Prachi Kumar

Department of Artificial Intelligence and Data Science

Vishwakarma Institute of Technology, Pune

[[1]](#footnote-1)

Signature Forgery Detection & Recognition System

.

*Abstract*—*Every individual has a distinctive signature that is primarily used for personal identification and to confirm the authenticity of significant papers or legal transactions. Static and dynamic signature verification are the two types of verification available. Dynamic (on-line) verification occurs as a person makes his or her signature on a digital tablet or a similar device, whereas static (off-line) verification occurs after the person has signed. For a lot of papers, offline signature verification is ineffective and sluggish. We have observed an increase in online biometric personal verification such as fingerprints, eye scans, etc. only to get around the limitations of offline signature verification. Using Python and machine learning, an offline signature verification model is being developed in this research, and after training and validating, the testing accuracy was highly decent.*

***Keywords*** *— Matplotlib, TensorFlow, Computer Vision, Keras, Euclidean distance*, *TF model*

1. **INTRODUCTION**

The act of automatically and instantaneously confirming signatures to determine whether or not they are authentic is known as "signature verification and forgery detection." Static and dynamic signature verification are the two primary types. Dynamic or on-line verification occurs when a person signs a document using a digital tablet or a similar device, as opposed to static or off-line verification, which takes place after the signature has been made. Following that, the signature in question is compared against earlier examples of that person's signature that were used to create the database. While a digital signature that is already saved in a data format may be utilized for signature verification, a handwritten signature on a document requires the computer to scan samples in order to conduct an inquiry. One of the most often used personal characteristics for confirming identification, whether it be for banking or business, is a handwritten signature.

Based on their distinguishing characteristics, handwritten signature forgeries have been divided into several categories. The following forms of signature forgeries can be generally categorized:

1. Random Forgery - Also known as a simple forgery and random forgery, this forgery is created by the signer using the victim's name in his own unique manner.

2. Unskilled Forgery: The signer tries to imitate the signature in his own manner without having any prior expertise or understanding of the spelling.

3. Skilled Forgery - Without a doubt, the most challenging forgeries are produced by experienced copycats or professional impostors.

In this work, we'll show how to use several geometric measurements to accomplish offline verification. I utilized python libraries like TensorFlow, Matplotlib, Pandas, and Keras in my project and make a tensor flow and neural network model.

The objective of this project, signature forgery detection and recognition system (SRVS) is basically to be able to address two individuals:

1. Identification of the signature owner
2. Decision whether the signature is genuine or forged

So the model will be able to detect if the signature the system gets is forged or real along with the person, if in the database, actually attempted to forge the signature. This is possible because when a man is writing his signature, the information or attributes like hand speed and pressure at certain points are unique and there are many other attributes which we can take as key point to identify between a real or forged image.

**RELATED WORK**

[1] Offered an off-line handwritten signature verification approach based on convolution neural networks in this research (CNN). Net banking, passport verification systems, credit card transactions, and bank checks are all examples of when signature forgery detection is used. The goal of this software was to determine whether a signature was genuine or counterfeit, to comprehend signature features, and to execute the system. This study used convolution filtering by back-propagation algorithm. CNN was used as a feature extractor and classifier. Drawbacks of CNN were mentioned such as the gradient decreasing exponentially and quickly to zero as they back propagate from the final layer, back to the first layer.

[2] This study explained how the brain controls one's nerve impulses by not being able to pay attention to details, which is why if one tries to forge, it will be detected. The study went on to explain the various types of forgeries that can be done, including Simulation Forgery, Blind Forgery, Tracing, and Optical Transfer.

The model described in this research collected data by collecting numerous signatures before moving on to the pre-processing stage, where the images were transformed from RGB to Grayscale using DIP. Before training the model, moisture was removed, the image was converted to a bitmap, resized, and a CSV file was created.

The model was trained using CNN and ACNN, and the data was split and trained/tested using MatLab.

[3] They investigated a two-part approach for detecting splicing forgeries. A coarseto-refined convolutional neural network (C2RNet) and a diluted adaptive clustering network make up the two elements. The suggested model finds picture differences by cascading a coarse CNN and a refined CNN (C-CNN and R-CNN respectively). As a result of the cascading, scales where the image has been tampered are created, with differences in their attributes. Instead of employing a patch level CNN in C2RNet, an image level CNN reduces the computational cost of the entire model. Because the differences in attributes are compared, the findings are stabilized. Even in assault situations, it was discovered that the suggested method outperforms existing splicing strategies for forgery detection. The scale of these datasets, however, limits training, hence the suggested model has not yet produced ideal results.

