Using Residual Networks and Cosine Distance-Based K-NN Algorithm to Recognize On-Line Signatures

GIBRAEL ABOSAMRA AND HADI OQAIBI
Department of Computer Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
Corresponding author: Gibrael Abosamra (gabosamra@kau.edu.sa)

ABSTRACT As a result of the urgent need to immediately identify individuals through the Internet, especially given the Coronavirus (COVID-19) pandemic at present, the recognition of online handwritten signatures has quickly evolved to become an urgent and necessary matter. However, signature identification remains challenging in the pattern recognition field due to intra-class variability and inter-class similarity. Intra-class variability is a characteristic of human behavioural activities, particularly in handwriting where no two handwritten signatures of any person can exactly coincide. The inter-class similarity is also a characteristic of human movement-based activities such as handwritten signatures particularly when the number of writers is large. In this research, an optimized transfer-learning-based architecture is proposed as a highly accurate identification technique for online-signatures using ResNet18 as a feature extraction deep-learning module. The X-Y time-series signals of the signatures were initially converted into images and used in retraining the ResNet18 model to achieve relatively high accuracy. The retrained ResNet18 model was then used to extract features that possess high discriminative distances among different classes of handwritten signatures. The model’s deep layers were searched to determine the best layer that provided the most discriminative features when using a 1-nearest neighbour learning algorithm based on the cosine distance. By using an ensemble of five models trained on rotated versions of the original signatures and using only three training samples from each writer, the classification accuracy achieved 100% when applied on the genuine signatures of public datasets such as SVC 2004 TASK1 and TASK2, and a new proprietary dataset composed of 120 genuine users. When the abovementioned technique was tested on the aggregated version of the aforementioned datasets, the resultant accuracy was still above 99%. Moreover, the robustness of the technique was proven by testing the generated models trained with one dataset with the other two datasets resulting in accuracy above 99% for all combinations.

INDEX TERMS Convolutional networks, cosine distance, deep residual networks, K-nearest neighbour algorithm, machine learning, online signature recognition, transfer learning.

I. INTRODUCTION
The development and spread of e-commerce systems and the abrupt rise in online commercial and managerial transactions resulting from the present Coronavirus (COVID-19) pandemic have caused online signature recognition and verification to be one of the top requirements to support online identification and authentication. The problem of online handwritten signature identification (OHSI) is a distinct challenge in the field of pattern recognition given the intra-class variability and the high similarity between some writers’ signatures [1]. The third difficulty associated with signature classification is the small number of training samples per user that can be practically available to build the trained model of the recognition system; particularly in building deep-learning-based models [2]. Traditional pattern recognition techniques have been developed to address this challenge, though resulting in a level of accuracy that is insufficient for real governmental or business applications. Recently deep-learning techniques have presented significant and exciting breakthroughs in the image classification process, which upgraded the recognition accuracy of intelligent systems to...
a level comparable with human experts [3]. Even though most of the research carried out on online-signatures is concentrated on the signature verification process [3], this research focuses solely on the identification process as a first phase, hoping to achieve 100% accuracy to allow the use of signatures instead of user names or other IDs. Traditional identification and authentication methods based on the person’s knowledge, such as their personal identification number (PIN), passwords, and tokens, are prone to theft, misuse, and forgetfulness risks [4]. The handwritten signature recognition technique, when combined with another biometric test, could be used for identification and authentication tasks.

Based on the above discussion and background, this study aims to address the OHSI challenges and to meet the substantial needs for generalization of the OHSI by examining the features produced by different layers of a deep residual network and the best distance to use in the discrimination among the signatures of different writers. The adoption of the new approach to OHSI is proposed by asking four questions: 1. Which DCNN provides the best accuracy in the identification of signatures? 2. Which DCNN layer provides distinct features for the discrimination of signatures? 3. Which type of distance is best when using a k-nearest neighbour algorithm to differentiate between different user signatures? 4. How to build an ensemble of extractors and aggregate them to enhance the identification accuracy?

In answering these four questions, many experiments were carried out on the benchmark data set, SVC 2004 TASK 1 published in the first signature verification competition held in 2004 [1] to achieve an optimized architecture. Next, the optimized architecture was tested on SVC TASK2 and a new proprietary dataset. The new data set that was constructed and tested by our proposed architecture was then published to enrich the signature research resources. Finally, comparisons with state-of-the-art techniques were demonstrated.

In Section II, a brief survey is introduced to summarize the efforts performed in online-signature recognition during the past four years concentrating on the work conducted on the SVC 2004 TASK1 and TASK2 datasets. In Section III, a concise presentation of the required scientific background is introduced. In Section IV, an overview of the proposed technique is presented, and the first phase of the solution is demonstrated with detailed experiments in conjunction with the second phase of the solution. In Section V, the recognition accuracy is enhanced by using features from an ensemble of 3 to 7 models of the ResNet18 trained at different rotation angles. In Section VI, further enhancements and evaluation of the proposed technique are introduced. In Section VII, comparisons with state-of-the-art techniques are conducted with detailed discussion. Finally, our contributions are summarized in the conclusion section of this paper.

II. BRIEF SURVEY

In this section, we will summarize the work done on the datasets SVC2004 TASK1 and TASK2. Given we are concerned with the recognition only at this stage, we mention all the techniques used in recognition of TASK1 and TASK2 over the past four years (2017-2020) with the inclusion of several techniques employed in the verification process that recorded verification based on the random forgery test. Since random forgery samples are drawn from the genuine signature samples, its recorded equal error rate (ERR) metric can be considered an indication concerning the technique’s accuracy when used in the identification process, especially when all remaining genuine signatures are used in the random forgery test. The recognition accuracy in the case of using all genuine samples in the random forgery can be approximated to 100 minus EER. The approximation here is due to the absence of the decision to whom a test signature will be assigned if both the true positive, genuine user and the false-positive user accept this signature based on the decision threshold for each. If the random forgery EER is zero, then the recognition accuracy is equal to 100% without approximation.

A very important parameter in all the surveyed work is the number of genuine samples used in the training process (NTGS) of a machine-learning (ML) or deep-learning (DL) model or as reference samples when using matching techniques. Another important feature relates to the signals used in the mentioned techniques which are x, y, and (P and A), which represent the pen (pressure and inclination angles) during the generation of the signature points. In TASK1, only x, y, timestamp, and pen-up/down information signals are recorded, while in TASK2 there are x, y, inclination angle (Azimuth, Altitude), and pressure information. TASK1 and TASK2 each contain signatures of 40 genuine users having 20 genuine signatures combined with 20 skilled forgery signatures. The 20 genuine signatures were conducted in two sessions separated by a period of two weeks. In this research, we are concerned only with the identification of genuine signatures.

