Online Signature Verification using Deep Descriptors

Abigail Singh and Serestina Viriri
School of Mathematic, Statistics and Computer Science
University of Kwa-Zulu Natal
Durban, South Africa

2020 Conference on Information Communications Technology and Society (ICTAS) 978-1-7281-3770-4/20/$31.00 ©2020 IEEE 10.1109/ICTAS47918.2020.233999

***Abstract*—Signature verification is a technique used to counter signature forgery. In the past, the process began with staff at a bank, who is an expert, would confirm if a signature is genuine or forged. With the development of technology, now people no longer sign on paper, rather on a digital pad which can take more data which is recorded on paper, for example, pressure, azimuth and altitude angles. Examples of details captured from a digital pen include pen pressure, azimuth and altitude angles. This data is now used in various dynamic signature verification systems that achieve high accuracy on evaluation tests using different forms of artificial intelligence. This paper investigates using artificial intelligence in the form of a Convolutional Neural Network (CNN) followed by a Recurrent Neural Network (RNN) to verify signatures using the SVC 2004 and SigComp2009 online datasets and it achieved a testing accuracy of 97.05%.**

Keywords*—Signature; Signature forgery; Artificial Intelligence; Convolutional Neural Network; Recurrent Neural Network.

I. INTRODUCTION

A. Signatures and Counterfeit Signatures

Biometrics is defined as the measurement and statistical analysis of people's unique, universal, permanent and collectable physical and behavioural characteristics. From this definition, we draw the following two facts: each person biometrics' are unique hence it can be used for verification and identification and a person's biometrics is split into two categories: physical and behavioural. Examples of physical biometrics are a person's fingerprints, iris and face. An example of a person's behavioural biometrics is their voice. A signature of a person is a behavioural biometric. It is the more special biometric as it was the first behavioural biometric used for the verification of a person for legal matters in human history.

Signatures came into existence in 3100 BC which was an autograph by scribe Gar.Ama that was written on a Sumerian clay table. The first signature of a historical figure, however, dates to 1098 which belonged to the Spanish nobleman and military leader El Cid [1].

Presently, every person is required to have a signature, whether it is to open a bank account, obtain their driver's license or buy a home in any country. The importance of signatures has grown exponentially from 1098 to now and with the rise of this importance, came the rise of signature forgery. This gave way to several types of fraud; identity theft, check and credit card fraud are just a few examples. Some people specialize in forging other people's signatures whose forgeries are so deceptive to the naked eye.

A counterfeit signature or forgery has one of three forms:

1) Random forgery is when the person who forges the genuine signature without any information about the genuine signer.

2) Simple forger is when the forger forges the signature with the same shape of the genuine signature.

3) Skilled forgery is when the forger forges the signature with full information of the genuine signature and practices that signature to the nest of their ability.

B. Signature verification

In order to verify a signature, any verification process uses features. These features take the form of one of two types (2) (3):

1) Global features which are easily extracted from the whole signature and tough to noise. It does, however, deliver limited information. Examples are block codes, Fourier and Wavelet series.

2) Local features which are calculated to describe the geometrical characteristics and provides substantial descriptions of writing shapes. It is, however, difficult to extract. Examples are curving, location and tangent track.

Signature verification has two approaches [2][3]:

1) Offline or Static Signature Verification

In this approach, the signature that is being verified is stored as an image which is scanned from the piece of paper that has the signature in question on it. Hence this techniques uses several image processing techniques before actually implementing any type of classification which can delay the process of training and the overall verification of a signature.

2) Online or Dynamic Signature Verification

This approach, on the other hand, stores the signature as numbers that were obtained from a digital tablet or pad. The numbers represent various features, like velocity, pressure, various angles, etc. which makes it easier to extract local features (this is the reason why local features are preferred in this approach rather than offline signature verification).

The paper has the following sections. Section II mentions some related work in the signature verification field. Section III mentions the data, datasets and methodology used in this experiment. Section IV discusses some results from the experiment. Section V consists of a discussion.

II. Related Work

Online and offline signature verification using artificial intelligence in the form of neural networks has applied numerous amounts of times. For offline signature verification, the most common forms of artificial intelligence are CNN and Support Vector Machines (SVMs). For online signature verification, the most common forms of artificial intelligence are basic artificial neural networks.

Babita P. of the Jorhat Engineering College in Assam, India wrote and published a research article "Online Signature Recognition using Neural Networks" [4] in 2015. In the article, she created a neural network using backpropagation. The paper concluded with a 95% accuracy rate using the ATVS dataset.