[4] This study have proposed using Siamese Neural Networks for forgery detection in an offline signature verification procedure with a writer-independent context. Adding new signals for verification does not necessitate further training. In addition, three types of data were analyzed in order to enhance the number of samples and variability required for deep neural network training. The GAVAB dataset and the GPDS Synthetic dataset were used to create compositional synthetic signatures from primitives shapes.  The first two approaches can be classified as on-demand approaches since they can generate an endless number of synthetic signatures. Using the first described dataset, Siamese Neural Networks are used to train the provided technique. Further few other datasets such as GPSythetic, MCYT, Sigcomp11 etc, were also used to test and create a general model.

[5] To validate signatures, a signature verification system based on the Error Back Propagation Training Algorithm developed using the MATLAB Neural Network Toolbox has been suggested. The low cost, little intrusion, good performance, and use of an accepted and natural biometric are all appealing qualities of this technology (the signature). A two-stage procedure is proposed, with the first phase using signature identification and the second step involving individual verification. Both processes are completed by Neural Networks that have been trained with the Error Back-Propagation Training Algorithm.

[6] According to this report, biometric traits are becoming increasingly important in authentication systems. Signatures are one of the most essential and well-known biometrics. They propose a system with two distinct phases for offline signature identification and verification in this paper. Triangular Spatial Relationship (TSR), a rotation invariant feature extraction approach, is used in the identification step. To make TSR possible, a symbolic representation of signature has also been used. A hybrid method that incorporates Discrete Wavelet Transform (DWT), Gabor filter, and image fusion methods is proposed in the verification phase. The suggested method's robustness and precision, as well as its robustness against translation, scaling, and rotation, have been proven by experimental findings on several benchmarks.

[7] The of Neural network ease of usage and quick processing, signatures are a helpful biological tool for recognizing people. Offline signature identification and verification systems based on neural networks are discussed in this paper. This approach allows the user to determine whether a signature is genuine or counterfeit. There are three primary processes in the proposed system: picture pre processing, feature extraction, and classification. With a 93.5 percent identification rate, this system can recognize 700 signatures, and with a 97 percent verification rate, it can separate genuine signatures from fake ones.

[8] Presented a score-level fusion of complementary classifiers that use different local features (histogram of oriented gradients, local binary patterns, and scale invariant feature transform descriptors), where each classifier uses a feature level fusion to represent local features at coarse-to-fine levels. Two different approaches to classifiers are studied, namely global and user-dependent classifiers.

User-dependent classifiers are trained separately for each user to learn to distinguish that user's genuine signatures from other signatures, whereas a single global classifier is trained with difference vectors of query and reference signatures from all users in the training set to learn the importance of various types of dissimilarities.

# **METHODOLOGY**

First, we will use a database of signatures to teach the system. The system has already granted permission for this database of signatures. In order to verify other signatures, this system will analyze all signatures and create a reference signature.

Between the claimed signature and the signature that will be scrutinized, there are several parameters (such as the Euclidean Distance) in the feature space.

The signature is recognized as a counterfeit if the absolute difference between the parameters of the original signature and the verification signature exceeds a certain threshold.

1. ***Libraries used:***

➢ TensorFlow:

1. One of the well-known and free open-source

libraries of open-source software are used for machine learning.

2. It is utilized for a variety of purposes, with deep neural network training receiving special attention.

3. It is a dataflow-based math library.

➢ Matplotlib:

1. NumPy is a Python charting package that has been developed to include numerical mathematics.

2. It provides an object-oriented API that is employed in incorporating graphs into apps using of wide application GUI

➢Pandas:

1. Data processing and analysis are its principal uses.

2. It offers procedures and data structures to manage numerical tables and time series generally..

➢ Keras:

1. It gives a Python interface for artificial neural networks and is an open-source software library.

2. It serves as a TensorFlow library interface.

1. ***Dataset used:***

The project used a part of the dataset available on Kaggle. A selected lot of signature were taken. Each person that provided their signature signed five different times. A set of five forged signatures of each signature was also taken. Hence, the dataset had two folders, ‘forged’ and ‘real’. The signature picture files were named according to an ID number.

1. ***Proposed model:***

The system will start by prompting user for an user ID. The user will be required to key in his/her ID to the system and the system will decide whether the

user ID is registered or not. If the user is not registered that would mean the system do not have that person’s ID in the database to train on and decide whether the signature will be forged or not. Hence, If the user ID is not registered, the user has to drop down 5 sets of

signature for training purpose.

