INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

CNN with Batch Normalization Adjustment for Offline Hand-written Signature Genuine Verification

Wifda Muna Fatihia [a], Arna Fariza [a,*], Tita Karlita [a]

[a] *Informatics and Computer Engineering Department, Politeknik Elektronika Negeri Surabaya, Surabaya, 60111, Indonesia*
*Corresponding author: *arna@pens.ac.id*

***Abstract*—Signature genuine verifications of offline hand-written signatures are critical for preventing forgery and fraud. With the growth of protecting personal identity and preventing fraud, the demand for an automatic system for signature verification is high. The signature verification system is then studied by many researchers using various methods, especially deep learning-based methods. Hence, deep learning has a problem. Deep learning requires much training time for the data to obtain the best model accuracy result. Therefore, this paper proposed a CNN Batch Normalization, the CNN architectural adaptation model with a normalization batch number added, to obtain a CNN model optimization with high accuracy and less training time for offline hand-written signature verification. We compare CNN with our proposed model in the experiments. The research method in this study is data collection, pre-processing, and testing using our private signature dataset (collected by capturing signature images using a smartphone), which becomes the difficulties of our study because of the different lighting, media, and pen used to sign. Experiment results show that our model ranks first, with a training accuracy of 88.89%, an accuracy validation of 75.93%, and a testing accuracy of 84.84%—also, the result of 2638.63 s for the training time consumed with CPU usage. The model evaluation results show that our model has a smaller EER value; 2.583, with FAR = 0.333 and FRR = 4.833. Although the results of our proposed model are better than basic CNN, it is still low and overfitted. It has to be enhanced by better pre-processing steps using another augmentation method required to improve dataset quality.**

***Keywords*— Hand-written signature; signature verification; batch normalization; convolutional neural network.**

Manuscript received 5 Dec. 2022; revised 22 Dec. 2022; accepted 14 Jan. 2023. Date of publication 31 Mar. 2022.

I. INTRODUCTION

Signatures come from the Latin '*signa*re'. Signature means sign or initial, a unique form of hand-writing that contains special characters, and additional forms are often used as proof of authentication or verification of someone's identity. A signature is a symbol of the name as it is commonly used, including an initial, stamp, signature stamp, or other signs as the representative of a signature or an electronic signature as intended in the information and electronic transactions. Signatures are a type of biometric authentication used to verify a person's identity [1]. Since ancient times, signature has been the government's most widely used and accepted individual biometric mark [2]. Hand-written signatures are widely used for verifying contract letters, checks, bank applications, and statement documents [3]–[5]. Everybody has a unique signature pattern. Even an expert cannot duplicate their own signature precisely. This is referred to as a natural diversity [6]. However, if tolerance for natural diversity in agencies that require important valid documents happens, there will be various cases of forgery when signatures are verified manually. It can result in fraud, causing financial and emotional loss to the victim.

Human error in manual signature verification is the reason for the increasing number of signature forgery cases. Signature forgery is the fraudulent copying of another person's signature for a particular purpose. Due to every person's different nature and personality, every signature has a different pattern and shape, but human senses have limitations in the verification process if the patterns compared are very similar [7], [8]. This condition can result in a miss-verified signature strikingly similar to the genuine one. Signature forgery simulation levels are as follows:

- Skilled forgery when the forger knows the target's name and practiced imitating the target's genuine signature.
- Simple forgery when the forger knows the target's name but not the signature.
- Random forgery is when the forger does not have any information about the target's signature, so instead of it, he uses it [9].

The demand for protecting individual identity is growing, so an automatic signature verification system design is needed [10], [11]. It also helps previous manual verification by human sense. This system's purpose is to verify a person's identity based on their signature pattern and characteristics, and it also important for preventing signature forgery and fraud cases. Depending on the input format, there are two kinds of signature verification: online and offline signature verification [3], [10], [12]. Due to the necessity to tackle various problems, such as fraud detection, personal identity, banking transactions, and courtroom proceedings, many researchers study offline and online signature verification systems [13]. The online-based signatures are called "dynamic signatures."

In the online signature verification process, dynamic features like the movement of the pen used in the signature process—the location, speed, and pressure of the pen—will be recorded and identified [14]. This type of verification is used on real-time access media, such as credit card transactions. The benefit of online signature verification is the information gained about dynamic features acquired when someone signs the document, even though it needs a unique device to capture the features. Offline-based signature verification is static verification done by scanning physical documents for hand-written signatures after the signing is done [15]. The scan results are then verified using specific methods [16]. As no dynamic features could be acquired by offline signature verification, the problem becomes more complicated. Also, the same person's signature could have a bit of a different pattern because of sickness, psychology, age, and media condition [17]. It creates the difficulties and challenges of offline signature verification. In offline signature verification, the signer doesn't need a special device or to be present during the verification phase because the signature has already been stored in the database. This is the advantage of offline signature verification and makes it suitable for authorizing bank checks, documents, and other signed forms [18].

