JKAU: Comp. IT. Sci., **Vol**. **13** No. 1, pp: 86 – 102 (2024 A.D.)
DOI: 10.4197/Comp.13-1.5

Offline Signature Verification Using Deep learning and Genetic Algorithm

Abdoulwase M. Obaid Al-Azzani[1] and Abdulbaset M. Qaid Musleh[1]
[1]*Department of Computer Science and Information Technology, Sana'a University*
Sana'a, Yemen amalezzani71@gmail.com, aledresi200@yahoo.com

Abstract. the process of verifying signatures has wide-ranging applications in computer systems, including financial opera- tions, electronic document signing, and user identity verification. This approach has the advantage of community acceptance and presents a less intrusive alternative than other biological authentication methods. Deep learning (DL) and Convolutional Neural Networks (CNNs) have emerged as prominent tools in the field of signature verification, significantly enhancing the accuracy and effectiveness of these systems by effectively extracting discriminative features from signature images. However, optimizing the hyperparameters in CNN models remains a challenging task, as it directly affects the efficiency and accuracy of the models. Currently, the design of CNN architectures relies heavily on manual adjustments, which can be time consuming and may not yield optimal results. To address this issue, the proposed method focuses on employing a genetic algorithm to evolve a population of CNN models, thereby enabling the automatic discovery of the most suitable architecture for offline signature verification. By leveraging the optimization capabilities of the genetic algorithm, the proposed approach aims to improve the overall performance and effectiveness of the signature verification model. The effectiveness of the proposed method was evaluated using multiple datasets, including BHSig260-Bengali, BHSig260-Hindiin, GPDS, and CEDAR. Through rigorous testing, the approach achieved remarkable discrimination rates with a False Rejection Rate (FRR) of 2.5%-, False Acceptance Rate (FAR) of 3.2%-, Equal Error Rate (EER) of 2.35%-, and accuracy rate of 97.73%-.

Keywords—Offline Signature Verification, Convolutional Neural Network, Deep Learning, and Genetic Algorithm.

I. INTRODUCTION

relevance [5]. Given the continuous authorization of financial doc- uments and business transactions through signatures, the primary goal of handwriting signature verification systems is to differenti-

Biometric systems have become essential for personal authen- tication by employing behavioral or physiological characteristics. In the realm of biometrics, handwritten sig- natures have emerged as widely used tools for secure verification [1, 2]. Signature verifi- cation has been extensively researched, with a distinction between two main categories: online and offline [3]. Online signature verification focuses on capturing dynamic information during the writing process, whereas offline signature verification deals with static signature images, posing greater challenges and typically yielding lower accuracy compared to its online counterpart [4]. However, offline signature verification offers distinct advantages, despite its lower accuracy. It does not require specialized in- put devices, making it more accessible and applicable to a wider range of scenarios. Moreover, offline signature verification spans various domains, thereby expanding its potential applications and ate between genuine signatures created by authorized writers and

forged signatures produced by fraudulent individuals [6]. Forgery in the signature verification field can be categorized into three types [7]. Unskilled forgery occurs when a person forges another indi- vidual's signature without possessing knowledge of that person. Random forgery involves a person who knows only the signer's name without having previously seen its genuine signature. On the other hand, skilled forgery is performed by an individual who possesses knowledge of both the signer's name and the shape of their genuine signature. These distinctions highlight the complex- ity and importance of offline signature verification as they play a critical role in safeguarding against fraudulent activities. Further
advancements in this field have the potential to enhance security measures and improve the accuracy of signature-verification systems [8]. Handwritten Signature Verification systems employ two classifications of learning: writer-independent (WI) and writer- dependent (WD) [9, 10]. In the Writer-Independent state, learning is performed by all signatures in the database collectively, whereas in the Writer-Dependent state, learning is conducted independently for individual signatures. The WI method has gained popularity because it simplifies the addition of new individuals to the system, as the classification is based on a single category for all per-sons [11, 12]. In recent years, numerous automated systems have been developed to verify the authenticity of handwritten signatures us- ing various algorithms and methods. Deep learning, specifically Convolutional Neural Net-works (CNNs), has emerged as a dom- inant approach owing to its effectiveness in image classification and processing [13, 14]. CNNs, such as VGGNet, GoogleNet [15], ResNet [16], CapsNet [17], and DenseNet [18] have demon- strated significant improvements in efficiency and performance in real-world applications [19, 20]. The performance of CNNs re- lies heavily on their architecture [21, 22]. Experts in this field have designed different structures and versions to address specific classification problems. However, it is challenging to find a CNN model that can effectively solve all classification problems. The manual design of CNN architectures involves iterative attempts to find suitable parameters that yield the best results, which often requires a substantial amount of time [23]. Figures (1,2, and 3) show some samples from the dataset used.

Fig. 1. Sample of Signatures in BHSig260-Bengali Dataset.

Fig. 2. Sample of Signatures GPDS-300 Dataset.

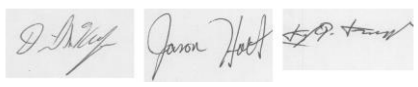

Fig. 3. Sample Signatures from the CEDAR Dataset.

To address this challenge, this study proposes a method that utilizes a genetic algorithm to optimize the hyperparameters of the CNN architecture for offline signature verification. The genetic algorithm assists in determining the optimal combination of hy- perparameters, significantly reducing the time required for manual design. By leveraging the genetic algorithm, the proposed method aims to enhance the performance and efficiency of the CNN model for offline signature verification, providing more accurate and re- liable results.