If the user ID entered is registered, the signature image that will be input in the system will be able to tell if that signature is forged or not straight away. As, the input training signatures and the input testing signature will proceed to go in the pre-processing stage. After this stage, the input training signature will be

saved as reference signature in the database while

the input testing signature will be saved as sample

signature and proceed to Verification stage.

In the Verification stage, the sample signature

will be compared with the reference signature

which stored in the database. If the difference between two signatures does not exceed the Threshold values of the different features in the csv files, the sample signature will be accepted as genuine signature and if they do, signature will be rejected as forged.

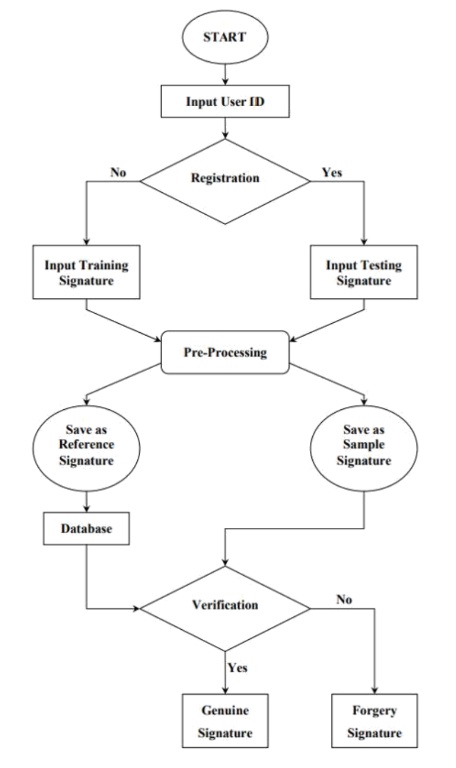


Fig 1. Flow of proposed model

1. ***Model building:***
2. Import libraries listed in section a.
3. Set path to the database location so model can access the database.
4. Pre-processing stage:

* Convert the image to grey scale
* Convert from grey scale to binary by using ‘blur\_radius’ of 0.8 which helps minimize the blur or noise

1. Feature extraction:
   * The different features the model was made to study on were, ratio, centroid, eccentricity solidity, skew kurtosis
   * Made a getCSV function to save all the features of the different signatures in a csv file, so the features can be retrieved from the csv file
2. Saving the features
   * The model will create a ‘Feature’ folder at the location of the path we give the model
   * That folder will contain all the data in form of csv files of each user id that is in database
   * The features folder will have sub folder of training and testing too
   * A test feature file will be created separately. It will contain the features of the signature we pass into the model to test
3. Making TF model
   * Consist of three layers and one output layer
   * Prediction of the image will generate an array that will have two columns; confidence value against genuine and confidence against forged

# **RESULTS**

As a result, our model will be able to tell if the image part we entered into the model is forged or real. We will be able to check the model’s result by tracing back to the image file number to ensure the model has given the correct output.

Format of signature image path in our model is XXXZZZ\_YYY.png

XXX refers to the id of the person who has signed on the document(i.e -001)

ZZZ refers to the id of the person to whom the sign belongs in actual(i.e- 001)

YYY deontes the n'th number of attempt

We can verify if the model output is correct by the file name as XXX == ZZZ then image is genuine otherwise the signature is forged.

Taking a case example to demonstrate, a person comes into a bank claiming they are person ID 3, the bank worker shall input ‘003’ in the id and ask for a signature from that person.

The image of that signature will be put in the signature folder and its path will be entered in the model.

The model will then verify but comparing the test signature with the other five signatures user id 003 has provided earlier from the database.

In this case, the person claiming to be person 003 was fraud hence the model detected the signature to be false.

