pen on a paper, movement perhaps is the most important aspect of a signature.

* Corresponding author. Tel.: +919632454142;
E-mail address: sjeromegideon@karunya.edu.in

10.1016/j.procs.2018.10.336

Once a person is used to signing his signature, these nerve impulses are controlled by the brain without any particular attention to detail. Signature verification and forgery detection is the process of verifying signatures automatically and instantly to determine whether the signature is genuine or not. There are two main types of signature verification: static and dynamic. Static, or off-line verification is the process of verifying an electronic or paper signature after it has been made, while dynamic or on-line verification takes place as a subject creates his signature on a digital tablet or a similar device. The signature in question is then compared to previous samples of the target's signature, which constitute the database or knowledge base. In the case of an ink signature on paper, the computer requires the sample to be scanned for analysis, whereas a digital signature which is already stored in a data format can be used for signature verification.

Nowadays, handwritten signature is one of the most widely accepted personal attributes for confirmation with identity whether it may be from banking or business sector. The people from the lower society prefer to write their signature in free handwriting due to lack of education and knowledge. Therefore, these types of signatures are easy to forge in certain circumstances. In this case four types of forgeries are possible [2].

- **Simulation Forgery:**
  In which the forger has a sample of the signature to be forged. The quality of a simulation depends on how much the forger practices before attempting the actual forgery, the ability of the forger, and the forger's attention to detail in simulating the signature. Based on forger's experience, known forgeries are classified into unskilled and skilled forgeries [2].

- **Unknown/Random/Blind Forgery**:
  This is when the forger has no idea what the signature to be forged looks like. This is the easiest type of forgery to detect because it is usually not close to the appearance of a genuine signature. This type of forgery will sometimes allow an examiner to identify who made the forgery based on the handwriting habits that are present in the forged signature.

- **Tracing:**
  The third type of forgery is a tracing. Tracing can be done by holding the model document and the questioned document up to light and using a pen to trace the lines of the model signature onto the questioned document. A tracing can also be done by using a blunt stylus on the questioned document to create an impression of the model signature in the paper. This impression is then filled in with a pen to create the appearance of the model signature. If the model signature used by the forger is not found, this type of forgery is sometimes difficult to detect from a photocopy.

- **Optical Transfer**
  It is one in which, a genuine signature is transferred onto a document by use of a photocopier, scanner, facsimile machine, or photography. With this type of forgery, an examiner cannot positively identify a signature as genuine without having the Original for comparison.

Regardless, official documents are generally passed successfully once a signature is received, which may lead to many problems for the person involved. Signature forgery detection finds its application in the field of net banking, passport verification system, provides a form of confirmation of identity to candidates in public examinations, credit card transactions, and bank checks. Therefore, with the growing demand for protection of individual identity, the design of an automatic signature system is needed.

There are several verification methods for the online and offline signature verification which most of the companies use nowadays. Online systems use the dynamic features of a handwritten signature considering the time and frequency factors which involves signal processing techniques like the normalization, Fourier transforms and correlation functions for the proper analysis of the handwritten signatures. On the other hand, the offline signatures use the static features of the system which involves image processing techniques to analyse 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. Our project falls in the category of offline signature verification systems.

2. Related works

In [3], [4], and [5], various explanations of offline signature verification are explored. In offline signa 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 fin 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 anothe that the original shape is maintained.

In HMM, stochastic matching (model and the signature) is involved. This matching is done by step probability distribution of features involved in the signatures or the probability of how the original signatu calculated. If the results show a higher probability than the test signatures probability, then the signatures are by original person, otherwise the signatures are rejected. In both of these methods, a large number of data poin needed and significant preprocessing work should be done in order to reduce the threshold of false positives false negatives. Sometimes, totally different signatures can yield a positive result when an attempt is made to m them.

On the other hand, [6] uses online signature verification and goes towards deep learning. It captur signature live using specialized hardware (Anoto pen) to capture various dynamic features of a signature like number of strokes, the time taken and the stretching, and then sends it into a neural network to predict the validit a signature. While this system is robust, its deployment may tend to be really costly especially in a time where t is a transition towards more secure authentication methods.