II. Literature Review

In the field of artificial intelligence, particularly deep learning, Convolutional Neural Networks (CNNs) have been widely used in various applications, including computer vision, pattern recog- nition, and natural language processing [24]. CNNs consist of several key components, including a Convolutional Layer, Acti- vating function, pool-ing layer, and fully connected layer. The Convolutional Layer applies filters (kernels) to extract features or patterns from the input image matrix, and multiple filters can be used to capture the different features. The Pooling Layer reduces the size of the matrices by applying functions, such as Max or Average pooling. The Fully connected layer is a multilayer percep- tron, where neurons are connected to all the nodes of the previous layer and are responsible for the final classification. Different ap- proaches have been proposed for offline signature verification. A method known as the Siamese network was introduced in [25]. It utilizes writer-independent (WI) feature learning and measures the similarity or dissimilarity between Siamese network outputs using the Euclidean distance. Another study [26] employed a Siamese Neural Network for signature verification, training, and evalua- tion of two similar neural networks on the same data. The use of the Siamese network architecture helped reduce the required training data volume and resulted in a 13 %- increase in system efficiency. Genetic algorithms have also been applied to optimize CNN architectures. For example, in a study by [27], two models for predicting the strength of adhesively bonded joints were de- signed using a CNN. The architecture of one model was manually developed, whereas the architecture of the other model was opti- mized using a genetic algorithm. The improved model with genetic algorithm optimization demonstrated better results. In image clas- sification tasks, genetic algorithms have been employed to optimize CNN architectures using datasets such as CIFAR10, MNIST, and Cal-tech256 [23]. By automatically adjusting the model's param- eters, the genetic algorithm improved the accuracy compared to the other tested models. In [28], the authors presented a hybrid ap- proach for extracting features from signature images. We utilized a Convolutional Neural Network (CNN) and Histogram of Oriented Gradients (HOG) techniques, followed by a feature-selection algo- rithm (Decision Trees) to identify important features. The CNN and HOG methods were combined. We evaluated the effective- ness of our hybrid approach using three classifiers: long short-term memory, support vector machine, and K-nearest Neighbor. The experimental results demonstrated that the proposed model per- formed satisfactorily in terms of efficiency and predictive ability. It achieved accuracy rates of 95.4 %-, 95.2 %-, and 92.7 %- with the UTSig dataset and 93.7 %-, 94.1 %-, and 91.3 %- with the CEDAR dataset. Another study [29] applied a genetic algorithm to select parameters such as the number of filters, filter size, and number of layers added to the trainable layers of a CNN transfer model. The proposed method achieved an accuracy of 97 %- in classify- ing cat and dog datasets over 15 generations. In the domain of finger-vein recognition, a system called a Genetic Algorithm with a Convolutional Neural Network (GA-CNN) was developed [30].

The GA-CNN system utilizes a genetic algorithm to initialize the training phase of the CNN, resulting in improved accuracy, sensitivity, and precision. Genetic algorithms have also been used for feature selection in signature verifications. In one study [31], a genetic algorithm was employed to select the optimal set of partial curves and features encoded into chromosomes for verification. In addition, genetic algorithms have been applied to weigh individual feature components in offline signature verification systems[32]. In [33], four different pattern representation schemes using genetic algorithms were used to determine the weights of feature-based classifiers, leading to increased verification accuracy. Further- more, a model was developed for offline signature verification using CNNs (VGG16, VGG19, and ResNet50) with additional parameters, and trained and tested on the SigComp2009 dataset. The VGG16 model demonstrated a high efficiency of 97 %- com- pared with the other models. In [34], a method was proposed to investigate the feasibility of employing Genetic Algorithms to automatically design CNN architectures. The Genetic Algorithm generates CNN architectures, which are then trained from the be- ginning using a Gradient-Descent Algorithm. The performance of the evolved CNN architecture was evaluated at each step of the evolutionary process, using a validation set. This algorithm does not require any preprocessing or post-processing of data before or after executing the Genetic Algorithm. In summary, this study focuses on developing an offline signature verification system us- ing Convolutional Neural Networks (CNNs) in combination with a genetic algorithm. A genetic algorithm was employed to search for the best model architecture hyperparameters and optimize the performance and accuracy of the system.

III. METHODOLOGY

Offline signature verification is a complex pattern-recognition problem that involves recognizing and verifying genuine handwritten signatures while detecting forgery attempts. To address this challenge, a comprehensive model for offline signature verifica- tion needs to be developed. Convolutional Neural Networks are particularly suitable architectures for signature verification [35]. The proposed model consists of the following stages. The first stage was the preprocessing stage, in which the signature image was prepared for further analysis. This typically involves tasks such as noise removal, image enhancement, and normalization to ensure consistent input for the subsequent stages. The second crucial stage is GA-based hyperparameter selection. Hyperpa- rameters are essential variables that determine the architecture and behavior of the Convolutional Neural Network (CNN). How- ever, manually finding optimal hyperparameters is a challenging and time-consuming task. By employing a genetic algorithm, the model can automatically search for and select the best combination of hyperparameters, leading to improved performance and accu- racy. The third stage involves the CNN itself, which is responsible for the feature extraction, training, and testing. CNNs are powerful deep-learning architectures that excel in the extraction of meaning- ful features from images. They consist of multiple convolutional
and pooling layers that learn the hierarchical representations of signature data. The extracted features are then utilized for training the model on a labeled dataset and for subsequent testing to evalu- ate the model's performance in signature verification. Each stage within the model comprises multiple steps, such as data prepro- cessing techniques, genetic algorithm initialization and evolution, CNN architecture design, training data preparation, model train-

ing, and testing. These steps work in conjunction to create an effective and robust offline signature verification system. Figure 4 visually represents the main stages and associated steps within each stage of the proposed model, providing a clear overview of the workflow involved in the offline signature verification.