The techniques used in signature recognition can be classified into three categories. The first category uses matching techniques (MT) where a limited number of samples for every genuine user are stored as reference samples. An unknown sample is compared with all the reference samples of all users using (in most cases) Dynamic Warping Techniques (DWT) which assigns the unknown sample to the user Id whose reference sample has the smallest distance. The second category uses machine-learning techniques (MLT) where a model is trained using hand-crafted features extracted from a certain number of genuine signature samples and then the trained model is used in the classification of the unknown samples. The third category uses deep-learning techniques (DLT), where features are learned by the deep-learning model using a relatively large number of signature samples from every genuine user. Some hybrid techniques (HBT) combine two or more techniques to solve the signature recognition problem. All three mentioned techniques and the hybrid ones can be categorized into more detailed categories that can be reviewed in the detailed signature recognition and/or
verification surveys, such as [2], [5], [6]. In this research, we use a hybrid technique based on DL and ML.

A. MACHINE-LEARNING-BASED TECHNIQUES

The researchers [7] used a Dual-Tree Complex Wavelet Packet Transform (DTCWPT) to extract features from the three signals \(x, y,\) and \(p\). When they used Support Vector Machines (SVM) based on radial basis function (RBS) in the classification of SVC2004 TASK2 using 12 genuine signature training samples (NTGS = 12), they achieved recognition accuracy of 99.84%. The researchers [4] combined the probabilistic output of the Hidden Markov Model (HMM) and SVM using Dempster-Shafer Theory (DST) in the classification of online-signatures. When they tested their technique on SVC2004 TASK1 based on 4 and 16 genuine signature samples (NTGS = 4 and 16), they achieved 95.04% and 99.17% respectively.

In [8], local dynamic features (LDF) are collected from each segment of a signature and random forest (RF) was used in the recognition process after optimization of the model parameters and the best feature set using particle swarm optimization (PSO). The recognition accuracy of 99.63% was achieved when this technique was applied to TASK2 using 5 genuine samples per user for training.

B. MATCHING BASED TECHNIQUES

In [9], a two-stage verification technique is proposed where shape-context features (SCF) are first extracted and compared with the corresponding features of the reference samples of the claimed user to filter-out non-skilled forgeries, while in the second stage, function-based features (FBF) are compared using shape context-dynamic time warping (SC-DTW). After applying this technique to the SVC2004 TASK2 dataset using five reference samples per genuine user, it achieved 0.3% EER in the random forgery test where 20 signatures are randomly selected from the other 39 users. Although not all the samples of other users were used in the test we can consider the recognition accuracy to be approximately 99.7%. In [10], one genuine signature was used for each user to generate many (16 or 64) duplicate reference samples based on a sigma-lognormal-signature model. A DTW verification system was then used in the comparison between the unknown sample and the stored duplicates of each user. In the random forgery test, the system achieved 1.5% EER and 0.5% EER when applied to TASK1 and TASK2, respectively. In [11], for each writer, 80 specific features were selected that were compared with their corresponding features of the unknown-signature based on a fuzzy similarity measure. An EER of 0.902 was reported when the random forgery test was applied to TASK2 in which one random signature was tested from each one of other genuine users. Hence there is an approximate recognition accuracy of 99.098%.

C. DEEP-LEARNING-BASED TECHNIQUES

In [12], the researchers used a shallow CNN (sCNN) composed of three convolutional layers with three max-pooling layers, ending with a fully-connected layer and Softmax layer for classification purposes. By using 15 training samples (5 for testing), they achieved recognition accuracy of 97.0% when their model was tested with the SVC 2004 dataset (TASK1 and/or TASK2 not specified). In [13] a very complex system was introduced where both traditional statistical features and deep features from a convolutional auto-encoder were fused and then supplied to a stack of a depth-wise separable based convolutional network (DWSCNN) and a long short-term memory (LSTM). The output of the LSTM was then fed to a multi-layer perceptron (MLP) which had two outputs indicating either genuine or forgery. When their complex architecture was tested on SVC 2004 TASK2 using one training sample an EER of 3.01 was recorded for random forgery that used only one random sample from each one of the other genuine users in the testing phase. Hence an approximate accuracy of 96.99% can be considered for the identification process.

III. SCIENTIFIC BACKGROUND

The construction and operation of the convolutional networks in conjunction with the basic concept of the residual networks are reviewed in this section.

A. CONVOLUTIONAL NETWORKS

To understand the convolutional network, we will describe the function of one layer of a convolutional network first. The output of a convolutional layer depends on the number of convolutional filters (CFs) and the size of the filter. In the light of the neural networks we can imagine a CF as a moving neuron with a rectangular weight matrix. Since the input image has a larger size, the CF is applied to every rectangle of pixels matching its size inside the input image generating a feature map having a size equal or slightly smaller than the input image throughout a scanning operation from left to right and from top to down. If the moving filter has a step greater than 1 in any direction, then the output feature map will contain a down sampled filtered version of the input image with a factor proportional to the step (stride) length in each direction.

The output of a single filter can represent different types of image processing operations based on the filter weight matrix values such as edge detection, sharpening, smoothing, contrast adjustment, and so on. In a CNN, the filter weights are randomly initialized and then learned during the training process.

Since we have many CFs (N for example) at each convolutional layer, we will get an N output filtered images called feature maps or channels (CHs).

Moreover, by cascading many convolutional layers, successive mapping of the input image is carried out until reaching the required final representation. By including pooling layers that perform down sampling between different convolutional layers the final output size will be smaller and can be mapped easily to single or multiple outputs using a fully-connected layer to present the single or multiple class outputs.
of the deep network. Through the proper training of all the CFs and the fully-connected layer using a backpropagation algorithm that optimizes all the network weights to satisfy the input-output relation between the input training samples and their associated output labels or regression values, the trained model can be used in classifying or determining the corresponding values of an unknown image sample. Fig. 1 presents a convolutional network having a single-channel input layer with two convolutional layers and one fully-connected layer. As can be seen in Fig. 1, each CF is responsible for generating a single feature map. The CFs in the first convolutional layer have 2-D weight matrices because the input is a single-channel while the CFs in the second convolutional layer have 3-D weight matrices (tensors) since the input to this layer has N CHs.