CNNs were also used numerously as they are usually used in object recognition. One example is by B. Cozzens, R. Huang, M. Jay, K. Khembunjong, S. Paliskara, F. Zhan, M. Zhang and S. Tayeb from Las Vegas, Nevada who jointly wrote and published the research article "Signature Verification Using a Convolutional Neural Network" [5]. The system they created worked on images of signatures (i.e. offline signatures). They implemented a CNN and trained it using genuine and forged signatures separately from the SigComp 2011 dataset. It achieved 98.8% accuracy on the training set, 95% of the training set and 85.43% on the reference set of the genuine dataset and 85% accuracy on the training set, 82.09% of the training set and 83.12% on the reference set of the forged dataset. The difference in these percentages is the fact that the number of signatures in the forged training and validation sets are significantly lower than that in the genuine sets.

RNNs are used in natural language processing (NLP) hence Songxuan Lai and Lianwen Jin, Weixin Yang tested RNN's ability to differentiate between genuine and forged signatures in their article "Online Signature Verification using Recurrent Neural Network and Length-normalized Path Signature Descriptor" [6] in May 2017 using the SVC 2004 dataset. The researchers concluded with an equal error rate (EER), which is used to predetermine the threshold values for its false acceptance rate (FAR) and its false rejection rate (FRR) in a biometric security system, of 2.37%.

The power of CNNs lays behind the fact that they are excellent feature extractors. Seungsoo Nam, Hosung Park, Changho Seo and Daeseon Choi integrated this with an autoencoder in their paper "Forged Signature Distinction Using Convolutional Neural Network for Feature Extraction" [7]. The researchers' implementation was unique in the fact that they combined the use of CNN in the form of a feature extractor and an autoencoder as the classifier. The researchers created their dataset with signatures that were taken with some period between them and skilled forgeries. Their paper concluded with an EER of 4.5% for the time difference dataset and 2.7% for the skilled forgery dataset.

III. Methodology

A. Dataset

As stated earlier, this paper will research online signature verification. Because of security reasons, most datasets contain very few samples of genuine and more of forgeries (as there is less risk involved) hence, this paper uses two datasets to even out the number of genuine and forged signatures.

1) SigComp 2009 Dataset

This dataset [8][9] was created using 12 genuine signers, whom each signed 5 times, and 31 skilled forgers, who also sign 5 times per genuine signer. The collection of the dataset occurred by each person signing 5 times on an A4 sheet of paper that was placed on top of a Wacom Intuos2 A4-oversized tablet using a Wacom Intuos black inking pen.

Each one of those signatures was stored as an HWR file (HandWriting Recording file) which is a text file with the following format:

<label> <npoints>

The <label> is the filename, without the ".hwr" extension and <npoints> denotes the number of subsequent sampled trajectory points.

Each point contains a five-tuple recorded pen-tip coordinates (x,y,p,a,e):

x = x-position on tablet (raw tablet values)

y = y-position on tablet (raw tablet values) with (0,0) being the bottom-left

p = pen pressure in raw tablet values, between [0-1023]

a = azimuth angle of the pen (yaw) between [0-3600] (10*degrees)

e = elevation angle of the pen (pitch) between [0-900] (10*degrees)

2) SVC 2004 Dataset

This dataset [10][11] was created using 20 genuine signatures and 20 forgeries. Each signature is represented as a sequence of points and is stored in a signature file with a .TXT extension. which has the following format:

<npoints>

<npoints> denotes the number of subsequent sampled trajectory points. Each line that follows represents a point. Each point contains a seven-tuple of recorded pen-tip coordinates (x, y, t, b, a, a, p):

x = x-position on tablet

y = y-position on tablet

t = timestamp

b = button status (i.e. button up or down)

a = azimuth angle of the pen

a = altitude angle of the pen

p = pen pressure in raw tablet values

The terms "altitude" and "elevation" are used interchangeably in the context of angles [12].

This dataset came in three parts: sample, Task 1 and Task 2. The signatures in Task 1 could not be used in conjunction with the SigComp2009 as it misses pressure values and azimuth elevation angles hence the signatures in the sample and Task 2 datasets was processed to remove the time stamp and button status features to match the features of the SigComp 2009 dataset. The datasets were first evened out (i.e. the number of genuine signatures is equal to the number of forgeries) to prevent overfitting. The datasets are merged in the code to

produce larger training and testing datasets. The testing datasets are prepared in the following ways:

• An even number of genuine and forgeries were removed from the SigComp2009 training and evaluation datasets (none are used for training or validation).