[7] shows the use of Deep CNNs to identify who the signature belongs to and whether it is a forgery or This is done in a two-phase approach: writer-independent feature learning, and writer dependent classification. workin 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 behavioural biometric which is not based on any physiolog characteristics of the individual signature but on the behaviour that change over time. Since an individual's signa alters over time the verification and authentication for the signature may take a long period which include the for the errors to be higher in some cases. Inconsistent signature leads to higher false rejection rates for an indivi who did not sign in a consistent way.

3.1. Data Acquisition

Handwritten signatures are collected and some unique features are extracted to create knowledgebase 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 techni on the same database.

3.2 Pre-processing

- RGB to Grayscale

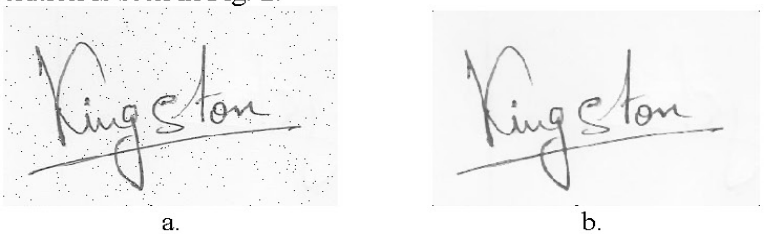
  In layman's terms, any RGB image is represented as matrix of X, Y dimensions and depth of 3 pla where each plane comprises of Red, Green and Blue values ranging from 0 to 255. Whereas, grays image is represented as matrix of X, Y dimensions and depth of only 1 plane. Each cell value ranges fro to 255.Any RGB (Red, Green, Blue) image which has to undergo Digital Image Processing (DIP) neec be converted to grayscale image. By doing this the computational complexity of DIP decreases drastic and helps to run image processing algorithms in much smoother way. This operation is seen in Fig. 1.

a. b.

Fig.1. Gray-scaling of a signature: a) Original Signature b) Grayscale signature

- Noise Removal

Noise is the result of errors which is caused due to various types of acquisition process which results in pixel values that do not reflect to the true intensities. The image which is taken from a camera contains the film grains in the source of noise, and also may be caused due to the damage when it is introduced in the scanner itself, and also electronic transmission of image can also introduce noise.Any form of noise present in the image is unknown to the analyst and its quantity is undefined. In order to improvise the noise removal process, a known form of noise is introduced in a very small quantity to the grayscale image. This ensures that the threshold of noise level in the image increases and henceforth easily detectable by de-noising filters. To remove the noise in the scanned image averaging and mean filters are used. This operation is seen in Fig. 2.



a. b.

Fig.2. Noise Removal process: a) Addition of salt and pepper noise. b) Removal of all noise

- Grayscale to Bitmap

The Grayscale image format is converted into bitmap where image file format is used to store digital images. Raster image is in general is also referred to as a bitmap. The result of this operation is seen in Fig. 3.

- 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 256 X 256 in this case. The result of this operation is seen in Fig. 4.

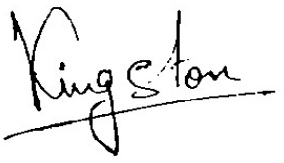


Fig.3. Grayscale without noise, to bitmap.

Fig.4. Resized bitmap.

- File Management

Firstly, we take the name of the subject as an input. It is later used to create CSV (Comma Separated Values) file. Then the directory of the raw images is loaded in to the file system. We have to

include Genuine 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 to avoid directory error. Then, the images from the selected source directories, i.e, Genuine 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. Once a complete batch of images is processed the CSV file is updated.

3.3. Training the model

In this application, we use CNNs. ACNN (or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analysing 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 with different training-testing split ratios to evaluate the performance.