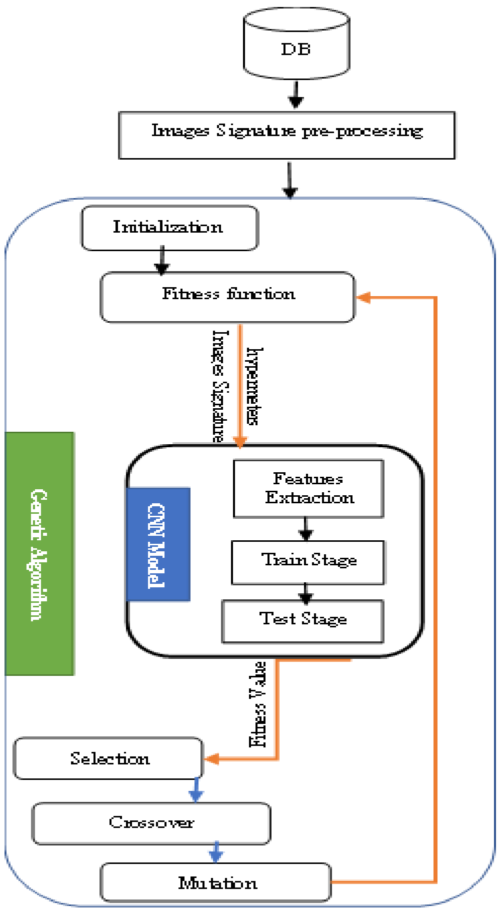

Fig. 4. Proposed model architectures

A Signature Images Preprocessing

Before starting feature extraction, essential processes must be applied to the image signature. These operations include the following.

- The color images were converted into grayscale images.
- Each image was resized to 100 × 100 pixels.
- The image points were read and stored in an image matrix.

B The Feature Extraction with CNN

The primary obstacle in addressing the issue of hand-written signature verification lies in identifying the distinguishing features that allow the system to differentiate between authentic and forged signatures [36]. Convolutional Neural Networks (CNNs) are a type of deep learning model primarily used for image and video analy- sis tasks[37]. They are designed to automatically learn and extract meaningful features from input data, making them well-suited for tasks such as image classification, object detection, and image segmentation. CNNs were inspired by the organization of the visual cortex in the human brain, which contains specialized neurons that respond to specific receptive fields. Similarly, CNNs consist of interconnected layers of artificial neurons that learn to recognize patterns and spatial hierarchies in the data. The key components of a CNN are convolutional, pooling, and fully connected layers. Here's how they work: Convolutional Layers: These layers per- form convolution operations on input data. A convolution involves sliding a small window, called a filter or kernel, over the input and computing dot products between the filter and the local patches of the input. This process captures the local patterns and features. Convolutional layers can have multiple filters to learn different fea- tures simultaneously. Pooling Layers: Pooling layers reduce the spatial dimensions of the data, helping to make the learned features more robust and invariant to small translations and distortions. The most common pooling operation is max-pooling, which selects the maximum value within each local region of the input. Fully Con- nected Layers: After several convolutional and pooling layers, the output is flattened and connected to the fully connected layers. These layers resemble traditional neural networks, in which each neuron is connected to every neuron in the previous layer. Fully connected layers learn global patterns and make predictions based on the extracted features. During the training process, CNNs learn to optimize their internal parameters (weights and biases) by min- imizing a chosen loss function. This is usually performed using gradient-based optimization algorithms, such as stochastic gradi- ent descent (SGD) or its variants. The backpropagation algorithm computes the weight loss gradients, allowing for efficient parame- ter updates. The proposed model function creates a convolutional neural network (CNN) model based on the parameters provided by the genetic algorithm and trains it using the given training data. Here, is a breakdown of the steps performed by function:

1) We defined an early stopping callback to monitor the valida- tion loss and stop training if the loss did not improve after two epochs.
2) A 2D convolutional layer is added to the model with a spec- ified number of filters, kernel size, and activation function. The input shape was set to the provided input shape.
3) Another 2D convolutional layer is added to the model with a specified number of filters, kernel size, padding, and activa- tion function.
4) A max pooling layer is added to the model with a specified pool size.
5) Steps 2 and 3 are repeated for two more convolutional layers.
6) Add a dropout layer with a specified dropout rate.
7) The output is flattened from the previous layers.
8) Add a dense (fully connected) layer to the model with a spec- ified number of units and activation function 'ReLU' [38].
9) Add another dropout layer with a specified drop-out rate.
10) A dense output is added to the layer with the activation func- tion 'Softmax' [39] and the specified number of classes and activation function.
11) Return the trained model using the training dataset.

C The Genetic Algorithm

AG is used to determine the best combination of hyperparame- ters for the convolutional neural network model, which can achieve high accuracy in classifying signatures. It begins by randomly ini- tializing a population of network configurations, where each con- figuration representation of genes is a set of hyperparameters for the convolutional neural network. The fitness of each network was evaluated by training and testing the corresponding convolutional neural network model on the signature data. The genetic algo- rithm then applies selection, crossover, and mutation operations to the population. Selection favors networks with higher fitness, allowing them to pass their genetic material (hyperparameters) to the next generation.

Crossover combines the genetic material of two-parent networks to create new child networks, potentially inheriting beneficial hyperparameter combinations. Mutation introduces random changes to the hyperparameters of the networks, promoting the exploration of the search space. This process of evaluating the fitness, selecting the best networks, generating new networks through crossover, and introducing mutations is repeated for multiple generations. The algorithm aims to iteratively improve the population by evolving networks with improved performance.

Table I. Random Generated Initial Population.