If the first CF has a weight matrix of size \( F_w \times F_h \) with a stride of 1 by 1 and the input image IM has a size of \( I_w \times I_h \), then, each channel \( k \) of the N output channels (CH\( k \)) can be calculated using the following formula:

\[
CH^k(x, y) = \sum_{i=-F_w/2}^{F_w/2} \sum_{j=-F_h/2}^{F_h/2} IM(x + j, y + i) \times CF^k(j, i) \text{ for } \frac{F_w}{2} \leq x \leq \frac{I_w}{2} - \frac{F_w}{2}, \frac{F_h}{2} \leq y \leq \frac{I_h}{2} - \frac{F_h}{2} \quad (1)
\]

where \( k \) varies from 1 to N (number of CFs).

If there is no zero-padding as assumed in the previous formula, the output CH will have a size of \( (I_w - F_w + 1) \times (I_h - F_h + 1) \). If the input layer has N channels, the CFs will have weight matrices (tensors) of dimension \( F_w, F_h, N \) as shown in the second convolutional layer in Fig. 1.

In the general case, the number of FLOPS needed to generate NO output channels (OCH\( s \)) with size \( OCH_w \times OCH_h \) from NI input channels, can be estimated as

\[
NFLOPS = F_w \times F_h \times OCH_w \times OCH_h \times NI \times NO \quad (2)
\]

The max-pooling layer does not have learnable weights since it simply selects the maximum value within the specified 2-D size. Here, the number of output channels OCH\( s \) of a pooling layer is equal to the number of input channels (ICH\( s \)) and hence the number of FLOPS used by the pooling layer will be \( F_w \times F_h \times ICH_w \times ICH_h \times NI / 4 \) in case of a stride of 2 \( \times 2 \). Since the number of FLOPS of the 1\textsuperscript{st} layer is affected linearly by the input image’s size \( (I_w \times I_h) \), all the number of FLOPS of the subsequent layers including the max or average-pooling layers, will also be affected linearly by \( (I_w \times I_h) \). Only the 1\textsuperscript{st} layer will be affected by the number of channels of the input image. Hence the total inference time of the whole DCNN is mostly proportional to \( (I_w \times I_h) \).

Accordingly, the time complexity of DCNN is \( O(d) \) where \( d = (I_w \times I_h) \), because the image is the feature vector in this case. In multi-classification tasks, the network is always finalized by a Softmax layer that transforms the input real values into relative class probabilities such that all the outputs sum to 1. Further details can be found in [14].

**B. RESIDUAL NETWORKS**

Residual networks are special types of convolutional networks that are introduced to allow deeper networks to function properly by avoiding the vanishing gradient problem that appears when the number of layers becomes large [15]. This is accomplished by introducing a short-cut path from the residual block’s input and adding it to the output of the convolutional layer. In this case, the purpose of the internal mapping would be learning the residual of the mapping function instead of learning the original mapping. In other words, instead of learning the mapping \( MP(x) \), the layer or two layers will learn \( 1-\text{MP}(x) \) preventing the vanishing gradient problem when training the network by reusing activations from a previous layer until the adjacent layer learns its weights. Residual networks proved superior in handwritten character classification [16]. In this research, we will use the lightest version of the residual networks’ family (ResNet18). The prebuilt version of ResNet18 contains eight residual blocks. Five residual blocks have direct short-cuts and the other three use point-wise CFs (the filter size is \( 1 \times 1 \)) instead of the direct short-cuts to perform down sampling by 2 using a stride of \( 2 \times 2 \). The time complexity of the used point-wise CFs is \( ICH_w \times ICH_h \times NI \times NO / 4 \). Seventeen convolutional layers are doing standard convolution, in which all of them have filters of size \( 3 \times 3 \), and only the first convolutional layer before the first residual block has a \( 7 \times 7 \) filter size which does down sampling by 2 of the input image. Only one max-pooling layer is used for down sampling by 2 after the first CF and one global average-pooling filter after the final residual block to
down sample the feature maps into $1 \times 1$. The input image size is down sampled five times before the global average-pooling filter to produce $512 \times 7 \times 7$ feature maps. Our research will prove empirically that the $512 \times 7 \times 7$ feature maps provide the best representation of the signature image. After each convolutional layer, there is a batch normalization layer to normalize the output feature maps and a linear rectifying layer (ReLU) to rectify the negative output values. The time complexity of the batch normalization layer is proportional to $IICH_w \times IICH_h \times NI$. Accordingly, the time complexity of ResNet18 is almost proportional to the input image dimensions $I_w \times I_h$ (only the fully connected layer and the Softmax layer have independent time complexities proportional to the number of output classes). The memory size of ResNet18 is 44 Mb and the number of parameters is $1.17 \times 10^7$. The ResNet18 inference phase consumes about 1.82 G FLOPS when the input image size is $224 \times 224 \times 3$ and the mini-batch-size is 1. If the multiply-accumulate (MACC) operations are not fused, then all the mentioned convolutional time complexities should be doubled. Using high-performance hardware such as a workstation equipped with an NVIDIA Titan X Pascal, ResNet18 achieved super-real time performance in the image classification (> 60 frames per second (FPS)) [17]. Also, in [17], it was proven empirically that ResNet18 is one of the best DCNNs if considering inference time and memory usage (maximum memory utilization <0.7 Gb; this is why ResNet18 is categorized as a low memory usage model).

### C. K-NEAREST NEIGHBOR ALGORITHM (K-NN)

K-NN is an instance-based learning algorithm that is proposed by Thomas Cover, in which a certain number of reference samples or all previous samples of every class are stored. The samples are stored in n-dimensional space based on the number of features defined for the sample type. When an unknown sample is inputted to the system, a search for the k-nearest samples is performed, and then the sample is assigned to the class having the majority among the found k-samples. Different types of distances can be calculated to find the k-nearest samples. Table 1 displays some of the most used distances and the formula used in their calculations. Different versions and improvements for the k-NN can be found in [18], [19].

### IV. THE ARCHITECTURE OF THE PROPOSED TECHNIQUE

The block-diagram of the proposed technique is shown in Fig. 2. The proposed technique is implemented in two phases. In the first phase, the online signatures’ signals are smoothed, normalized and converted into images and then split into training and testing datasets. A convenient DCNN is then selected from many on-the-shelf models based on the test accuracy when using a variable number of training samples per user. In the second phase, the selected DCNN is used as a feature extractor.