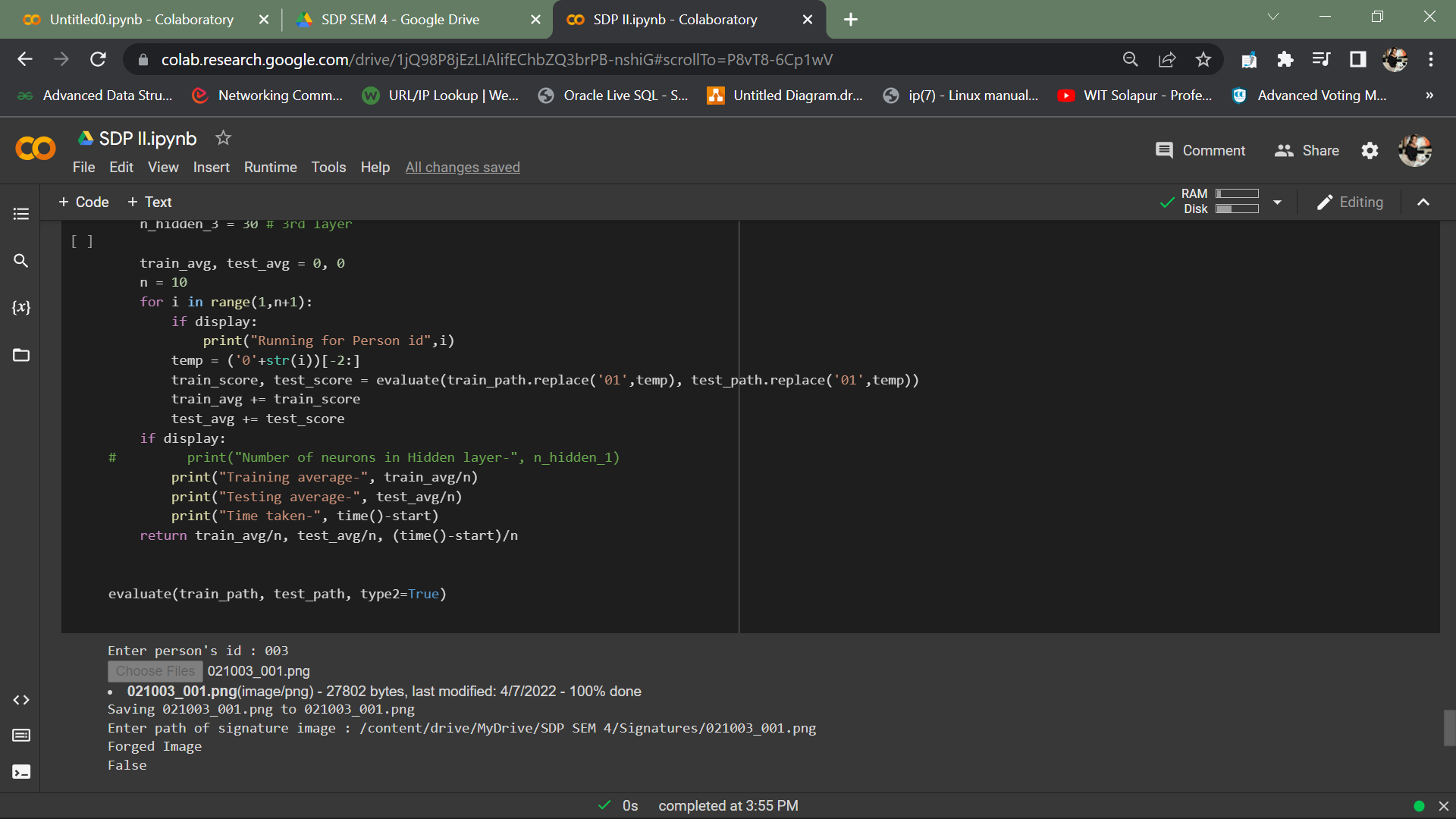


Fig 2. Output of model incase of forgery

As this model is not deployed out and the proof of working is necessary, we should be able to check the the model’s output. This can be done so by

(i) Tracing the signature file name. In the output above, the image name is, 021003\_001 when we check in our database where this image number is stored, we will find it to be in the forged folder. Hence the model output was accurate