Since a few decades ago, original signature verification has been a difficult research subject in the field of computer vision (CV) because of extremely significant applications like banks and others [2]. The signature image verification process separates the image into two classes: genuine and forged signature. According to the findings of the research literature review, previous researchers used a variety of approaches to verify offline signature images, including machine learning and deep learning approaches [19]. Deep learning with deeper neural networks proves to be more robust in performing image classification than machine learning methods. Convolutional Neural Networks inspired by human neural networks are the most successful deep learning models in computer vision [20]. In recent years, Convolutional Neural Networks (CNN) have thoroughly outplayed the machine vision field [21] and reliability in the image classification field [22]–[24].

Loussaief and Abdelkrim [25] experimented with CNN's ability to feature image extraction for classification using the AlexNet architecture for the Caltech101 dataset. He compared the image extraction feature using AlexNet with the Bag of Features method, which was inspired by Bag of Words. The results show that the classification performance after using deep learning-based CNN increases by 20%, so CNN is very effective in obtaining the image information needed in data classification. Rabbi et al [26] compared several deep learning signature identification methods, including CNN, CNN with data augmentation, MLP (Multilayer Perceptron), and Single Layer Perceptron (SLP). The dataset is derived from a private dataset of 30 individuals, each with 24 authentic and 24 forged signatures. The results show that accuracy with the CNN algorithm is 82.75%, accuracy with SLP is 39.91%, and accuracy with MLP is 63.57%. The deeper architectures of deep learning, the more training examples are needed when trained from scratch [27]. Vorugunti, Gorthi, and Pulabaigari [28] states that the current challenge for hand-written signature verification is to make a small number of signature training datasets that already represent the user to perform signature image training based on intra- or inter-individual variability. Another problem with using deep learning for signature verification is the computational training cost or time consumed to train the deep learning model. The computational cost of training these complex networks might need large amounts of data to reach the precision needed to give the necessary accuracy. Contributing factors that make the situation worse are: finding an "optimal" network architecture that has the potential to perform as desired, choosing the best set of hyper-parameters for the network, estimating the data needed for training, and figuring out whether training can be done within the necessary cost envelope [29].

We also faced a degradation problem of using deep learning while the deeper networks started converging. As the network is deeper, the accuracy gets saturated and degrades quickly [30]. Several studies have used Batch Normalization (BN) to overcome this problem. Another benefit of Batch Normalization is its capacity to regularize, which has a better convergence speed. Consequently, BN will add some noise with a similar purpose to drop out. However, the trained model's generalization ability will improve [31]. Wang et al. [32] show that enhanced LSTM with Batch Normalization gives faster convergence than LSTM variants and gains higher classification accuracy of learning tasks on a sequence. Batch Normalization significantly outperforms the original model, achieving the same accuracy with 14 times fewer training steps [33]. They improved upon the best-published result on ImageNet classification using an ensemble of Batch-Normalized networks, achieving 4.9% top-5 validation error (and 4.8% test error). Wu and He [34] stated that Batch Normalization is the milestone of deep learning development, which various networks are enabled to train. This is the cause of the batch statistics' stochastic uncertainty functions as a regularization term that can improve in generalization.

This paper proposed a CNN with Batch Normalization Adjustment for Offline Hand-written Signature Genuine Verification. It is the adaptation of CNN architecture with Batch Normalization to obtain faster training time and higher testing accuracy. This study collected one hundred individuals, each providing three original signatures. Three signatures collected for each person aim to increase the inter-individual variability training dataset. Then to increase the data training intra-individual variability, each original signature was forged by three others. It is hoped that this can enrich the characteristics of genuine and forged signatures. The collected signature images will be subjected to pre-processing and augmentation. Image training data is also related to several conditions, such as lighting when capturing signature images, the pens used, and different paper media.

Therefore, to overcome the problems mentioned above, we have the following contributions:

- To propose a novel CNN-based architecture with the addition of Batch Normalization.
- The proposed model is aimed to obtain the optimal CNN model for offline hand-written signature verification with high testing accuracy levels and faster training time so that it can be implemented in a mobile application for future work.
- To know model enhancement using Batch Normalization, we compare CNN and CNN-Batch Normalization models using the comparison of EER network performance evaluation of each model.

The arrangement of this paper is as follows: material and method of our proposed method in section II. Section III discusses the results of our model experiments compared with the other model. The last section, section IV, is the conclusion of our study.

II. Material and Method

Figure 1 depicts the study's procedure. There are five distinct phases, and the phases are as follows: data collection, data pre-processing and augmentation, architecture or model generation, model validation testing, and model evaluation.

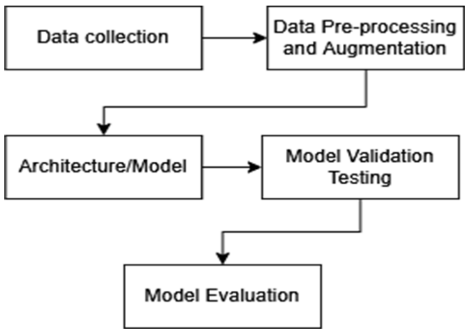

Fig. 1 The Procedure of the Study

A. Data Collection

Our private dataset was used in this study. It is signature image data from 101 people, with three genuine and three forged signatures. The signature images are created by writing a signature on paper and photographing it with a cell phone camera. We obtain the dataset both directly and through file-sharing services. Meanwhile, we used skilled forgery to create the forged signatures. Figure 2 depicts genuine signature images from our private dataset.