Since many layers can be used in the feature extraction process, a search for the best layer in the chosen model is

| TABLE 1. The most familiar distances used in k-NN with their formulas when measuring the distance between two n-dimensional real value vectors. |
| --- |
| **Metric** | The distance between two n-dimensional vectors X and Y | Notes |
| City block/Manhattan distance | $\sum_{j=1}^{n} |x_j - y_j|$ | |
| Chebychev Distance | $\max_{i=1}^{n} |x_i - y_i|$ | Maximum coordinate difference |
| Euclidian Distance | $\sqrt{\sum_{j=1}^{n} (x_j - y_j)^2}$ | Calculated based on the Pythagorean theorem but in n dimensions. |
| Standardized Euclidian Distance | $\sqrt{\sum_{j=1}^{n} \frac{1}{S_j^2} (x_j - y_j)^2}$ | Standardized Euclidian distance is an Euclidian distance of normalized data where each $j^\text{th}$ sample value is divided by its standard deviation ($S_j$). |
| Cosine distance | $1 - \frac{\sum_{j=1}^{n} x_j \cdot y_j}{\sqrt{\sum_{j=1}^{n} (x_j)^2} \cdot \sqrt{\sum_{j=1}^{n} (y_j)^2}}$ | One minus cosine of the included angle between points (treated as vectors) |
| 1-cosine similarity | $1 - \frac{\sum_{j=1}^{n} (x_j - \bar{x}) \cdot (y_j - \bar{y})}{\sqrt{\sum_{j=1}^{n} (x_j - \bar{x})^2} \cdot \sqrt{\sum_{j=1}^{n} (y_j - \bar{y})^2}}$ | Correlation similarity is equal to cosine similarity after subtracting the mean from every element value. |
| Spearman distance | $1 - \frac{\sum_{j=1}^{n} (R(x_j) - \bar{R}(x)) \cdot (R(y_j) - \bar{R}(y))}{\sqrt{\sum_{j=1}^{n} (R(x_j) - \bar{R}(x))^2} \cdot \sqrt{\sum_{j=1}^{n} (R(y_j) - \bar{R}(y))^2}}$ | The spearman is a Pearson correlation of ranked data where: $R(x_j)$: rank of $x_j$ in x. $R(y_j)$: rank of $y_j$ in y. |
performed based on the accuracy resulting after using a k-NN algorithm. Since many distance types can be used in the k-NN algorithm, an experiment is conducted to select the best distance type based on the accuracy resulting after using a 1-NN algorithm. Finally, an ensemble of $m$ (3, 5, or 7) models trained with rotated versions of the training dataset is used as a feature extractor. When a new test signature is inputted to the system, its corresponding $m$ sets of features are extracted from the $m$ DCNN models and combined in one array. The cosine distances between this array and the combined features of all the training samples of all genuine users are calculated. The ID of the signature is assigned to the one with the minimum distance. The two phases are explained in more detail in the following subsections.

**A. FEATURE EXTRACTOR TRAINING PHASE**

In this phase, the time-series signals of every signature sample are smoothed and normalized and then converted into a colour bitmap image. Next, genuine images are split into training and testing based on the chosen number of training samples (NTGS). After evaluation of many on-the-shelf DCNN models on the base dataset, the best DCNN is selected to use for feature extraction in the second phase.

1) **SIGNATURE SIGNAL PREPROCESSING AND NORMALIZATION**

To remove noise due to fluctuations of the electricity or friction during the movement of the pen on the surface of the tablet, a simple 5 pin moving average filter is used as follows:

\[
x(t) = 0.2 \times \sum_{t-2}^{t+2} X(t)
\]

\[
y(t) = 0.2 \times \sum_{t-2}^{t+2} Y(t)
\]

Next, the signal values are normalized such that the X-Y values lie between 0 and 1 using the following two formulas:

\[
xn(t) = \frac{x(t) - xmin}{xmax - xmin}
\]

\[
yn(t) = \frac{y(t) - ymin}{ymax - ymin}
\]

2) **SIGNATURE TIME SERIES CONVERSION INTO 2-D IMAGES**

The points corresponding to the signature $X_Y$ time-series signals are drawn on a coloured bitmap image with a black background where each point is drawn with an RGB colour intensity (0.3,0.3,0.3). The $X$ and $Y$ positions are scaled based on the target image size leaving a 10% blank area.

**FIGURE 2.** The block diagram of the proposed technique to identify signatures.
around the signature image on each side to avoid the CFs padding effects [16]. Linear interpolation is used to fill the gaps between existing points. The scaling in the case of a bitmap size of $SZ \times SZ$ is as follows:

$$xp(t) = SZ \times (0.1 + 0.8 \times xn(t))$$ (7)

$$yp(t) = SZ \times (0.1 + 0.8 \times yn(t))$$ (8)

Figure 3 shows the smoothed x-y signals (2-a, 2-b) and the corresponding signature image (2-c).

3) CHOOSING A PROPER READY-MADE DCNN

In order to select the best ready-made DCNN, we carried out numerous experiments with different DCNNs in classifying the genuine dataset by splitting the genuine dataset into a training dataset that includes NTGS samples per genuine user and a testing data set that contains the remaining (NGS-NTGS) samples for testing; assuming the original dataset having NGS samples for every genuine user.

The tested DCNNs are chosen such that their training is relatively fast and possible under the available hardware configuration with 16 GB random-access memory (RAM), Intel core i7-8700K CPU@3.70 GHz and a graphical accelerated processing unit (GPU) of NVIDIA GeForce GTX 1080 Ti with 11 GB RAM. Most of these advanced DCNNs architectures are described in detail in [20].