Hyperparameter	**Range**
Epoch	Random (2, 25)
Filter Size	Choice (16, 32, 64, 96)
Kernel Size	Choice [(3x3), (5x5)]
Unit	Choice (128, 256, 512)
Dropout	Choice (0, 0.25, 0.50)

a) **Initialization** The constructor initializes the hyperparame- ters randomly, including the number of epochs, filter size, kernel size, dropout rate, activation function, loss function, optimizer, and accuracy. This step returns a dictionary that contains the current values of the hyperparameters. A CNN model was built based on the given hyperparameters. It uses a combination of convolutional, pooling, dropout, and dense layers to construct the model. The model was compiled us- ing a specified optimizer, loss function, and metrics. This method initializes the attributes of an instance with random or predefined values, as listed in Table I. The attributes used included the following:

1) epoch: An integer attribute is randomly initialized be- tween 1 and 25.

2) filter1 and filter2: Integer attributes were randomly cho- sen from the values 64, 32, and 16.

3) units1: An integer attribute is randomly chosen from values 128, 256, and 512.

4) kernel1 and kernel2: Tuple attributes randomly chosen from the values (3, 3) and (5, 5).

5) dropout1 and dropout2: Float attributes randomly cho- sen from the values 0.25 and 0.5.

b) **Fitness Function** The fitness function equation represents the fitness of each network in the population. It trains and tests the Convolutional Neural Network model with the given hyperparameters, and calculates its accuracy. The accuracy was then stored. In addition, the function prints the accu- racy and classification reports for each network. The fitness function evaluates the fitness of each network in the network list by training and evaluating a convolutional neural network model for each network parameter configuration. This is an explanation of the steps performed by the function.

1) Selecting for each one the population dictionary from the list.

2) For convenience, we take the parameter values from the population dictionary and assign them to the corre- sponding variables.

3) We attempted to create and train a CNN model using the CNN model function with the extracted parameter values and input data.

4) The performance of the trained model is evaluated using the evaluation method on the test data, and the accuracy score is stored in the accuracy attribute of the network.

5) The accuracy of the model was obtained as a percentage.

6) Generate predictions using the trained model and obtain classification reports comparing predicted labels with true labels.

7) The updated list of networks is returned, including the accuracy values for each network.

c) **Selection** The selection function performs selection by sort- ing the population based on the accuracy of each network and retaining the top individuals. The number of individuals selected was equal to the population size.

1) The population list is sorted in descending order based on the individual accuracy attributes.
2) The top individuals are selected from the sorted popu- lation based on a specified percentage or number.
3) The selected population is then returned.

d) **Crossover** The crossover function is responsible for perform- ing a crossover by randomly selecting two parent networks from the population and creating two child networks. The cross-over process follows these steps for each of the two parents from the population, using a selection process:

1) The total number of attributes (hyperparameters) in the parents was divided into half.
2) Take the first half of the attributes from the first parent and assign them to the corresponding attributes of the second child.
3) Take the first half of the attributes from the second parent and assign them to the corresponding attributes of the first child.
4) Combine the offspring list, which includes two newly created child networks, with the current population, forming a new population.
5) Return to the new population.

This process enables the exchange of genetic information be- tween parent networks, allowing the child networks to inherit certain hyperparameters from their parents. By combining attributes from different parents, crossover promotes the ex- ploration and exploitation of potential solutions within a pop- ulation.

e) **Mutation** The mutation function is responsible for introduc- ing random mutations to the hyperparameters of the networks within the population. The mutation process follows these steps for each newly generated child from the previous pro- cess.

1) A random uniform function is used to generate a random number between 0 and 1.
2) The random " function" was used to generate a random integer within the specified range.
3) A random module was imported at the beginning of the code to access these functions.
4) The mutation process remains the same, where the "epoch" and "units" attributes of each individual are modified if the generated random number is less than or equal to 0.1.
5) Finally, the modified population was returned.

D Training and Testing Stage

In the training and testing stages of the signature verification process, our objective was to develop an efficient model for of- fline signature verification using a genetic algorithm and evaluate its performance on multiple datasets. Table II shows lists of the datasets used in the proposed model. Our study employed Con-

Table II. Dataset Used in the Proposed System.

Type	GPDS-300	CEDAR	Bengali	Hindi
Signers	300	100	100	160
Genuine	24	24	24	24
Forged	30	24	30	30
Training	10200	1540	3400	5440
Testing	6000	1100	2000	3200

volutional Neural Networks (CNNs), a deep learning technique known for its effectiveness in feature extraction for signature verification. However, manually designing CNN models often leads to suboptimal results because of the challenge of determining optimal architecture and hyperparameters. To address this challenge, we propose the use of a genetic algorithm, inspired by natural selection to evolve a population of CNN models. The genetic algorithm explores different combinations of architectural configurations, such as the number and size of convolutional layers, pooling layers, fully connected layers, activation functions, and regularization techniques. The fitness of each CNN model was evaluated using a fitness function that considers metrics such as accuracy, loss, and convergence speed. The algorithm selects the most promising models based on their fitness scores and applies genetic operators such as crossover and mutation to create a new generation of models. This evolutionary process continues, gradually improving the fitness of the models until the genetic algorithm identifies the CNN architecture with the best performance on the training dataset. Once the optimal architecture is determined, we proceed to the testing phase, where we evaluate the selected model on multiple datasets, including BHSig260-Bengali, BHSig260- Hindi, GPDS[40], and CEDAR, to ensure the robustness and generalization of our approach.

IV. RESULTS

During testing, we computed various performance metrics, such as the False Rejection Rate (FRR), False Acceptance Rate (FAR), Equal Error Rate (EER), and overall accuracy, to assess the effectiveness of our method. Our experimental results demonstrated impressive discrimination rates, with an FRR of 2.5%-, FAR of 3.2%-, EER of 2.35%-, and accuracy rate of 97.73%-. These findings highlight the effectiveness of our GA-based ap- proach in designing highly efficient and accurate offline signature verification models. The results are summarized in Table III, which presents the parameters and accuracy during the testing stage. Our study underscores the significance of leveraging genetic algorithms to optimize CNN architectures in signature verification systems. The superior performance of our proposed method has the poten- tial for various real-world applications that require reliable and non-intrusive signature verification.