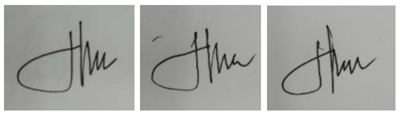

Fig. 2 Genuine Signature Dataset Examples

It is genuinely signed three times by the same person, and we can see that each picture has inter-individual variability. Whereas Figure 3 depicts forged signature images forged by three different people.

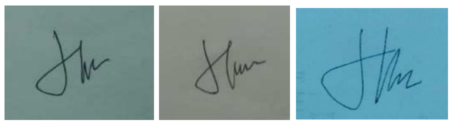

Fig. 3 Forged Signature Dataset Examples

Figures 2 and 3 are raw dataset images before the data pre-processing step. The characteristics of the initial image contained in the dataset vary, ranging from the use of pen, paper, background or image background, size, and color, so it is necessary to pre-process the dataset to reduce the bias factor of the dataset. It is one of the research's difficulties. The procedure was divided into two parts: training and testing data, and the generated model was then tested against testing data.

B. Data Pre-processing and Augmentation

Pre-processing data is a step that improves image quality and reduces the effect of the background image, grey variation, size, and signature location on verification performance [15], [23], [35]. Gideon et al. [11] propose using CNN to verify hand-written signature forgery using several pre-processing steps. He converts RGB images to grayscale to simplify digital image processing. He then removes noise, converts the image to grayscale, and resizes it to a standard resolution that matches the network input.

The main features for identifying signatures can be obtained by pre-processing the training data. During this process, we resize the signature image to match the initial convolution layer size. Furthermore, we convert RGB color photos to greyscale images. After that, we normalize the images so the network can efficiently process them. The image is reduced to a single channel with '0' and '1' values. The '0' value represents the background image value, whereas the '1' value represents image feature information [36], [37].

We use the data augmentation process during the training stage to add sample signature variants to the network during the training process. Data augmentation is modifying image data so that the computer recognizes it as a different image [6], [26], [38]. Otherwise, it will be perceived as the same image object. The goal is to improve generalization by enriching the training dataset. Augmentation can increase the amount of data the model receives, but it must be balanced with good tuning to produce high performance [39]. This study employs data augmentation through rotation. It can improve training accuracy by enhancing the existing features or characteristics of the signature image. The parameters used in the pre-processing and augmentation of signature images are shown in Table I.

TABLE I
Pre-processing and Augmentation Parameter

Parameter	Value	Description
Resize	100x100	A process to change the image size into a certain pixel size to match the convolution layer.
Normalization	std = 0.1; mean = 0.5	Normalization process is used to change the image canal.
Augmentation	Rotation (-20, 20)	Random rotation of the training data in the range of -20 to 20 degree.

First, the RGB image dataset is converted into greyscale. Second, to sharpen the signature pattern, the image is subjected to the random auto-contrast function from PyTorch.

Third, the image size is changed to be smaller to 100x100 pixels so that it is faster in convolution and dataset training. Next, the reduced image will be rotated (augmented) and normalized. The result examples of data pre-processing are shown in Figure 4.

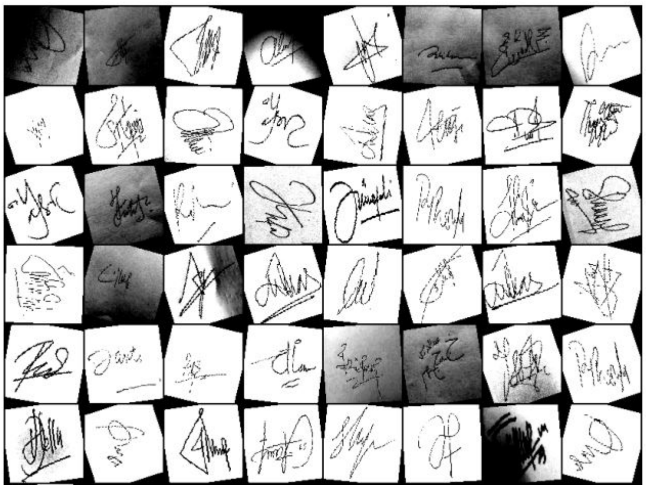
Fig. 4 Signature Dataset After Pre-processing and Augmentation

C. Model

This research compares basic CNN architecture with the CNN Batch Normalization model (our proposed model).

1) CNN: One of the deep learning models, Convolutional Neural Network (CNN), is used to classify signature images. CNN extracts feature directly from a raw image dataset. Related features are not pre-trained; they will be learned during training. This is the most accurate learning model for computer vision tasks such as object identification, categorization, and recognition. CNN consists of three block layers: convolutional, pooling, and fully connected [7], [18], [40].

- Convolutional layers are made up of multiple convolution operations. It is the process of repeatedly applying functions to the output of other functions, and Convolution is applied to the input image to extract features.
- Subsampling or pooling layers are used to reduce image size. This layer employs max pooling with a specific kernel filter size, dividing the previous convolution layer's output into several small grids and rearranging the reduced image matrix. The goal is to reduce bias as much as possible. At a later stage, only the most important features will be processed.
- The final layer, fully connected, aims to transform image dimensions so that they can be classified linearly. The previous layer's nodes are flattened so that neurons can be connected.