All the signature images are rescaled to be adapted with each DCNN input size requirement (mostly $224 \times 224$ pixels). Image augmentation is performed during training using random rotation within $[-20^\circ, 20^\circ]$ angle range. In the training experiments, the Stochastic Gradient Descent with Momentum (SGDM) optimization algorithm is used with a fixed learning rate of 0.001 and a mini-batch-size of 100 for a total of 600 epochs. Table 2 displays the resultant accuracy for each DCNN versus the NTGS. The same data are plotted in Fig. 4 for visual comparisons. Table 2 shows that ResNet18 has the best overall performance if we consider

**TABLE 2. The recognition accuracy of the test set versus the number of training samples for different DCNN models.**

| NTGS | ResNet18 | ResNet50 | GoogleNet | NasMobNet | SqueezeNet | ShuffleNet | InceptionV3 |
|------|----------|----------|-----------|-----------|------------|------------|-------------|
| 1    | 94.60%   | 92.63%   | 95.79%    | 70.00%    | 89.61%     | 90.79%     | 88.03%      |
| 2    | 96.80%   | 96.25%   | 97.36%    | 82.50%    | 83.33%     | 93.61%     | 94.58%      |
| 3    | 97.94%   | 97.21%   | 95.44%    | 71.62%    | 95.00%     | 96.76%     | 94.26%      |
| 4    | 99.84%   | 97.66%   | 97.66%    | 89.53%    | 92.66%     | 96.09%     | 96.72%      |
| 5    | 98.83%   | 98.33%   | 98.67%    | 92.50%    | 95.67%     | 97.50%     | 97.33%      |
| 6    | 99.10%   | 98.04%   | 98.75%    | 94.29%    | 92.14%     | 98.04%     | 97.14%      |
| 7    | 99.40%   | 98.08%   | 98.46%    | 95.58%    | 96.35%     | 98.08%     | 97.50%      |
| 8    | 99.80%   | 97.50%   | 98.75%    | 91.04%    | 98.54%     | 98.13%     | 98.33%      |
| 9    | 99.10%   | 97.27%   | 99.32%    | 95.68%    | 97.95%     | 98.64%     | 98.41%      |
| 10   | 99.25%   | 99.00%   | 99.25%    | 96.25%    | 97.00%     | 97.25%     | 97.75%      |
| 11   | 99.45%   | 99.44%   | 99.72%    | 98.06%    | 98.61%     | 99.72%     | 99.72%      |
| **12** | **100.00%** | **99.38%** | **100.00%** | **95.94%** | **96.23%** | **100.00%** | **99.38%** |
| 13   | 100.00%  | 100.00%  | 100.00%   | 88.21%    | 99.64%     | 100.00%    | 100.00%     |
| 14   | 100.00%  | 100.00%  | 100.00%   | 95.83%    | 100.00%    | 99.17%     | 100.00%     |
| 15   | 100.00%  | 100.00%  | 100.00%   | 99.50%    | 100.00%    | 100.00%    | 100.00%     |
| 16   | 100.00%  | 100.00%  | 100.00%   | 100.00%   | 99.38%     | 100.00%    | 100.00%     |
| 17   | 100.00%  | 100.00%  | 100.00%   | 100.00%   | 100.00%    | 100.00%    | 100.00%     |
| 18   | 100.00%  | 100.00%  | 100.00%   | 100.00%   | 100.00%    | 100.00%    | 100.00%     |
| Average | 99.22% | 98.38% | 98.84% | 92.03% | 96.23% | 97.99% | 97.73% |
| STD | 1.37% | 1.82% | 1.40% | 8.74% | 4.31% | 2.43% | 2.94% |
FIGURE 4. The recognition accuracy of the test datasets using different DCNN models versus the number of training samples (NTGS).

FIGURE 4. The recognition accuracy of the test datasets using different DCNN models versus the number of training samples (NTGS).

accuracy (mean and STD) and time because it is the fastest (excluding squeezenet) DCNN net of all the shown models. It also has the highest accuracy (97.94%) at NTGS equals 3.

For instance, at NTGS equals 3, the training and testing times of the most accurate four DCNN networks (ResNet18, ResNet50, Shufflenet, and GoogleNet) are (5.68 min, 2.66 s), (15.73 min, 4.99 s), (8.1 min, 2.94 s), and (8.75 min, 3.17 s) respectively, which emphasizes that ResNet18 is the fastest (excluding squeezenet) and most accurate network in the tested DCNN networks.

We also note that at NTGS equals 12, the accuracy of ResNet18, GoogleNet, and Shufflenet reaches 100%. At NTGS equals 17, all the models give an accuracy of 100%. Hence, according to Table 2, the minimum number of training samples (MNTGS) required to get a 100% accuracy is 12 if using any of the three ready-made DCNNs (ResNet18, GoogleNet, and Shufflenet). Our goal in this research is to minimize MNTGS as much as possible to be acceptable and practical in real applications.

B. USING THE FEATURE EXTRACTOR IN THE TESTING PHASE

In this phase, different layers of the retrained ResNet18 and different distance metrics are evaluated using only three training samples per genuine user to select the best layer and distance type that result in the highest accuracy in the classification of the genuine signatures stored in the testing dataset.

1) TESTING ACCURACY BASED ON FEATURES EXTRACTED FROM DIFFERENT LAYERS OF THE RESNET18 RETRAINED MODEL

In this experiment, all the features of both training (NTGS = 3) and testing (NGS - NTGS) signature images are extracted from a certain layer (L) of the retrained ResNet18 model. Then the cosine distance is calculated between the testing features and the training ones. As stated previously, the test signature is assigned to the genuine ID corresponding to the training signature having the minimum distance from the current test signature. Accordingly, the test accuracy is calculated for the chosen layer (L). By changing L from the 19th layer to the last layer (71st) and calculating the corresponding accuracy, we come out with the Bar-Chart shown in Fig. 5.

From the results shown in Fig. 5, we conclude that the maximum accuracy of 99.85% based on the minimum cosine distance is obtained at layers 66 and 67. This means that by using a k-NN algorithm with k equal to 1 and based on the cosine distance between the testing and training features extracted from higher layers (before the last one), we achieved a very high accuracy of 99.85%, which is higher than the accuracy obtained from the global average-pooling layer, the Softmax layer or the classification layer of the adapted ResNet18, which was 97.94%.

2) TESTING THE BEST DISTANCE TYPE USED IN K-NN TO GET THE HIGHEST ACCURACY

In comparing the different distance types, we repeat the previous experiment using only five layers spaced by 10 layers in the adapted ResNet18, namely layers (27, 37, 47, 57, and 67).

Table 3 displays the accuracy obtained at each layer for each distance type and Fig. 6 depicts a clustered column chart showing the accuracy distribution among the five layers for each distance type.

Table 3 shows that the best distance types are the correlation and the cosine distances because they give the highest averages. We decided to use the cosine distance because it gave higher accuracy (99.85%) at layer 67 than the correlation (99.71%) and its time of execution was faster than...
FIGURE 5. A Bar-Chart showing the percentage of the recognition accuracy versus the layer name and number.
the correlation which includes more calculations due to the subtraction of the mean from all feature values (zero-mean normalization) before performing the dot product.