V. DISCUSSION

Each dataset was subdivided into two parts: a training set and testing. The performance of the proposed system was measured using three global scales which are as follows: Accuracy is the ratio of the number of correctly categorized signatures to the total number of complete signatures. These are the False Acceptance Rate (FAR) and False Rejection Rate (FRR), which are the pres- ence of the forgeries signatures that are incorrectly classified. An equal Error Rate (EER) is applied to evaluate the equilibrium point where the FRR equals the FAR. A lower EER indicates a better per- formance for the model. The results obtained from the proposed method of constructing the CNN model using the genetic algorithm were compared with those of other methods using hand-built CNN models. The results were compared with those of other studies. Table IV. presents a comparison of the performance of different methods with our method on the CEDAR dataset in terms of FAR, FRR, EER, and accuracy. ”Our method” outperforms the other methods in terms of FAR, FRR, EER, and accuracy, indicating its superior performance on the CEDAR dataset. The ”Surround- edness Features” meth-od [11] achieves a False Acceptance Rate (FAR) of 8.33%-, False Rejection Rate (FRR) of 8.33%-, Equal Error Rate (EER) of 8.33%-, and accuracy of 91.67%-. On the other hand, the ”Multi-Path Siamese (MA-SCN)” method [41] yields an FRR of 18.35%-, FAR of 19.21%-, EER of 18.92%-, and accuracy of 80.75%-. Additionally, the ”Siamese CNN” method

[42] achieves a FAR of 6.78%-, FRR of 4.20%-, and accuracy of 95.66%-. In comparison, our proposed method outperformed these approaches with a FAR of 2.5%-, FRR of 2.2%-, EER of 2.35%-, and accuracy of 97.73%-.

Table III. The Parameter Setting of The Proposed System.

Parameter	Dataset Name			
	GPDS-300	***CEDAR***	***BHSig260-B***	***BHSig260-H***
Max Epochs	20	21	12	14
Parameters	2,402,731	2,545,911	2,426,120	1,241,250
Layer 1	Conv2D (32, 3x3)	(32, 3x3)	(64, 3x3)	(32, 3x3)
Layer 2 & MaxPool(2, 2)	Conv2D (32, 3x3)	Conv2D (64, 3x3)	Conv2D (32, 3x3)	Conv2D (32, 3x3)
Layer 3 & MaxPool(2, 2)	Conv2D (32, 3x3)	Conv2D (32, 3x3)	Conv2D (64, 3x3)	Conv2D (32, 3x3)
Layer 4 & MaxPool(2, 2)	Conv2D (32, 3x3)	Conv2D (64, 3x3)	Conv2D (32, 3x3)	Conv2D (32, 3x3)
Dropout	0.50	0.25	0.25	0.50
Flatten	512	264	512	256
Accuracy	0.93	0.977	0.958	0.922

Table IV. Comparison Results for CEDAR Dataset.

Method	CEDAR		
	FAR	***FRR***	***EER***
Surroundedness Features [11]	8.33	8.33	8.33
Multi-Path (MA-SCN) [41]	19.21	18.35	18.92
Siamese CNN [42]	6.78	4.20	–
Our method	**2.5**	**2.2**	**2.35**

Table V provides a comparison of different methods, includ- ing ”CNN-GP” [43], ”GoogLeNet Inception-v1 and Inception-v3” [44], and our method, on the GPDS-300 dataset in terms of FAR, FRR, EER, and accuracy. The ”CNN-GP” method achieves a FAR of 9.08%-, while the specific FAR value for the ”GoogLeNet Inception-v1 and Inception-v3” methods is not provided. In con- trast, our method achieved a FAR of 9.1%-. The ”CNN-GP” method has an FRR of 20.60%-, the specific FRR value for the ”GoogLeNet Inception-v1 and Inception-v3” methods is not pro- vided, and our method achieves an FRR of 20%-. Furthermore, the ”CNN-GP” method has an EER of 12.83%-, the ”GoogLeNet Inception-v1 and Inception-v3” methods have an EER of 26%-, whereas our method achieves an EER of 11%-. Finally, the ”CNN- GP” method has an accuracy of 92%-, the ”GoogLeNet Inception- v1 and Inception-v3” methods have an accuracy of 72%-, and our method achieves an accuracy of 93%-. Figures (5,6,7 and 8) show the graphs of the curves of the results of the proposed study, where the curves are loss, val-loss, val-accuracy, and accuracy with the used dataset.

Table V. Comparison Results for GPDS-300.

Method	GPDS-300		
	FAR	*FRR*	*EER*
CNN-GP [43]	9.08	20.60	12.83
GoogLeNet V1 and V3 [44]	–	–	26
Our method	**9.1**	**20**	**11**