• An even number of genuine and forgeries were removed from the SVC 2004 datasets and is joined with the above-created dataset (some are used for training or validation).

• All signatures of the SVC 2004 are joined with the evaluation dataset created from the SigComp2009 dataset (some are used for training).

B. Data Normalization

The signatures were then pre-processed by normalizing the alignment and size. The reason for the alignment normalization is the fact that every person does not start their signature in the same pair of co-ordinates and normalizing this to equivalent conditions will improve the precision, learning, modelling and verification processes. Let a signature S with points xi with i = 0 to n-1 where n = total number of points used to represent the S. Let xi be represented by the tuple (x, y, p, a, e). The signature is aligned to start at point (0, 0, 0, 0, 0). Let Sa be the signature that underwent the alignment normalization process:

$$S_a = (x_i - x_0, y_i - y_0, p_i - p_0, a_i - a_0, e_i - e_0) \text{ for } i = 0, ..., n \quad (1)$$

The pressure and azimuth and elevation angles also go under the process of alignment normalization to aid in the size normalization (i.e. to get their values between 0 and 1). Each person requires different amounts of area for their signature. For example, a person with the surname "Ramrathan" will probably take more space for their signature when compared to someone with the surname "Ngidi". For this reason, all signatures go under the process of size normalization. Let Ss be the signature that underwent the size normalization process:

$$S_s = \begin{pmatrix} \left(\frac{x_i - \max(x)}{\min(x) - \max(x)}\right), \\ \left(\frac{y_i - \max(y)}{\min(y) - \max(y)}\right), \\ \left(\frac{p_i - \max(p)}{\min(p) - \max(p)}\right), \\ \left(\frac{a_i - \max(a)}{\min(a) - \max(a)}\right), \\ \left(\frac{e_i - \max(e)}{\min(e) - \max(e)}\right) \end{pmatrix} \quad (2)$$

The following three figures illustrate the normalization process of each signature file in each dataset.

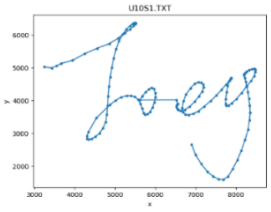


Figure 1: A signature before preprocessing

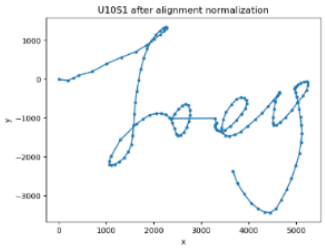


Figure 2: The same signature in Figure 1 after Alignment Normalization

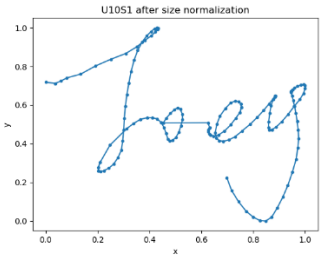


Figure 2: The same signature in Figure 1 after Alignment Normalization

C. CNN RNN Model Process

The model comprises of two parts: convolutional neural network (CNN) and recurrent neural network (RNN). The CNN is used as feature extractor and the RNN is used as the classifier. it was constructed using the Keras library in Python with a TensorFlow backend. This section includes a model breakdown and an algorithm of the system.

1) Model breakdown

The experiment passes the pre-processed data to a CNN. A CNN is defined as a multi-layer neural network with deep supervised learning architecture that has the ability to extract features for classification on its own [7]. The CNN consists of a feature extractor and a trainable classifier; for this paper, only the feature extractor is utilized. The feature extractor uses convolution filtering and downsampling operations to extract the features. The convolution filtering generates unique filters that can detect several features. The more convolutional layers the model contains, the more abstract and high-level the filters become which results in the ability to recognize objects. In this experiment, CNN was used in the form of CNN 1D. This is because the experiment deals with the signature being represented as text files that contain real values; not images of the signatures. A 1D CNN is very effective when expecting to derive interesting features from shorter (fixed length) segments of the overall data set and where the location of the feature within the segment is not of high relevance [13].

Normally, this is then passed to the trainable classifier to produce classification results but, for the purpose of this paper, it is rather passed to an RNN. An RNN has the ability to track the inputs it obtains. It was created to work with sequence prediction problems. It is usually is used for text and speech recognition (natural language processing), classification and

regression prediction and generative models. In this experiment, RNN was used in the form of a simple RNN.