Regarding the number of FLOPS, for a feature vector of length L, the computational complexity of the cosine distance is 3L, 2L FLOPS for computing the norms of the two feature vectors and L FLOPS for performing the dot product. The correlation distance will require an extra 2L FLOPS for computing the means of the two feature vectors in addition to 2L FLOPS for subtractions. However, based on our hardware and software configuration, the correlation distance calculation time is about 1.4 times the calculation time of the cosine distance.

V. USING AN ENSEMBLE OF MODELS TRAINED AT DIFFERENT ROTATION ANGLES

Given the difficulty of reaching an optimal method for angle-normalization, we decided to perform rotation to the signature training samples with different angles and produce a multi-angle dataset (m datasets) where each dataset instance is generated at a certain angle with a 5° difference between the instances of the rotated datasets. By training m DCNN models on the m datasets, we build an ensemble of m models that will address and tackle the problem of angle variation within the samples of most genuine signatures if aggregated properly.

In the testing phase, the training features produced by the m models at the best layer (66) are concatenated in a single feature row for each training sample of the NTGS samples of each genuine user. When a new test signature is inputted, it will be converted into a bitmap image and transformed by the m models into a single feature row by concatenating the outputs of the m models at the same layer. The distances between the feature row (or vector) of the test signature and each of the stored feature rows (or vectors) corresponding to all the genuine users’ training samples are then computed. The test signature will be assigned the user ID corresponding to the training sample that has the minimum cosine distance using a 1-NN algorithm.

The algorithm for the recognition of (S) test samples given (NGU) genuine users each having (NTGS) training samples and using (m) rotated models will be split into four parts that are run in the following sequence:

- **PART1- COMPOSING THE TRAINING FEATURES**
  For \( g = 1 \) to \( \text{NGU} \)
  For \( ntgs = 1 \) to \( \text{NTGS} \)
  \( \text{FetT}((g-1) \times \text{NTGS} + ntgs) = \text{NULL} \)
  For \( mdl = 1 \) to \( m \)
  \( \text{Fetm} = \text{getfeature (model(mdl), layer = 66, TrainImage ((g-1) \times \text{NTGS} + ntgs)}) \);
  \( \text{FetT}((g-1) \times \text{NTGS} + ntgs) = \text{concatenate (FetT((g-1) \times \text{NTGS} + ntgs), Fetm)} //row-wise concatenation \)
  End mdl
  End ntgs
  End g

- **PART2- COMPOSING THE TESTING FEATURES**
  For \( tst = 1 \) to \( S \)
  \( \text{FetTs(tst)} = \text{NULL} \)
  For \( mdl = 1 \) to \( m \)
  \( \text{Fetm} = \text{getfeature (model(mdl), layer, TestImage(tst))}; \)
  \( \text{FetTs(tst)} = \text{concatenate (FetTs(tst), Fetm)} \)
  End mdl
  End tst

- **PART3-COMPUTING THE COSINE DISTANCE MATRIX BETWEEN THE TEST FEATURES AND THE TRAINING FEATURES.**
  From Table 1, to compute the cosine distance, first compute the norms as follows:
  \[
  \text{normT}(i) = \sqrt{\text{FetT}(i) \cdot \text{FetT}(i)} \quad i = 1: \text{NTGS} \times \text{NGU} \]  
  \[
  \text{MATLAB implementation:} \quad \text{normT} = \sqrt{\text{sum}((\text{FetT}.)^2));} 
  \]
  \[
  \text{normTs}(j) = \sqrt{\text{FetTs}(j) \cdot \text{FetTs}(j)} \quad j = 1: S \]  
  \[
  \text{MATLAB implementation:} \quad \text{normTs} = \sqrt{\text{sum}((\text{FetTs})^2));} 
  \]
  Then compute the cosine distances as follows:
  \[
  \text{CSDIST (j,i)} = 1 - \frac{(\text{FetTs}(j) \cdot \text{FetT}(i)^T)}{(\text{normTs}(j) \times \text{normT}(i)))} \quad j = 1: S, i = 1: \text{NTGS} \times \text{NGU} \]  

| Distance Type | Layer Number | Average |
|---------------|--------------|---------|
| Cityblock     | 84.559%      | 94.853% | 97.353% | 99.412% | 98.529% | 94.941% |
| Chebyshev     | 2.353%       | 51.176% | 55.588% | 91.765% | 87.647% | 57.706% |
| Correlation   | 90.294%      | 96.471% | 97.500% | 99.559% | 99.706% | 96.706% |
| Cosine        | 90.588%      | 96.029% | 97.500% | 99.559% | 99.853% | 96.706% |
| Euclidean     | 84.706%      | 95.441% | 96.765% | 99.559% | 97.941% | 94.882% |
| Spearman      | 2.500%       | 94.265% | 2.500%  | 99.265% | 2.500%  | 40.206% |

TABLE 3. The accuracy obtained with different distance types at layers 27, 37, 47, 57 and 67.
MATLAB implementation:

\[
\text{CSDIST} = 1 - \frac{(\text{FetTs} - \text{FetT'})}{(\text{normTs} + \text{normT'})};
\]

**PART4- DETERMINING THE UNKNOWN CLASS**

For a single test signature (j), the best training sample index \( ti(j) \) will be given by the following formula:

\[
ti(j) = \arg \min_i \text{CSDIST}(j,i) \quad j = 1:S, \quad i = 1: \text{NTGS} \times \text{NG}
\]

(12)

MATLAB implementation:

\[
\text{For all the S test signatures, the best training samples indices will be stored in the vector TIS as follows: } \\
\text{[~.TIS] = min(CSDIST,[],2);}
\]

Assuming the first class id is 0 and the arguments starts from zero, then the ID corresponding to each test signature (j) will be:

\[
\text{ID}(j) = \text{Integer}(ti(j)/\text{NTGS}) \\ j = 1:S
\]

(13)

MATLAB implementation:

The test signatures’ corresponding genuine identification numbers will be stored in the IDS vector using the following MATLAB assignment statement:

\[
\text{IDS} = \text{floor}((\text{TIS}-1)/\text{NTGS}) + 1; \quad / \text{first class’s index is 1}
\]

In testing the proposed ensemble method, odd values of \( m \) were tried to allow symmetry around the original non-rotated versions of the genuine training signatures starting from 1 up to 7 which corresponds to rotation angles from \(-15^\circ\) to \(+15^\circ\). The number of training genuine signatures NTGS has been varied from 1 to 3. The datasets used in testing are TASK1, TASK2, and our new dataset.