4. Implementation

In this work, the signature images are pre-processed in a batch by batch mannerand split them into training and test sets based on a split ratio (which is chosen). This is done in MatLab, with functions from the image processing toolbox. After these signatures are preprocessed, it is stored in a file directory structure which the keras python library can work with. Then the CNN has been implemented in python using the Keraswith 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.Fig. 5 contains a detailed architecture diagram of the implementation.

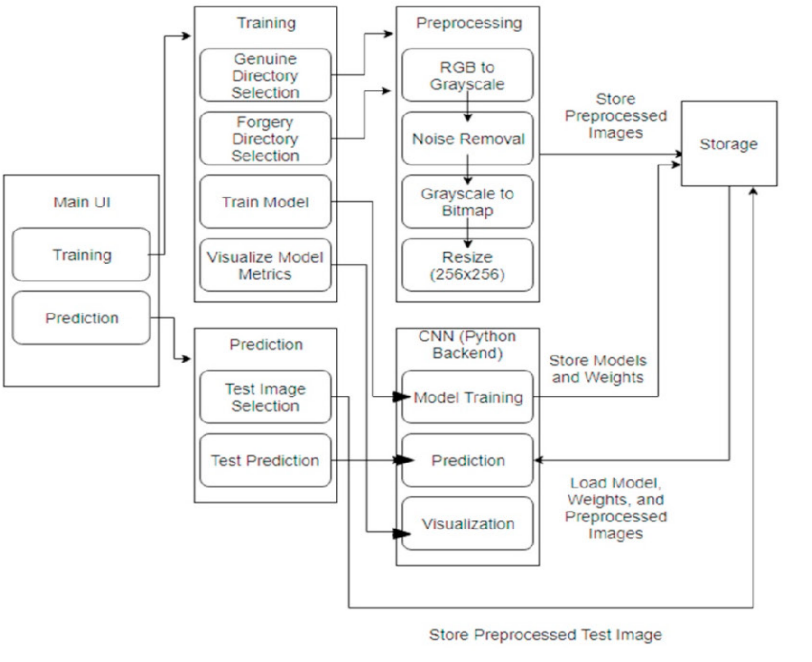


Fig.5. Architecture Diagram.

Table 1. List of formulae 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	zl = Wlhl−1
ReLu (Rectifier)	$ReLU(z_i) = \max(0, z_i)$
Softmax	$\text{softmax}(z_i) = e^{z_i}/\Sigma_j e^{z_j}$

In our implementation (as shown in Fig. 6), an image goes through 3 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 maps 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 third max pooling layer. The pooled feature map from the last max 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 1.

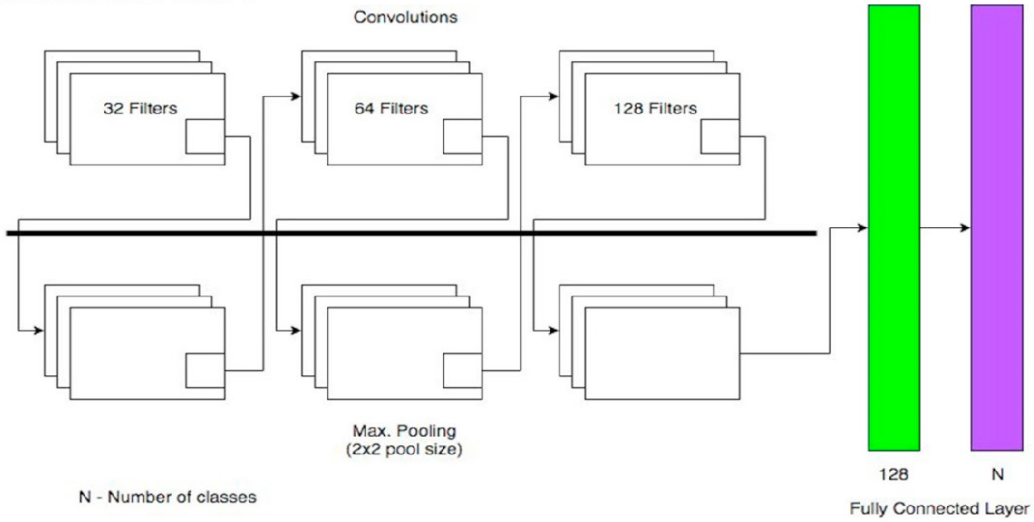


Fig.6. Illustration of the CNN architecture used in this work.