Tables VI and VII compares the results of this study with the performance of our method on the BHSig260-B and BHSig260- H datasets. The ”Multi-Path Siamese (MA-SCN)” [41] method

achieves FAR values of 5.73%- and 9.96%- for BHSig260-B and BHSig260-H, respectively. The ”Siamese CNN” [**?**] method has FAR values of 14.25%- and 12.29%- for the respective datasets. For the ”Multi-scripted with CNN” [10] method, the FAR val- ues were 1.50%- and 2.31%- for BHSig260-B and BHSig260- H, respectively. In contrast, our method achieved FAR values of 1.3%- and 6.8%- for the same dataset. Regarding the FRR, the ”Multi-Path Siamese (MA-SCN)” method achieved rates of 4.86%- and 5.85%- for BHSig260-B and BHSig260-H, respectively. The ”Sia-mese CNN” method has FRR values of 6.41%- and 9.6%- for the respective datasets. The ”Multi-scripted with CNN” method achieves FRR values of 3.14%- and 6.65%-. In contrast, our method achieved FRR values of 2.1%- and 4.7%- for BHSig260-B and BHSig260-H, respectively. For the EER met- ric, the ”Multi-Path Siamese (MA-SCN)” method achieved rates of 8.18%- and 5.32%- for BHSig260-B and BHSig260-H, respectively. The ”Sia-mese CNN” method does not provide a spe- cific EER value, and the ”Multi-scripted with CNN” method lacks this information. In comparison, the proposed method achieved EER values of 1.7%- and 5.2%- for the respective datasets. In terms of accuracy, the ”Multi-Path Siamese (MA-SCN)” method achieved accuracy rates of 94.99%- for BHSig260-B and 92%- for BHSig260-H. The ”Siamese CNN” method achieves accuracy rates of 90.64%- and 88.98%- for the respective datasets. The ”Multi-scripted with CNN” method achieves accuracy rates of 95%- and 90%-. Our method outperformed the other methods, achieving the highest accuracy rates of 95.82%- for BHSig260-B and 92.26%- for BHSig260-H.

Table VI. Comparison Results for BHSig260-B

Method	**BHSig260-B**		
	FAR	***FRR***	***EER***
Multi-Path Siamese (MA-SCN) [41]	5.73	4.86	8.18
Siamese CNN [42]	14.25	6.41	–
Multi-scripted with CNN [10]	1.50	3.14	–
Our method	**1.3**	**2.1**	**1.7**

Table VII. Comparison Results for BHSig260-H

Method	**BHSig260-H**		
	FAR	***FRR***	***EER***
Multi-Path Siamese (MA-SCN) [41]	9.96	5.85	5.32
Siamese CNN [42]	12.29	9.6	–
Multi-scripted with CNN [10]	2.31	6.65	–
Our method	**6.8**	**4.7**	**5.2**

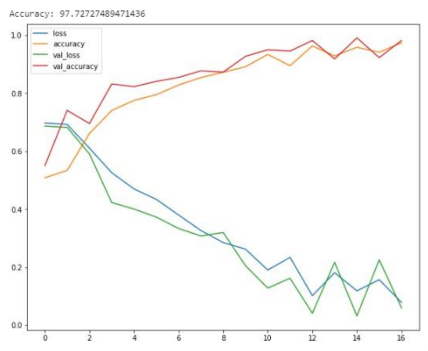

Fig. 5. The Resulting CEDAR Dataset

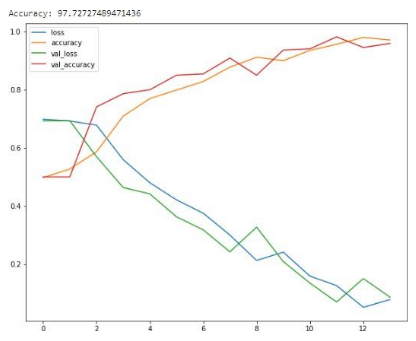

Fig. 6. The Resulting BHSig260-H.

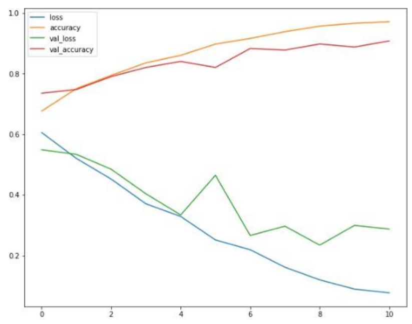

Fig. 7. The Resulting BHSig260-B.

VI. CONCLUSION

This study highlights the significance of the signature verifica- tion process in various applications, such as financial operations,

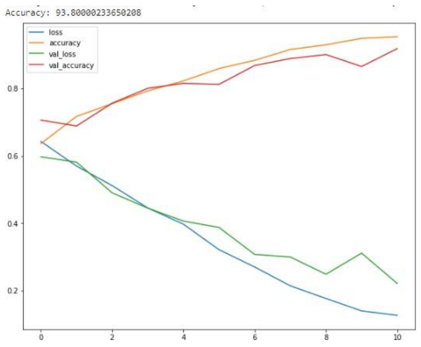

Fig. 8. The Resulting GPDS-300.

electronic document signing, and identity verification in computer systems. Compared with other biological methods, signature ver- ification offers community acceptance and is less invasive. Deep learning (DL) and Convolutional Neural Networks (CNNs) have significantly contributed to the advancement of signature verifi- cation systems by effectively extracting features from signatures. However, the optimization of hyperparameters for CNN models remains a challenging task in the design of highly efficient models with accurate results. Currently, CNN models are predominantly manually designed, which can be time-consuming and may not yield the best possible outcomes. To address this challenge, the proposed method utilizes a genetic algorithm to develop a pop- ulation of CNN models and identify the most suitable architec- ture for offline signature verification. The model was evaluated using multiple datasets including BHSig260-Bengali, BHSig260-Hindiin, GPDS, and CEDAR. The results of the proposed approach demonstrated its effectiveness, with the highest discrimination rates achieved. The False Rejection Rate (FRR) was 2.5%-, False Acceptance Rate (FAR) was 3.2%-, Equal Error Rate (EER) was 2.35%-, and accuracy rate was 97.73%-. In summary, the utiliza- tion of a genetic algorithm for optimizing the architecture of CNN models in signature verification leads to improved discrimination rates and accuracy. This study contributes to the development of highly efficient offline signature verification systems with poten- tial applications in various domains. Future work could focus on further enhancing the proposed method by exploring additional techniques for hyperparameter optimization. In addition, inves- tigating the generalizability of the developed model by testing it on larger and more diverse datasets would be valuable. Moreover, considering the robustness of the model against various types of signature forgeries and exploring methods to mitigate potential vul- nerabilities is an important direction for future research. Finally, incorporating real-time processing capabilities and evaluating the model's performance on streaming data can be explored to enhance its practical applicability in real-world scenarios.