We collected a new signature dataset that contained only genuine signatures from 120 students studying in the Faculty of Computing and Information Technology (FCIT) termed (FCITSig). Each student provided 20 signature samples naturally in one or two sessions based on his availability in the university’s campus that was restricted, given the COVID-19 pandemic. The signatures were captured by a Wacom STU540 tablet and the signals (x, y, p, Angle (Azimuth, Altitude), timestamp, and pin-up-down state) were recorded in the FCITSig dataset published online in [21]. The Wacom STU540 tablet has a surface size of 162 mm \( \times \) 8 mm with a resolution of 540 pixels. The maximum sampling rate is 200 points/s.

When training each of the \( m \) models, angle augmentation is performed in the range of \([-3^\circ \, 3^\circ]\) to get continuous angle augmentation when the selected models are aggregated. The training parameters are the same as mentioned in Section IV-3, except that the mini-batch-size is 120. Tables 4-6 display the recognition accuracy versus \( m \) and NTGS for the three datasets TASK1, TASK2, and FCITSig, respectively.

The recognition accuracy using the original ResNet18 as a classifier is also shown in the same tables to highlight the enhancement gained by using the ResNet18 as a feature extractor and using the 1-NN with cosine distance as a classifier.

From Tables 4-6, we found that:

- Three models trained on samples rotated with angles \((-5^\circ, 0^\circ, 5^\circ)\) are sufficient to give 100% accuracy using only three training samples from every genuine user (NTGS = 3) for TASK1 while five models trained on samples rotated with angles \((-10^\circ, -5^\circ, 0^\circ, 5^\circ, 10^\circ)\) are required for TASK2 and FCITSig.

- Using a single sample from each genuine signature yielded an accuracy of 99.08%, 99.58%, and 97.63% for TASK1, TASK2, and FCITSig, respectively. Since our system uses m ResNet18 models and k-NN(k = 1) based on cosine distance we call it RES1NNCOS(m) where \( m \) can be tuned for different datasets to achieve the best accuracy.

Fig. 7 displays the recognition accuracy for the three datasets versus the number of training samples (NTGS) for different numbers of models (m = 1 : 7) and the original ResNet18. From Fig. 7, the enhancement gained by using ResNet18 as a feature extractor is very clear, especially in the case of a single training sample, where an enhancement of 4.41%, 5.01%, and 6.32% is gained in the case of a single model for TASK1, TASK2, and FCITSig datasets, respectively. Also, from the performed experiments on the three datasets, an accuracy of 100% is achieved by three training samples while the 100% accuracy was achieved with 12, 10, and 12 samples using the ResNet18 as a mere classifier for TASK1, TASK2, and FCITSig datasets, respectively.

### TABLE 4. Recognition accuracy of TASK1 versus m and NTGS and ResNet18 accuracy versus NTGS.

| NTGS | m→ | 1 | 3 | 5 | 7 | ResNet18 |
|------|-----|---|---|---|---|---------|
| 3    | 99.85% | 100% | 100% | 100% | 98.68% |
| 2    | 99.03% | 99.03% | 99.31% | 99.58% | 97.42% |
| 1    | 98.42% | 98.95% | 98.95% | 99.06% | 94.01% |

### TABLE 5. Recognition accuracy of TASK2 versus m and NTGS and ResNet18 accuracy versus NTGS.

| NTGS | m→ | 1 | 3 | 5 | 7 | ResNet18 |
|------|-----|---|---|---|---|---------|
| 3    | 99.71% | 99.85% | 100% | 100% | 98.8% |
| 2    | 99.31% | 99.72% | 99.72% | 99.72% | 98.3% |
| 1    | 98.42% | 99.21% | 99.21% | 99.34% | 93.4% |

### TABLE 6. Recognition accuracy of FCITSig versus m and NTGS and ResNet18 accuracy versus NTGS.

| NTGS | m→ | 1 | 3 | 5 | 7 | ResNet18 |
|------|-----|---|---|---|---|---------|
| 3    | 99.51% | 99.85% | 100% | 100% | 98.38% |
| 2    | 98.89% | 99.44% | 99.54% | 99.35% | 96.66% |
| 1    | 95.09% | 97.15% | 97.41% | 97.63% | 88.77% |
A fourth experiment was conducted after merging the three datasets into one dataset with 200 genuine users each having 20 genuine signature samples. Table 7 displays the recognition accuracy versus m and NTGS for the merged dataset. From Table 7, it appears that using three models at NTGS = 3, the accuracy of the system is 99.88% although different datasets are merged that were captured from a variety of users from different nations with no restrictions imposed on the signature lengths, the number of strokes or the direction of writing.

The improvement gained by the proposed technique is still very high, especially when using one training sample (NTGS = 1) where a 7.4% increase of the accuracy is gained when using seven models compared to the original ResNet18 accuracy.

**VI. TECHNIQUE ENHANCEMENTS AND EVALUATION**

This Section considers two possible methods to enhance the proposed technique by testing the effect of changing the angle augmentation range parameter and testing another augmentation method, specifically shear augmentation. We then test the robustness of the system when enrolling new users. The testing time complexity is also discussed, and an approximation formula for the inference time is proposed based on the recorded times while performing our experiments.

By adding shear augmentation in the range of $[-6^\circ + 6^\circ]$ at the time of the network training, we gained the results shown in Table 8 for TASK1.

From Table 8, it is apparent that a 100% accuracy is achieved at NTGS = 2 and m = 3 instead of at NTGS = 3 and m = 3 in Table 4. Moreover, the accuracy is increased at m = 7 to 99.74% for NTGS = 1, instead of 99.08 shown in Table 4.

Another way of enhancement has been achieved by changing the augmentation angle range during model training. For example, when changing the augmentation angle to $[-5^\circ + 5^\circ]$ and using a shear augmentation range of $[-6^\circ + 6^\circ]$, we achieved the following results for TASK2, as shown in Table 9.

From Table 9, we observe that we achieved 100% at m = 3 and NTGS = 3 instead of 99.85% shown in Table 5. When testing this change on TASK1, we didn’t get the
same enhancement. Hence, the augmentation angle can be added as a second argument of the RES1NNCOS technique when presenting the technique’s final results. In other words, for TASK2, we achieved 100% accuracy using RES1NNCOS (3, 5).