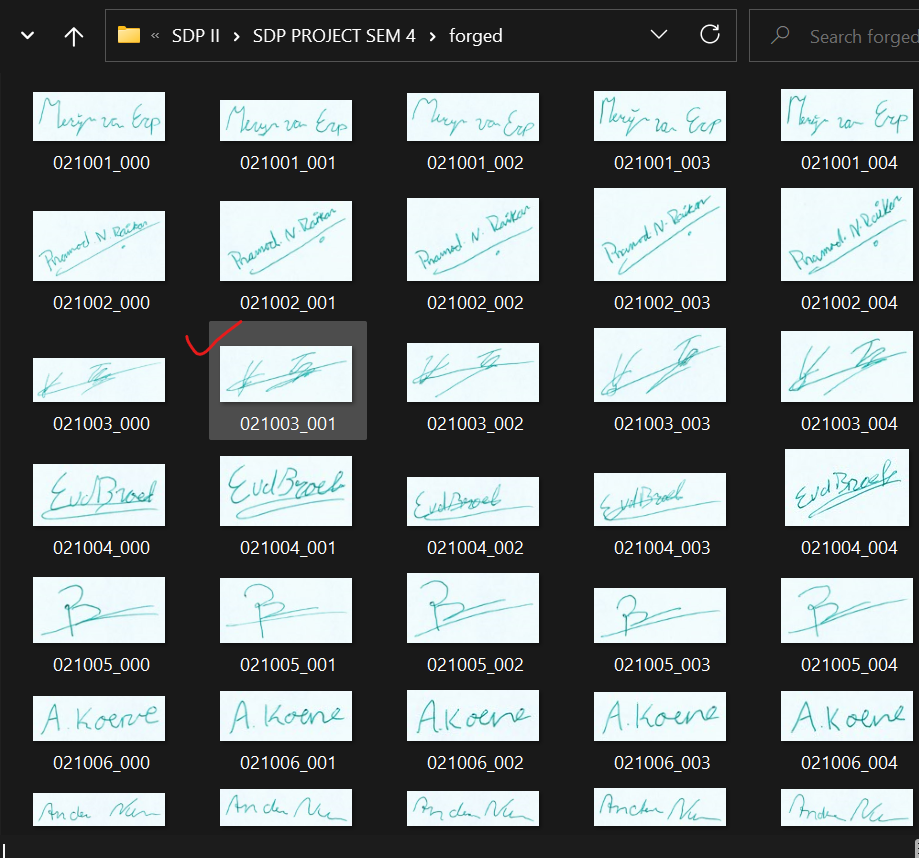


Fig 3. The signature being traced to forged folder in system

(ii) Or we can straight away tell from the file name, 021!=003, hence it is indeed a forged signature and our model output was accurate

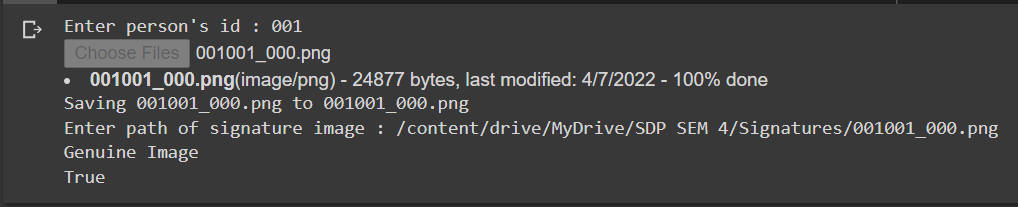


Fig 3. Output incase of real signature

The way we know this is an accurate out given by the model will be the same as described for forged output case.

# **CONCLUSION**

The model, which incorporates structural characteristics in its exponential membership function, models an off-line signature verification and forgery detection system. The box technique is used to extract the features made up of angles. Due to the variances in handwritten signatures, dataset of each user containing the feature values, produces a csv file set when the model is training on the dataset provided of the users. Each rule in this formulation is represented by a single feature. On a vast database of signatures, this system's effectiveness has been evaluated. The verification method can accurately identify all kinds of forgeries, including random, expert, and unskilled ones. The first parameter selection is vital but not absolutely necessary. However, we only need to choose well once, and it applies to all different kinds of signatures. Due to lack of simplicity during the implementation stage, we haven't implemented global learning approaches.

# **REFRENCES**

[1] Handwritten Signature Forgery Detection, Kshitij Swapnil,Udit Amit Patel, Rushab Kheni, 2021

[2] Handwritten Signature Forgery Detection using Convolutional Neural Networks, Jerome gideion, Anurag Kandulna, Aron Abhishek Kujur, Diana A, Kumudha Raimond, 2018

[3] Image Splicing Forgery Detection Combining Coarse to Refined Convolutional Neural Network and Adaptive Clustering, Bin Xiao, Yang Wei, Xiuli Bi, Weisheng Li , 2020

[4] Off-line Handwritten Signature Verification Using Compositional Synthetic Generation Of Signatures And Siamese Neural Networks, Ruiz, V., Linares, I., Sanchez, A., Velez, J. F, 2020

[5] Online Signature Forgery Prevention, Bhatia, Shalini, Pratik Nagpal, Dheeraj,Nayak Sandhya, 2013

[6] A new system for offline signature identification and verification, Ghandali, S., M.

Moghaddam , M. Khosravi, 2012

[7] High Performance Offline Signature Verification and Recognition Method using Neural Network, Saeid Fazli, Shima Pouyan, 2015

[8] Score Level Fusion of Classifiers in Off-line Signature Verification, Yilmaz, Yanikoglu, 2016

1. [↑](#footnote-ref-1)