- Dataset
  The dataset which has been used in this research work is a collection of 6000 signatures, with 1000 genuine and 1000 forged signatures per subject. This dataset was carefully prepared by the authors with one person making the originals and two others making the forgeries. All the images are in RGB format. For this work, two datasets have been formed from the same set of 6000 images. The first dataset has a split ratio of 4:1, and the second has a split ratio of 7:3. Additionally, there are 16 extra images (8 genuine and 8 forgeries) per subject for the holdout set.

- GUI
  GUIs provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application.MATLABapps are self-contained programs with GUI front ends that automate a task or calculation. The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox, Signal Processing Toolbox, and Control System Toolbox include apps with custom user interfaces. We can also create our own custom apps, including their corresponding UIs, for others to use. We use these tools that are available in

MATLAB to create our GUI.

- Pre-processing
The Matlab application has Image Processing Toolbox. This toolbox provides a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization, and algorithm development. We can perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration, and 3D image processing. Image Processing Toolbox apps let us automate common image processing workflows. We can interactively segment image data, compare image registration techniques, and batch-process large datasets.

- 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 genuine 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.

5. Results and Interactive Implementation

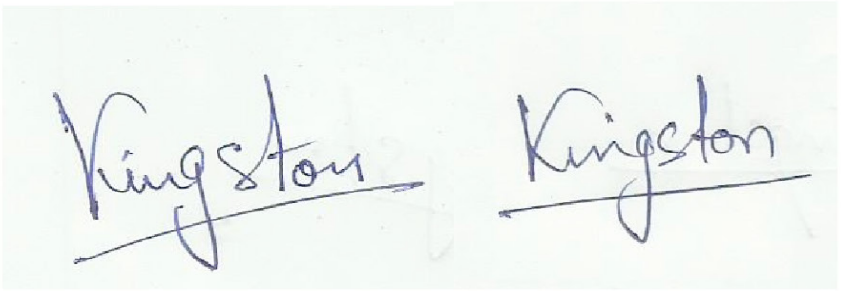
In our work, we first turn the raw RGB images into grayscale. We then add salt and pepper noise of density 0.01 and remove all the noise using average and mean filters. We then binarize the images and store them appropriately. We then do the required file handling and management operations to split the 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.

Table 2. List of formulae for calculating metrics for the plots of accuracy and loss.

Operation	Formula
Accuracy	$\Sigma n_{correct}/N$
Loss	$H(p, q) = -\Sigma p(x) \log q(x)$

N – Total no. of predictions; p – True distribution; q – coding distribution

We select a signature from the holdout set as shown in Fig. 7. Fig. 8 shows the results of predictionusing the model.



a b

Fig.7. Two sample signatures from input dataset: a) genuine signature and b) forgery.

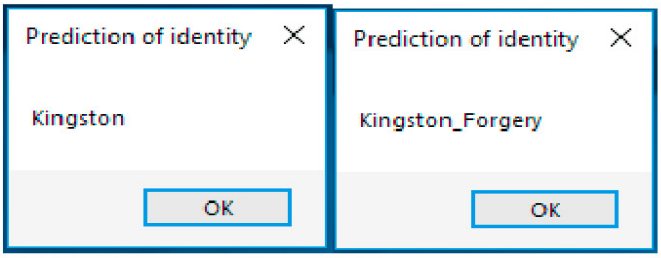


Fig.8. Test Results of the signatures

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 8:2 we obtained a maximum accuracyof ~98 percent on the validation set as shown in Fig 9. There is very little overfitting as the training and testing accuracies are almost equal to each other.