TABLE II
CNN ARCHITECTURE SUMMARY

Layer	Size	Extra Parameters
Input	100 x 100 x 1	none
Convolution 1	4 x 4 x 101	stride=1, pad=1
ReLU	4 x 4 x 101	none
Convolution 2	8 x 8 x 102	stride=1, pad=1
ReLU	8 x 8 x 102	none
Max-Pooling	8 x 8 x 51	stride=1
Convolution 3	16 x 16 x 52	stride=1, pad=1
ReLU 3	16 x 16 x 52	none
MaxPool	16 x 16 x 26	stride=1
MaxPool	16 x 16 x 13	stride=1
Sigmoid	1	none

Table II shows the architecture summary of the CNN model. The input size is 100x100 after pre-processing into greyscale format and resizing. The total parameters and trainable parameters used in this architecture are 542,086.

2) CNN Batch Normalization: CNN Batch Normalization is a CNN architectural adaptation model with a normalization batch number added on top. Batch normalization is a training process stabilization function. It reduces the number of epochs, which speeds up the training process. It also helps to reduce generalization errors [41].

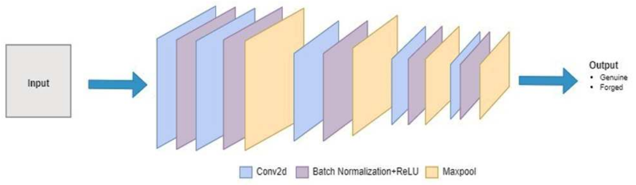

Fig. 5 CNN Batch Normalization

Figure 5 shows CNN Batch Normalization architecture (our purposed model). The addition of batch normalization in this model is intended to overcome the high loss during training, allowing it to reach convergence more quickly. A batch normalization layer can be applied to convolutional neural networks directly between the convolutional layer and the activation function layer, such as ReLU (Rectified Linear Unit), a batch normalization layer can be applied to convolutional neural networks [42]. Because CNN is the basic architecture we used, this model is built the same way as the CNN model, with a convolution layer (Conv2d), batch normalization, the ReLU activation function, and a max pooling layer (MaxPool2d).

By subtracting the batch mean and dividing it by the batch standard deviation, batch normalization ensures that the batch has a distribution of 0 and 1, respectively, for the mean and standard deviation. It makes intuitive sense that this output normalizes the inputs of a layer by statistically balancing the batch samples and lowering the internal covariate shift throughout the architecture, which may otherwise block deeper layers from learning a given task [33].

As in Avola et al. [27], the BN is calculated using the following equation for a layer activation function output of x for a mini batch β:

$$BN_{\gamma,\beta}(x_i) = \gamma\hat{x}_i + \beta \quad (1)$$

where $\hat{x}_i$ is the output of the normalized layer, and γ and β are trainable parameters used to learn the proper normalization for a certain layer as follows:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (2)$$

where μ_B and σ_B is mini-batch mean and standard deviation, respectively.

TABLE IIII
CNN BATCH NORMALIZATION ARCHITECTURE SUMMARY

Layer	Size	Extra Parameters
Input	100 x 100 x 1	none
Convolution 1	4 x 4 x 101	stride=1, pad=1
BatchNorm2d-2	4 x 4 x 101	none
ReLU	4 x 4 x 101	none
Convolution 2	8 x 8 x 102	stride=1, pad=1
BatchNorm	8 x 8 x 102	none
ReLU	8 x 8 x 102	none
Max-Pooling	8 x 8 x 51	stride=1
Convolution 3	16 x 16 x 52	stride=1, pad=1
BatchNorm	16 x 16 x 52	32
ReLU 3	16 x 16 x 52	none
MaxPool	16 x 16 x 26	stride=1
MaxPool	16 x 16 x 13	stride=1
Sigmoid	1	none

Table III shows the architecture summary of CNN model. The input size is 100x100 after pre-processing into greyscale format and resizing. The total parameters and trainable parameters used in this architecture is 542,142.

D. Model Evaluation

During the testing stage, we will use data testing to evaluate the model formed in the previous training stage. The testing dataset has been pre-processed to convert the signature picture to a binary image, decrease the image's bias factor, and shrink the image's pixel size. During the training process, the network is through a learning process and extract features from 90 individuals. The testing dataset is distinct from the training dataset used in the testing process. In this case, the signature image of 11 people out of a total of 101 people was used.

The Confusion Matrix results can be used to assess the model's accuracy. A confusion matrix provides a value for comparing the image classification result to the model performance. EER (Equal Error Rate) measures network accuracy by balancing the False Acceptance Rate (FAR) and False Rejection Rate (FRR) [19], [43]. A confusion matrix is a matrix that provides a comparison value for the classification result within the context of the model's prediction performance. The confusion matrix is depicted in Figure 6.

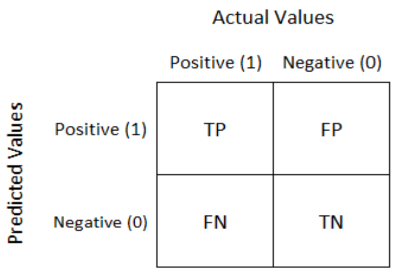

Fig. 6 Confusion Matrix

The components of the confusion matrix in Figure 6 are as follows:

- True Positive (TP) is the number of positive data predicted correctly. In this study, the genuine signature is part of the positive data.
- True Negative (TN) is the number of negative data predicted correctly. In this study, forged signatures are part of negative data.
- False Positive (FP) is the number of positive data predicted incorrectly. In this study, if data should be a genuine signature but is predicted to be forged, it will be included in the FP count.
- False Negative (FN) is the number of negative data predicted incorrectly. In this study, if a forged signature is detected as a genuine signature, it will be included in the FN calculation.