Table 1. Layer summary of the model

Layer	Output Shape	Parameter/s
Convolutional 1D	(None, None, 5)	Kernel size = 20 Strides = 4 Activation = ReLU Kernel Initializer = he_normal Kernel Regulaizer = L2
Max Pooling 1D	(None, None, 100)	Pool size = 8
Convolutional 1D	(None, None, 50)	Kernel size = 10 Strides = 2 Activation = ReLU
Max Pooling 1D	(None, None, 50)	Pool size = 4
Convolutional 1D	(None, None, 25)	Kernel size = 5 Strides = 1 Activation = ReLU
Batch Normalization	(None, None, 25)	None
Dropout	(None, None, 25)	Drop = 0.5
Masking	(None, None, 25)	Mask value = 0.0
SimpleRNN	(None, 10)	Number of neurons = 10 Activation = ReLU Dropout = 0.3
Dense	(None, 4)	Number of neurons = 4 Activation = Sigmoid
Dense	(None, 2)	Number of neurons = 2 Activation = Softmax

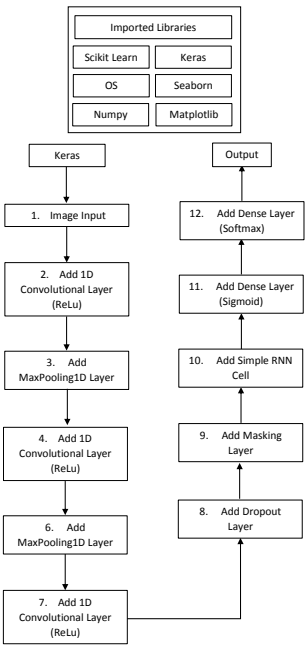


Figure 4: Model of the CNN RNN

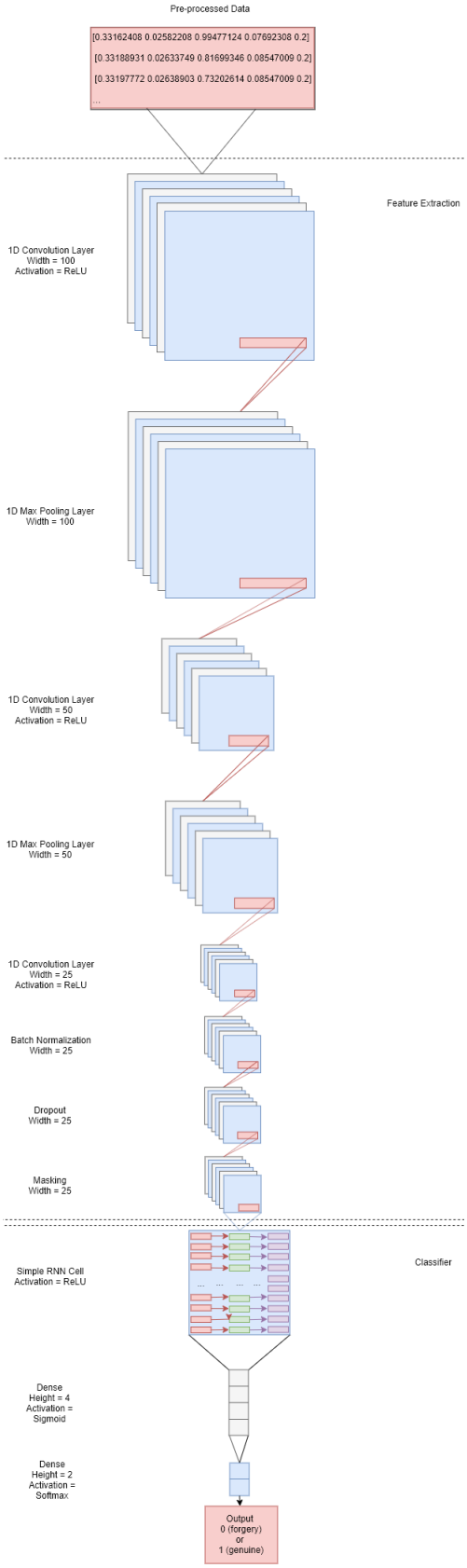


Figure 5: Graphical Representation of CNN RNN Model

a) Network architecture of CNN RNN

The architecture of the CNN RNN used in this experiment uses the elements as follows:

1D Convolutional Layers

This is the key layer of a CNN and performs a convolution on the input vector. The convolutional layers define filters or feature detectors with a height equal to the kernel size. However, signatures are very complex, hence, in each layer, n (which is different in each layer) filters are defined. Hence the output of this layer is a m x n where m is the number of points in a signature file - 20 + 1. With the defined kernel size and considering the length of the input matrix, each filter will contain m weights.