References

[1] A. K. Jain, A. Ross, and K. Nandakumar, *Introduction to Biometrics*. Springer, 2016.

[2] D. Impedovo and G. Pirlo, "Automatic signature verification: the state of the art," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38,
no. 5, pp. 609–635, 2008.

[3] C.-L. Liu and Y. H. Yin, "Offline handwritten signature verification–literature review," *Pattern Recognition*, vol. 40, no. 8, pp. 2293–2307, 2007.

[4] M. A. Ferrer and C. M. Travieso, "Offline signature veri- fication: An overview and some recent advances," *Pattern Recognition Letters*, vol. 34, no. 3, pp. 249–256, 2013.

[5] B. M. Al-Maqaleh and A. M. Musleh, "An efficient offline signature verification system using local features," *Interna- tional Journal of Computer Applications*, vol. 131, no. 10,
pp. 39–44, 2015.

[6] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Of- fline handwritten signature verification—literature review," in *Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, IEEE, Nov. 1-8 2017.

[7] R. Verma and D. Rao, "Offline signature verification and identification using angle feature and pixel density feature and both method together," *International Journal of Soft Computing and Engineering*, vol. 2, no. 4, pp. 740–746,
2013.

[8] L. V. Batista, D. Rivard, R. Sabourin, and P. Maupin, "State of the art in off-line signature verification," in *Iberoamerican Congress on Pattern Recognition*, pp. 227–234, Springer, 2009.

[9] Y. Muhtar, W. Kang, A. Rexit, and K. Ubul, "A survey of offline handwritten signature verification based on deep learning," in *2022 3rd International Conference on Pattern Recognition and Machine Learning in PRML*, pp. 391–397, IEEE, July 2022.

[10] T. Longjam, D. R. Kisku, and P. Gupta, "Multi-scripted writer independent off-line signature verification using convolu- tional neural network," *Multimedia Tools and Applications*,
pp. 1–18, Aug. 2022.

[11] S. Pal, A. Alaei, U. Pal, and M. Blumenstein, "Performance of an off-line signature verification method based on texture features on a large indicscript signature dataset," in *12th IAPR Workshop on Document Analysis Systems*, pp. 72–77, IEEE, April 2016.

[12] A. Foroozandeh, A. Hemmat, and H. Rabbani, "Offline hand- written signature verification and recognition based on deep transfer learning," in *International Conference on Machine Vision and Image Processing (MVIP)*, pp. 1–7, IEEE, Feb. 2020.

[13] N. Sharma, S. Gupta, P. Mehta, X. Cheng, A. Shankar,
P. Singh, and S. R. Nayak, "Offline signature verification using deep neural network with application to computer vi- sion," *Journal of Electronic Imaging*, vol. 31, pp. 041210– 1–041210–10, Jul. 2022.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Na- ture*, vol. 521, pp. 436–

444, May 2015.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed,
D. Anguelov, and A. Rabinovich, "Going deeper with convo- lutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Confer- ence on Computer Vision and Pattern Recognition*, pp. 770–
778, 2016.

[17] E. Parcham, M. Ilbeygi, and M. Amini, "Cbcapsnet: A novel writer-independent offline signature verification model us- ing a cnn-based architecture and capsule neural networks," *Expert Systems with Applications*, p. 115649, Dec. 2021.

[18] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, (Honolulu, HI, USA), pp. 2261–2269, 2017.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[20] C. Clark and A. Storkey, "Training deep convolutional neural networks to play go," in *International Conference on Machine Learning*, pp. 1766–1774, PMLR, Jun. 2015.

[21] N. Purohit, S. Purohit, and C. S. Satsangi, "Offline handwrit- ten signature verification using template matching and clus- tering technique," *International Journal of Computer Science and Mobile Computing*, vol. 2, pp. 295–301, Apr. 2014.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint*,
pp. 1409–1556, Sep. 2014.

[23] F. Johnson, A. Valderrama, C. Valle, B. Crawford, R. Soto, and R. Ñ anculef, "Automating configuration of convolutional neural network hyperparameters using genetic algorithm," *IEEE Access*, vol. 8, pp. 156139–156152, 2020.

[24] J. Donahue, L. Anne, S. Guadarrama, M. Venugopalan,
S. Rohrbach, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and descrip- tion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2625–2634, 2015.

[25] S. Dey, A. Dutta, J. I. Toledo, S. K. Ghosh, J. Llado´s, and
U. Pal, "Signet: Convolutional siamese network for writer independent offline signature verification," *arXiv preprint arXiv:1707.02131*, 2017.

[26] C. Yinka-Banjo and C. Okoli, "Signature verification using siamese convolutional neural networks," *Covenant Journal of Informatics and Communication Technology*, 2019.

[27] E. G. Arhore, M. Yasaee, and I. Dayyani, "Optimisation of convolutional neural network architecture using genetic algo- rithm for the prediction of adhesively bonded joint strength," *Structural and Multidisciplinary Optimization*, vol. 65, no. 9,
pp. 1–16, 2022.

[28] F. M. Alsuhimat and F. S. Mohamad, "A hybrid method of feature extraction for signatures verification using cnn and hog: A multi-classification approach," *IEEE Access*, vol. 11,
pp. 21873–21882, 2023.