The result is then reprocessed into a performance matrix to calculate the accuracy, precision, and recall, or sensitivity value, to measure network performance.

1) Accuracy is the percentage of time that a model correctly predicts the data. Accuracy is determined by comparing the correct prediction to the sum of all data. The equation is the formula for calculating the accuracy value.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

2) Precision is calculated by comparing the total number of correct positive predictions to the total number of positively predicted data. The precision value is calculated using the formula in equation (4) by comparing the total number of correct positive predictions to the total number of positively predicted data.

$$precission = \frac{TP}{TP+FP} \tag{4}$$

3) Recall or sensitivity will show the percentage of the model's success in finding information. The formula for calculating the precision value is in equation (5).

$$recall = \frac{TP}{TP+FN} \tag{5}$$

In the field of biometrics, network performance can be evaluated by calculating the EER (Equal Error Rate) value, which compares the FAR (False Acceptance Rate) with the FRR (False Rejection Rate). It is represented as the value of the ROC (Region of Convergence) curve crossed by the line traveling through the cartesian coordinates (1,0) and (0,1). (0,1). The more accurate a biometric system is, the lower the EER value.

- FAR (False Acceptance Rate) determines the probability of a forged signature being mistaken for a genuine signature. The value is calculated by dividing the number of false acceptance values by the number of identification attempts, as shown in equation (6).

$$FAR = \frac{FN}{TP+FN} \tag{6}$$

- The FRR (False Rejection Rate) assesses the probability that a genuine signature may be incorrectly identified as forged. As stated in the equation, the value is determined by dividing the number of false rejections by the number of identification attempts.

$$FRR = \frac{FP}{TP+FN} \tag{7}$$

In this instance, the EER value will indicate network precision, so a balance between FAR and FRR values is necessary. The better the network, the higher the accuracy and the lower the FAR and FRR values. These values will be used to evaluate the three models tested.

III. Results and Discussion

The test is carried out by training the dataset and testing the same classifier only once. For the system to test data, knowledge of feature extraction from genuine and forged signature images is expected.

A. Learning Stage

The type and amount of input data influenced the architecture's selection. Several learning parameters can be adjusted to optimize the learning process during the learning process. The same learning parameters are applied to each model. Table IV shows the learning parameters used in this experiment.

TABLE IV
LEARNING PARAMETER

Parameter	Value
learning rate	0.00001
image size	100x100
optimizer	Adam
epoch	150

This study's learning stage is using the CPU of intel core i5, with 12 GB RAM installed. CNN Batch Normalization architecture is consuming 2638.63 s for the training time. Meanwhile, CNN architecture consumes 2553.83 s for 150 epochs, using the same learning parameter as our proposed model.

B. Model Validation Testing

CNN and CNN Batch Normalization models were used, and they trained over 150 epochs. The test image is 100x100 pixels, has a grayscale color channel, and is drawn randomly from the testing data. The test results will be displayed in a classification report that includes recall, precision, and the f1-score, as well as a confusion matrix with false positive, false negative, true positive, and true negative components. They will be used to calculate the false rejection rate, false acceptance rate, and equal error rate resulting from model performance evaluation.

The test phase puts the model to the test with 66 signature images gathered from 3 signatures per individual that the network has not recognized. Figures 7–9 show the confusion matrix from CNN and CNN Batch Normalization models test results.

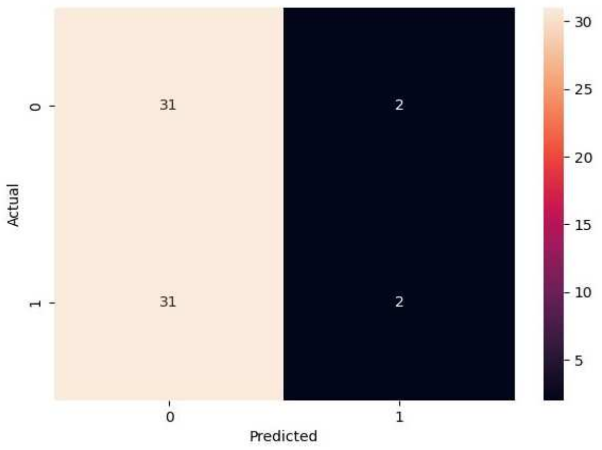

Fig. 7 Confusion Matrix of CNN

Figure 7 shows the classification results from the CNN Batch Normalization model. There are 31 data for TN, 2 for FN, 31 for FP, and 2 for TP.