1D Max Pooling Layer

This layer is used to normalize the activations of the previous layer at each batch. This means that the layer applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1. The layer enables the use of higher learning rates, greatly accelerating the learning process. It also enabled the training of deep neural networks with sigmoid activations that were previously deemed too difficult to train due to the vanishing gradient problem. Based on its success, other normalization methods such as layer normalization and weight normalization have appeared and are also finding use within the field [14].

Dropout Layer

The dropout layer randomly assigns 0 weights to neurons in the model. This layer is utilized to make the model less sensitive towards smaller variations in the data.

Masking Layer

This layer is used to mask out the padded sequences so that the model doesn't use the padding as features and only predicts on the correct length of the sample.

Simple RNN Layer

This is a fully connected RNN where the output is to be fed back to the input.

Dense Layer

The dense layers are used to reduce the height of the vector to 4 then 2, using matrix multiplication, as the system must be able to output or classify if a signature is genuine or forged.

2) Algorithm of experiment

Algorithm: Online Signature Verification using Deep Descriptors

```
Input 3 paths containing folders containing training, validation and testing datasets
for each file in training, validation and testing path:
    read data from file
    normalize data using alignment and size pre-processing
    store training, validation and testing dataset
end for
create model
fit model using training and validation datasets
save model
evaluate model using testing dataset
Output measures of error of model from evaluation
```

IV. Results and Discussion

The model was trained using samples from the SVC 2004 and SigComp2009. It achieved 95.04% training accuracy. However, it achieved low validation results, due to the low number of samples used.

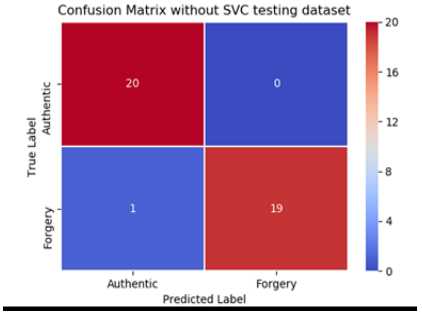
The experiment was carried on a system with GPU, which was utilized. This helped the program run smoothly and quickly training the network with an average of 12 seconds per epoch. As stated earlier, three testing datasets were created, one without using the SVC 2004 dataset, one that uses some of the SVC dataset and the last one that uses the complete SVC dataset.

From these test datasets the following performance measures: testing accuracy, False acceptance rate (FAR), False Rejection Rate (FRR), Equal Error Rate (EER) and Receiver Operating Characteristics Area Under Cover (ROC AUC).

Table 1. Results from the evaluation

	Testing Accuracy	FAR	FRR	EER	ROC AUC
Without SVC	97.50%	5%	0%	5%	0.9975
With cut SVC	90%	11.53%	8.57%	8.57%	0.9461
With full SVC	90.65%	15.43%	3.26%	5.22%	0.9689

Confusion matrices and the ROC curves for each testing dataset were also created.