[29] C. Li, J. Jiang, Y. Zhao, R. Li, E. Wang, X. Zhang, and
K. Zhao, "Genetic algorithm based hyper-parameters opti- mization for transfer convolutional neural network," in *In-ternational Conference on Advanced Algorithms and Neural Networks (AANN 2022)*, vol. 12285, pp. 232–241, SPIE, Jun.

2022.

[30] O. M. Assim and A. M. Alkababji, “Cnn and genetic al- gorithm for finger vein recognition,” in *2021 14th Interna- tional Conference on Developments in Systems Engineering (DeSE)*, pp. 503–508, IEEE, Dec. 2021.

[31] X. Yang, X. Zeng, H. Fu, and Y. Zhang, “Selection of features for signature verification using the genetic algorithm,” *Com- puters and Industrial Engineering*, vol. 30, no. 4, pp. 1037– 1045, 1996.

[32] V. E. Ramesh and M. Narasimha, “Off-line signature verifica- tion using genetically optimized weighted features,” *Pattern Recognition*, vol. 32, no. 2, pp. 217–233, 1999.

[33] D. P. Sudharshan and R. N. Vismaya, “Handwritten signa- ture verification system using deep learning,” in *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, pp. 1–5, IEEE, Jul. 2022.

[34] A. S. Mondal, “Evolution of convolution neural network ar- chitectures using genetic algorithm,” in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, Jul. 2020.

[35] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, “Learning features for offline handwritten signature verification using deep convolutional neural networks,” *Pattern Recognition*, vol. 70, pp. 163–176, 2017.

[36] V. Malekian, A. Aghaei, M. Rezaeian, and M. Alian, “Rapid offline signature verification based on signature envelope and adaptive density partitioning,” in *2013 First Iranian Confer- ence on Pattern Recognition and Image Analysis (PRIA)*, pp. 1–6, IEEE, Mar. 2013.

[37] A. M. Q. Musleh and A. M. O. Al-Azzani, “Developing a model for offline signature verification using cnn architec- tures and genetic algorithm,” *IEEE Access*, vol. 1, no. 3, pp. 1–1, 2023.

[38] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[40] M. A. Ferrer, J. F. Vargas, A. Morales, and A. Ordonez, “Robustness of offline signature verification based on gray level features,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 966–977, 2012.

[41] X. Zhang, Z. Wu, L. Xie, Y. Li, F. Li, and J. Zhang, “Multi- path siamese convolution network for offline handwritten sig- nature verification,” in *2022 The 8th International Confer- ence on Computing and Data Engineering*, pp. 51–58, Jan. 2022.

[42] W. Xiao and Y. Ding, “A two-stage siamese network model for offline handwritten signature verification,” *Symmetry*, vol. 14, no. 6, p. 1216, 2022.

[43] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, “Writer- independent feature learning for offline signature verifica- tion using deep convolutional neural networks,” in *2016 In- ternational Joint Conference on Neural Networks (IJCNN)*, pp. 2576–2583, IEEE, 2016.

[44] S. M. Sam, K. Kamardin, N. N. A. Sjarif, and N. Mo- hamed, “Offline signature verification using deep learning convolutional neural network (cnn) architectures googlenet inception-v1 and inception-v3,” *Procedia Computer Science*, vol. 161, pp. 475–483, 2019.

التحقق من التوقيع دون اتصال بالإنترنت باستخدام التعلم العميق والخوارزمية الجينية

عبدالواسع محمد عبيد العزاني [١]، عبدالباسط محمد قايد مصلح [٢]

[١] *قسم علوم الحاسبات، كلية الحاسبات وتقنية المعلومات*

جامعة الملك صنعاء، اليمن

Aledresi200@yehoo.com

مستخلص. إن عملية التحقق من التوقيعات لها تطبيقات واسعة النطاق في أنظمة الكمبيوتر، بما في ذلك العمليات المالية، التوقيع الإلكتروني للمستندات والتحقق من هوية المستخدم. يتمتع هذا النهج بميزة قبول المجتمع ويقدم بديلاً أقل تدخلاً من طرق المصادقة البيولوجية الأخرى. التعلم العميق والعصبية التلافيفية برزت الشبكات كأدوات بارزة في مجال لتحقق من التوقيع مما أدى إلى تعزيز دقة وفعالية هذه الأنظمة بشكل كبير من خلال استخلاص الميزات التمييزية بشكل فعال من صور التوقيع. ومع ذلك، يظل تحسين المعلمات الفائقة في نماذج ث مهمة صعبة، لأنه يؤثر بشكل مباشر على كفاءة النماذج ودقتها. وحالياً، يعتمد تصميم بنيات بشكل كبير على التعديلات اليدوية، والتي يمكن أن تستغرق وقتاً طويلاً ربما أيضاً لا تسفر عن النتائج المثلى. ولمعالجة هذه المشكلة، تركز الطريقة المقترحة على استخدام خوارزمية جينية للتطور مجموعة من نماذج، مما يتيح الاكتشاف التلقائي للبنية الأكثر ملاءمة للتوقيع دون اتصال بالأنترنت تحقق. من خلال الاستفادة من قدرات التحسين للخوارزمية الجينية، يهدف النهج المقترح إلى تحسين الأداء العام وفعالية نموذج التحقق من التوقيع. تم تقييم فعالية الطريقة المقترحة باستخدام مجموعة بيانات متعددة، من خلال اختبارات صارمة، حقق النهج معدلات تمييز ملحوظة مع معدل رفض كاذب بنسبة ٥٠٢ ومعدل قبول كاذب بنسبة ٢٠٣، ومعدل خطأ مساوٍ بنسبة ٣٥٠٢ ، ومعدل ودقة قدرة ٠٠٧٣٠٩٧

الكلمات المفتاحية. التحقق من التوقيع دون الاتصال، التعلم العميق، والخوارزمية الجينية.