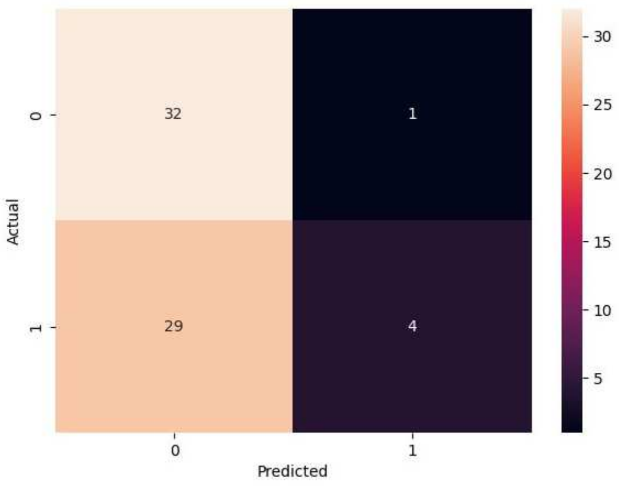

Fig. 8 Confusion Matrix of CNN Batch Normalization

Figure 8 shows the classification results from CNN model. There are 32 data for TN, 1 for FN, 29 for FP, and 4 for TP.

This is how the Confusion Matrix is described:

- Row 0 column 0: True Negative/TN (Sum of genuine signatures predicted to be forged).
- Row 0 column 1: False Negative/FN (Sum of forged signatures predicted to be genuine).
- Row 1 column 0: False Positive/FP (Sum of forged signatures predicted to be false).
- Row 1 column 1: True Positive/TP (Sum of genuine and predicted genuine signatures).

TABLE V
MATRIX EVALUATION OF CNN

	Precision	Recall	F1-Score	Support
0	0.52	0.97	0.68	33
1	0.80	0.12	0.21	33
accuracy			0.55	66
Macro avg	0.66	0.55	0.45	66

TABLE VI
MATRIX EVALUATION OF CNN BATCH NORMALIZATION

	Precision	Recall	F1-Score	Support
0	0.38	0.18	0.24	33
1	0.46	0.70	0.55	33
accuracy			0.44	66
Macro avg	0.42	0.44	0.40	66

C. Model Comparison Result

Table V displays the experimental results of CNN and CNN Batch Normalization models with 150 training epochs. The values in the table represent the final values obtained when the epoch reaches 150. The models with the highest testing accuracy values in Table VII are CNN Batch Normalization and CNN in that order.

TABLE VII
EXPERIMENT RESULT

Model	Training		Validation		Testing
	Accuracy (%)	Loss (%)	Accuracy (%)	Loss (%)	Accuracy (%)
CNN	81.28	0.483	**77.78**	0.639	78.78
CNN Batch Normalization	**88.89**	**0.291**	75.93	**0.499**	**84.84**

We evaluate network performance by calculating FAR, FRR, and EER using the confusion matrix components (TP, TN, FP, and FN). Table VI displays the model performance evaluation results for each model with 150 training epochs using FAR, FRR, and EER. Equations (6) and (7) are used to calculate the FAR and FRR of each model. The EER is the average of each model's FAR and FRR.

TABLE VIII
MODEL PERFORMANCE EVALUATION USING FAR, FRR, AND EER

Evaluation	CNN	CNN Batch Normalization
FAR	0.500	0.333
FRR	7.750	4.833
EER	4.125	**2.583**

The smaller and closer to zero, the FAR, FRR, and EER values are, the better the network or model performance. According to Table VIII, the CNN Batch Normalization model has the lowest FAR and FRR values. As a result, the CNN Batch Normalization model obtains the smallest EER value.

IV. CONCLUSIONS

When the training accuracy and validation accuracy results are compared to the results of the experiment for each model for 150 epochs, most models are still overfitting, despite having a high percentage of testing accuracy. This condition can occur because the data used during training is less varied, and the dataset is too small for complex models (CNN is data-hungry). We still have signature dataset images that were not affected by the normalization process as a result of the pre-processing results. The background and signature features are needed to be differed using any other pre-processing method to enrich the feature extraction.

However, our model has a faster training time than CNN without Batch Normalization added. With a training loss of 0.291%, training accuracy of 88.89%, a validation loss of 0.499%, an accuracy validation of 75.93%, and a testing accuracy of 84.84%, our model, CNN Batch Normalization, took first place in a testing scenario. In future work, we propose a model adjustment to achieve better model performance with fewer datasets. Additionally, better pre-processing steps using another augmentation method are required to improve dataset quality and prevent overfitting. The better results could be developed into a mobile signature verification application, allowing us to apply the benefits to a real-world signature verification problem.

REFERENCES

[1] C. Wencheng, G. Xiaopeng, S. Hong, and Z. Limin, "Offline Chinese signature verification based on AlexNet," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2018, vol. 219, pp. 33–37, doi: 10.1007/978-3-319-73317-3_5.

[2] A. Sanjoy, P. Soumya, R. Nayak, and T. Hanne, "Off-line signature verification using elementary combinations of directional codes from boundary pixels," *Neural Comput. Appl.*, vol. 6, 2021, doi: 10.1007/s00521-021-05854-6.

[3] F. E. Batool *et al.*, "Offline signature verification system: a novel technique of fusion of GLCM and geometric features using SVM," *Multimed. Tools Appl.*, 2020, doi: 10.1007/s11042-020-08851-4.