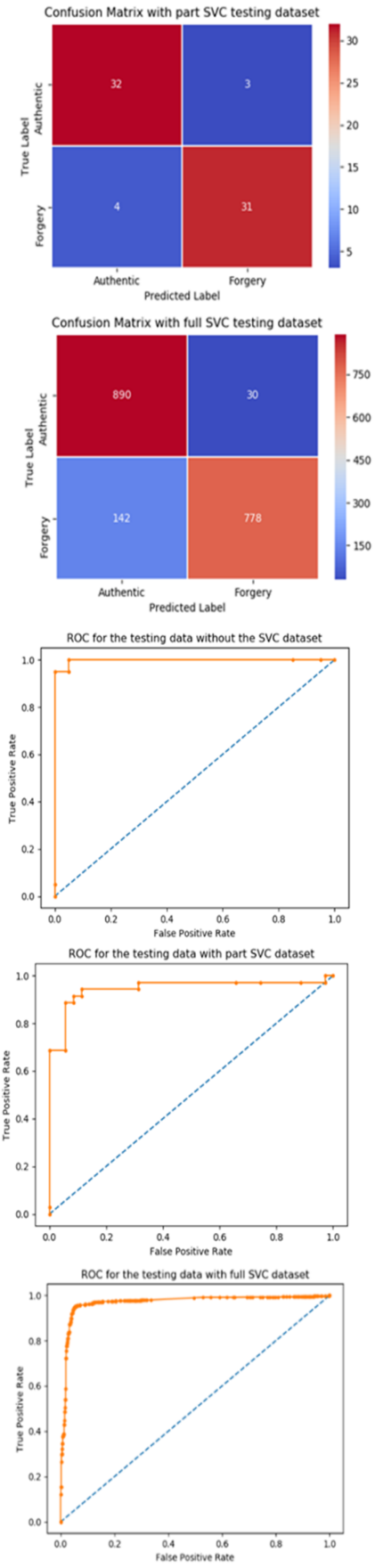


Fig 6: Graphs illustrating the results from the evaluation

V. CONCLUSION

This paper researched the use of artificial neural networks in the form of a CNN as a feature extractor and an RNN as the classifier in online signature verification. The results of this experiment indicate that this model is highly successful in its task. Due to 3 large convolutional layers, the model has the capability to build customized filters that are highly abstract leading the model to verify the signature as a whole. CNNs learns by generating these customized filters which are modified slowly to recognize features. This gives the model the ability to be independent of prior knowledge. The model will also not require pre-created filters which reduces the amount of computational time for the construction of the model. RNNs are used when a model requires context to provide an output based on the input. In an RNN model, all the inputs are related to each other hence signature verification works well using RNN model.

REFERENCES

[1] University, Michigan State. Signature Verification. Michigan : s.n.

[2] Artificial Neural Network Based Signature Recognition & Verification. Shiwani Sthapak, Minal Khopade, Chetana Kashid. 8, 2013, International Journal of Emerging Technology and Advanced Engineering, Vol. III, pp. 191-197.

[3] Automatic Signature Verification: The State of the Art. Donato Impedovo, Giuseppe Pirlo. 5, 2008, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Vol. 38, pp. 609-626.

[4] Online Signature Recognition Using Neural Network. P., Babita. 4, 2015, Journal of Electrical & Electronic Systems, Vol. 3.

[5] Signature Verification Using a Convolutional Neural Network. B. Cozzens, R. Huang, M. Jay, K. Khembunjong, S. Paliskara, F. Zhan, M. Zhang and S. Tayeb.

[6] Online Signature Verification using Recurrent Neural Network and Length-normalized Path Signature Descriptor. Songxuan Lai, Lianwen Jin, Weixin Yang. 2017.

[7] Forged Signature Distinction Using Convolutional Neural Network for Feature Extraction. Seungsoo Nam, Hosung Park, Changho Seo and Daeseon Choi. 2018, Applied Sciences.

[8] The ICDAR 2009 Signature Verification Competition. V.L. Blankers, C.E. van den Heuvel, K.Y. Franke, L.G. Vuurpijl.

[9] V.L. Blankers, C.E. van den Heuvel, K. Franke, L. Vuurpijl. The ICDAR 2009 Signature Verification Competition. [Online] 2009. http://www.iapr-tc11.org/mediawiki/index.php/ICDAR_2009_Signature_Verification_Competition_(SigComp2009).

[10] Dit-Yan Yeung, Hong Chang, Yimin Xiong, Susan George, Ramanujan Kashi, Takashi Matsumoto, Gerhard Rigoll. SVC2004: First International Signature Verification Competition. [Online] 2004. https://www.cse.ust.hk/svc2004/download.html.

[11] SVC2004: First International Signature Verification Competition. Dit-Yan Yeung, Hong Chang, Yimin Xiong, Susan George, Ramanujan Kashi, Takashi Matsumoto, Gerhard Rigoll. 2004, Proceedings of the International Conference on Biometric Authentication (ICBA).

[12] Fabio D. Vescovi, Charles Owen. Copernicus. [Online] 2019. https://spacedata.copernicus.eu/documents/12833/14549/CQC_TechnicalNote_09.

[13] Ackermann, Nils. Medium: Good Audience. [Online] September 4, 2018. https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf.

[14] Kurita, Keita. Machine Learning Explained. [Online] January 10, 2018. https://mlexplained.com/2018/01/10/an-intuitive-explanation-of-why-batch-normalization-really-works-normalization-in-deep-learning-part-1