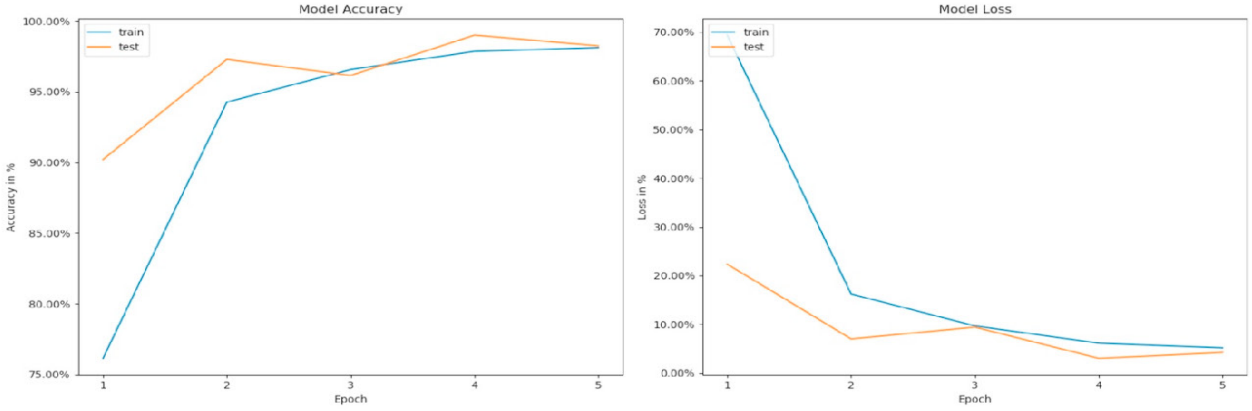


Fig.9. Model accuracy and loss for the split ratio of 4:1

On splitting our dataset in the ratio 7:3, we get a maximum accuracy of ~96 percent on the validation set as shown in Fig 10.at the third epoch. There is some underfitting here, and further epochs showoverfitting. The validation accuracy at the fifth epoch is ~94 percent.

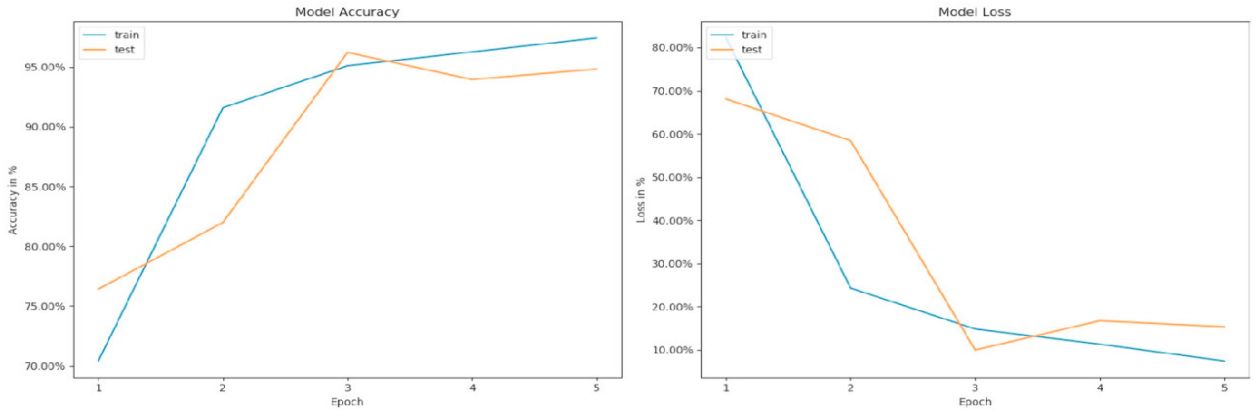


Fig.10. Model accuracy and loss for the split ratio 7:3

On splitting the dataset in the ratio of 6:4 we obtained a maximum accuracy of ~97 percent on validation set as shown in Fig 11 at the fourth epoch. There is some underfitting here, and the next epoch shows some overfitting.