[4] G. Suhas, S. Chiranjeevi, S. S. Mokshagundam, and S. Suraj, "SIFR-signature fraud recognition," in *2018 International Conference on Networking, Embedded and Wireless Systems, ICNEWS 2018 - Proceedings*, 2018, pp. 1–6, doi: 10.1109/ICNEWS.2018.8903995.

[5] S. N. Srihari, H. Srinivasan, S. Chen, and M. J. Beal, "Machine learning for signature verification," *Stud. Comput. Intell.*, vol. 90, pp. 387–408, 2008, doi: 10.1007/978-3-540-76280-5_15.

[6] M. M. Yapıcı, A. Tekerek, and N. Topaloğlu, "Deep learning-based data augmentation method and signature verification system for offline hand-written signature," *Pattern Anal. Appl.*, vol. 24, no. 1, pp. 165–179, 2021, doi: 10.1007/s10044-020-00912-6.

[7] M. M. Yapici, A. Tekerek, and N. Topaloglu, "Convolutional Neural Network Based Offline Signature Verification Application," *Int. Congr. Big Data, Deep Learn. Fight. Cyber Terror. IBIGDELFT 2018 - Proc.*, no. December, pp. 30–34, 2019, doi: 10.1109/IBIGDELFT.2018.8625290.

[8] A. Dutta, U. Pal, and J. Llados, "Compact correlated features for writer independent signature verification," in *Proceedings - International Conference on Pattern Recognition*, 2016, pp. 3422–3427, doi: 10.1109/ICPR.2016.7900163.

[9] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Meta-learning for fast classifier adaptation to new users of Signature Verification systems," *IEEE Trans. Inf. Forensics Secur.*, vol. PP, no. c, p. 1, 2019, doi: 10.1109/TIFS.2019.2949425.

[10] P. Singh, P. Verma, and N. Singh, "Offline Signature Verification : An Application of GLCM Features in Machine Learning," *Ann. Data Sci.*, no. 0123456789, 2021, doi: 10.1007/s40745-021-00343-y.

[11] S. Jerome Gideon, A. Kandulna, A. A. Kujur, A. Diana, and K. Raimond, "Hand-written Signature Forgery Detection using Convolutional Neural Networks," *Procedia Comput. Sci.*, vol. 143, pp. 978–987, 2018, doi: 10.1016/j.procs.2018.10.336.

[12] H. Bunke, J. Csirik, Z. Gingl, and E. Griechisch, "Online signature verification method based on the acceleration signals of hand-writing samples," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 7042 LNCS, pp. 499–506, doi: 10.1007/978-3-642-25085-9_59.

[13] T. Younesian, S. Masoudnia, R. Hosseini, and B. N. Araabi, "Active Transfer Learning for Persian Offline Signature Verification," *4th Int. Conf. Pattern Recognit. Image Anal. IPRIA 2019*, pp. 234–239, 2019, doi: 10.1109/PRIA.2019.8786013.

[14] M. M. Hameed, R. Ahmad, M. L. M. Kiah, and G. Murtaza, "Machine Learning-based Offline Signature Verification Systems: A Systematic Review," *Signal Process. Image Commun.*, vol. 93, no. October 2020, p. 116139, 2021, doi: 10.1016/j.image.2021.116139.

[15] E. Guerra-Segura, A. Ortega-Pérez, and C. M. Travieso, "In-air signature verification system using Leap Motion," *Expert Syst. Appl.*, vol. 165, no. August 2020, 2021, doi: 10.1016/j.eswa.2020.113797.

[16] D. Impedovo and G. Pirlo, "Automatic signature verification: The state of the art," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 38, no. 5, pp. 609–635, 2008, doi: 10.1109/TSMCC.2008.923866.

[17] S. C. Satapathy, A. Govardhan, K. Srujan Raju, and J. K. Mandal, "Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1," in *Advances in Intelligent Systems and Computing*, 2015, vol. 337, pp. 337–338, doi: 10.1007/978-3-319-13728-5.

[18] C. Ishikawa, J. A. U. Marasigan, and M. V. C. Caya, "Cloud-based Signature Validation Using CNN Inception-ResNet Architecture," 2020, doi: 10.1109/HNICEM51456.2020.9400027.

[19] M. Hanmandlu, A. B. Sronothara, and S. Vasikarla, "Deep Learning based Offline Signature Verification," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2018*, 2018, pp. 732–737, doi: 10.1109/UEMCON.2018.8796678.

[20] S. J. Lee, T. Chen, L. Yu, and C. H. Lai, "Image Classification Based on the Boost Convolutional Neural Network," *IEEE Access*, vol. 6, no. c, pp. 12755–12768, 2018, doi: 10.1109/ACCESS.2018.2796722.

[21] M. Hussain, J. J. Bird, and D. R. Faria, "A study on CNN transfer learning for image classification," *Adv. Intell. Syst. Comput.*, vol. 840, pp. 191–202, 2019, doi: 10.1007/978-3-319-97982-3_16.

[22] L. Liu, L. Huang, F. Yin, and Y. Chen, "Offline signature verification using a region based deep metric learning network," *Pattern Recognit.*, vol. 118, p. 108009, 2021, doi: 10.1016/j.patcog.2021.108009.