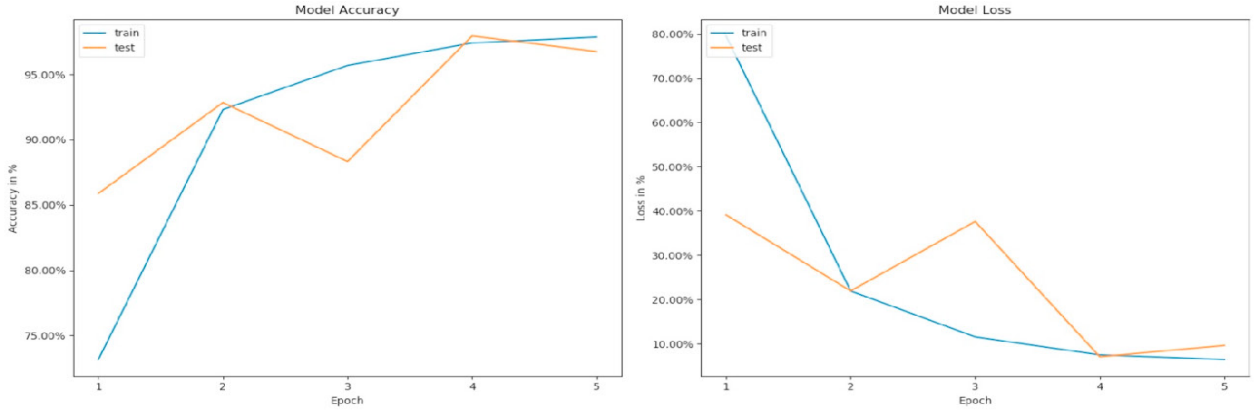


Fig.11. Model accuracy and loss for the split ratio 3:2

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 for the two split ratios of the dataset used at the fifth epoch.

Split	Training Accuracy	Validation Accuracy
8:2	98.11%	98.23%
7:3	97.42%	94.83%
6:4	97.86%	96.75%

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

6. Conclusion

A system 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 system 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. According to [7], this implementation may be considered extreme. In the model created in this work, two classes are created for each user (genuine and forgery). If the genuine and forgery signatures of 100 people are given, then the model would have 200 classes to predict, which would make the learning process longer. One future enhancement would be to do comprehensive research on loss functions and derive a custom loss functions (preferably two) which would predict the user to which the signature belongs to, and whether it is a forgery or not.

7. References

[1]Hilton O, Scientific Examination of Questioned Documents
https://books.google.co.in/books/about/Scientific_Examination_of_Questioned_Doc.html?id=nQlJw2_gE-cC&redir_esc=y
[2]Specin Forensics LLC, Handwriting and Forgery Examination
http://4n6.com/handwriting-and-forgery-examination/
[3]Zaidi S.F.A., Mohammed S, Biometric Handwritten Signature Recognition, TDDD17: Information Security Course, Linköpingsuniversitet, Sweden
[4]Srinivasan H., Srihari S.N., Beal M.J. (2006) Machine Learning for Signature Verification. In: Kalra P.K., Peleg S. (eds) Computer Vision, Graphics and Image Processing. Lecture Notes in Computer Science, vol 4338. Springer, Berlin, Heidelberg
[5]Bhattacharya I., GhoshP.,Biswas S. (2013) Offline Signature Verification Using Pixel Matching Technique http://www.sciencedirect.com/science/article/pii/ S2212017313006075
[6]Drott B. and Hassan-Reza T. On-line Handwritten Signature Verification using Machine Learning Techniques with a Deep Learning Approach (2015), In Master's Theses in Mathematical Sciences FMA820 20151, Mathematics (Faculty of Engineering) http://lup.lub.lu.se/student-papers/record/8055778
[7] Hafemann L. G., Sabourin R., Oliveira L.S., Learning features for offline handwritten signature verification using deep convolutional neural networks, Pattern Recognition, Volume 70, 2017, Pages 163-176, ISSN 0031-3203,https://doi.org/ 10.1016/j.patcog.2017.05.012.(http://www.sciencedirect.com/science/article/pii/ S0031320317302017)