[23] N. Sharma, V. Jain, and A. Mishra, "An Analysis of Convolutional Neural Networks for Image Classification," *Procedia Comput. Sci.*, vol. 132, no. Iccids, pp. 377–384, 2018, doi: 10.1016/j.procs.2018.05.198.

[24] S. Alkaabi, S. Yussof, S. Almulla, H. Al-Khateeb, and A. A. Alabdulsalam, "A novel architecture to verify offline hand-written

signatures using convolutional neural network," in *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2019*, 2019, pp. 0–3, doi: 10.1109/3ICT.2019.8910275.

[25] S. Loussaief and A. Abdelkrim, "Deep learning vs. bag of features in machine learning for image classification," *2018 Int. Conf. Adv. Syst. Electr. Technol. IC_ASET 2018*, pp. 6–10, 2018, doi: 10.1109/ASET.2018.8379825.

[26] M. T. F. Rabbi *et al.*, "Hand-written Signature Verification Using CNN with Data Augmentation," *J. Contents Comput.*, vol. 1, no. 1, pp. 25–37, 2019, doi: 10.9728/jcc.2019.12.1.1.25.

[27] D. Avola, M. J. Bigdello, L. Cinque, A. Fagioli, and M. R. Marini, "R-SigNet : Re duce d space writer-independent feature learning for offline writer-dependent signature verification," *Pattern Recognit. Lett.*, vol. 150, pp. 189–196, 2021, doi: 10.1016/j.patrec.2021.06.033.

[28] C. S. Vorugunti, R. K. S. Gorthi, and V. Pulabaigari, "Online signature verification by few-shot separable convolution based deep learning," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2019, pp. 1125–1130, doi: 10.1109/ICDAR.2019.00182.

[29] D. Justus, J. Brennan, S. Bonner, and A. S. McGough, "Predicting the Computational Cost of Deep Learning Models," *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*, pp. 3873–3882, 2019, doi: 10.1109/BigData.2018.8622396.

[30] Kaiming He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE*, 2015, [Online]. Available: http://image-net.org/challenges/LSVRC/2015/.

[31] H. Yong, J. Huang, D. Meng, X. Hua, and L. Zhang, "Momentum Batch Normalization for Deep Learning with Small Batch Size," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12357 LNCS, pp. 224–240, 2020, doi: 10.1007/978-3-030-58610-2_14.

[32] L. N. Wang, G. Zhong, S. Yan, J. Dong, and K. Huang, *Enhanced LSTM with batch normalization*, vol. 11953 LNCS. 2019.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 1, pp. 448–456, 2015.

[34] Y. Wu and K. He, *Group Normalization*, vol. 128, no. 3. Springer International Publishing, 2020.

[35] L. Liu, L. Huang, F. Yin, and Y. Chen, "Offline Signature Verification using a Region Based Deep Metric Learning Network," *Pattern Recognit.*, vol. 118, p. 108009, 2021, doi: 10.1016/j.patcog.2021.108009.

[36] D. Banerjee, B. Chatterjee, P. Bhowal, T. Bhattacharyya, S. Malakar, and R. Sarkar, "A new wrapper feature selection method for language-invariant offline signature verification," *Expert Syst. Appl.*, vol. 186, no. March, 2021, doi: 10.1016/j.eswa.2021.115756.

[37] S. Dey, A. Dutta, J. I. Toledo, S. K. Ghosh, J. Llados, and U. Pal, "SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification," *Elsevier*, no. 1, pp. 1–7, 2017, [Online]. Available: http://arxiv.org/abs/1707.02131.

[38] M. Hron and N. Obwegeser, "Scrum in Practice: An Overview of Scrum Adaptations," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2018-Janua, pp. 5445–5454, 2018, doi: 10.24251/hicss.2018.679.

[39] J. F. Vargas, M. A. Ferrer, C. M. Travieso, and J. B. Alonso, "Off-line signature verification based on grey level information using texture features," *Pattern Recognit.*, vol. 44, no. 2, pp. 375–385, 2011, doi: 10.1016/j.patcog.2010.07.028.

[40] S. M. A. Navid, S. H. Priya, N. H. Khandakar, Z. Ferdous, and A. B. Haque, "Signature Verification Using Convolutional Neural Network," *2019 IEEE Int. Conf. Robot. Autom. Artif. Internet-of-Things, RAAICON 2019*, vol. 16, no. 1, pp. 35–39, 2019, doi: 10.1109/RAAICON48939.2019.19.

[41] D. D. Franceschi and J. H. Jang, "Demystifying batch normalization: Analysis of normalizing layer inputs in neural networks," *Commun. Comput. Inf. Sci.*, vol. 1173 CCIS, pp. 49–57, 2020, doi: 10.1007/978-3-030-41913-4_5.

[42] Z. Q. Zhao, H. Bian, D. Hu, W. Cheng, and H. Glotin, "Pedestrian detection based on fast R-CNN and batch normalization," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10361 LNCS, pp. 735–746, 2017, doi: 10.1007/978-3-319-63309-1_65.

[43] A. Foroozandeh, A. Askari Hemmat, and H. Rabbani, "Offline Hand-written Signature Verification and Recognition Based on Deep Transfer Learning," in *Iranian Conference on Machine Vision and Image Processing, MVIP*, 2020, vol. 2020-Janua, doi: 10.1109/MVIP49855.2020.9187481.