A final experiment was conducted to test the robustness of the trained feature extractor DCNN models when new genuine users were enrolled in the recognition system. This was accomplished by testing each data set using the trained models of one of the other two datasets. In this experiment, we tested the case when only three models are generated (m = 3) and three training samples (NTGS = 3) are used. Table 10 shows the results of this experiment as a 3 × 3 matrix where the column represents the dataset used in generating the trained models while the row represents the tested dataset.

| TESTED TRAINING | TASK1 | TASK2 | FCITSig |
|----------------|-------|-------|---------|
| TASK1          | 100%  | 99.85%| 99.71%  |
| TASK2          | 99.56%| 100%  | 99.56%  |
| FCITSig        | 99.26%| 99.22%| 99.85%  |

From Table 10, we observe that adding new users without retraining the extractor models results in an accuracy that remains higher than 99.22% and may reach 99.85%, such as in the case of using the models of TASK2 to produce the training and testing features of TASK1. This proves that the trained extractor models have a high generalization property which allows them to perform very well for new users’s unseen signatures even when their number exceeds 100 new users. In other words, this test proves that the produced models are highly robust and highly accurate even with new datasets. Hence, in the practical use of the technique, when a new user is added, only the feature vectors of his/her reference signatures are generated using the current trained models and added to the database of the 1-NN feature vector database. As such, retraining of the system can be performed regularly, spaced with long intervals of time based on the frequency of enrolling new users.

If we consider the time complexity of our technique, we are aware of the testing time since it is needed at run time while the training time is needed offline.

The testing (or recognition) time is based on the last three parts defined in the recognition algorithm in Section V. The first part (2nd part in the recognition algorithm) is the time used in the extraction of the features of the test signature through the ResNet18 network which is proportional to the image dimensions (224 × 224). Hence, the number of FLOPS of this part is approximately the same as the number recorded for the ResNet18 (1.82 G FLOPS). The number of genuine users has no effect on the test feature extraction time because the fully connected layer and the Softmax layer are not used in the extraction of the features. The second part (3rd part in the recognition algorithm) concerns the calculation of the cosine distance between the unknown feature vector (FV), and all the genuine users’ training FVs, which is linearly proportional to the feature vector length (FVL) multiplied by the number of genuine users and the number of training samples per genuine user (NTGS × NGU × FVL). Since the FVL is proportional to the image size (because layer 66’s feature maps are the fifth down sampled version of the image (224 × 224/(2^5) × 2^5) = 7 × 7)) and the number of channels of layer 66 (512), hence the first two parts have a time complexity linearly proportional to the image size. The third part (4th part in the algorithm) concerns the search time for the minimum distance which requires a time complexity proportional to NTGS × NGU. Hence, in terms of the number of FLOPS, the time complexity of the used 1-NN (3rd and 4th parts of the recognition algorithm) is (NTGS×NGU) × FVL + NTGS × NGU = NTGS × NGU × (FL + 1) = 25089 × (NTGS × NGU) ≈ 25 × NTGS × NGU K FLOPS. By neglecting the search time (4th part of the recognition algorithm), the proposed technique’s inference algorithm will have a time complexity of O(d) where d = I_w × I_h. The time taken in the extraction of the training FVs and the calculation of their norms should be added to the technique’s training time. Based on the used hardware and software in the experiments conducted in this research, the time component of the feature extractor is about 3.9 ms for m = 1 for a single test signature which is constant regardless of the number of genuine users since the extraction is carried out before the global average-pooling layer. The time component of the cosine distance based 1-NN is about 64.2 microseconds (without using GPUs) for a single test signature (in the case of 40 users). Hence, the time taken in the recognition of a single test signature (Trecog) can be approximated (without considering the time used in the conversion of the time series signals into 2-D image) in milliseconds as follows:

\[ Trecog \approx 3.9 \times m + 0.0642 \times m \times NTGS \times NGU / 40 \text{ ms} \]

(14)

In the case of m = 3, NTGS = 3 and NGU = 40

Trecog = 12.2778 ms ≈ 12.3 ms.

VII. COMPARISON WITH STATE-OF-THE-ART RESULTS

This section compares our technique with other techniques based on the dataset used, identification accuracy, and the NTGS. Since our proposed technique assigns the unknown signature to only one genuine user, hence the produced false rejection rate will be equal to the false acceptance rate which will result in an EER for random forgery equal to 100 minus the recognition accuracy. In this manner, verification systems that calculate the EER for random forgery can be relatively compared with our recognition system. Table 11 displays a comparison between the state-of-the-art techniques and our proposed technique based on NTGS and the identification accuracy for work performed on TASK1 and TASK2.
Table 11 shows that our technique outperforms all the signature identification techniques based on both accuracy and NTGS.

If we consider techniques that did not mention the recognition accuracy explicitly, we can have the following comparison in Table 12, where we consider our technique in case of using 1 genuine training sample (NTGS = 1).

Therefore, from Table 12, we can conclude the following:

- Our technique outperforms the hybrid statistical and deep learning technique [13] with about a 2.35% increase in accuracy although not all genuine signatures are considered in the random forgery test in [13]. Regarding time complexity, assuming the same problem configuration and according to [22], CNN requires a smaller number of FLOPs than LSTM. However, both CNN and LSTM are used in [13](although using an enhanced configuration of the 1-D CNN module) in addition to a feature extraction module which is composed of a convolutional autoencoder part and a cluster-based feature selection part. Another point is that the architecture in [13] compares two input signatures for the purpose of verification and to be used without change for the recognition task it will require multiplication of the inference time by the number of genuine users; otherwise, the LSTM should be reconfigured and retrained for multi-user-classification. Reconfiguration of the technique in [13] for multi-user classification is not a simple task since [13] uses writer-dependent features while our technique is writer-independent.

- Our technique without using human-like generated duplicate samples outperformed the state-of-the-art accuracy achieved in [10] by 1.24% in the case of TASK1 and slightly less by 0.16% in the case of TASK2. If we consider the average accuracy of our technique for both TASK1 and TASK2, it will be 99.54% where the state-of-the-art technique [10]'s average accuracy will be 99.0% which indicates the superiority of our proposed technique. It is also worth mentioning that when no-duplicates were used in [10], the best recognition accuracies were 89.5% and 92.5% using DTW-based technique and Hidden Markov-Model-based technique for TASK1 and TASK2, respectively. The time complexity of the DTW technique is $O(d^2)$ which should be multiplied by the number of duplicates and the number of users in case of using it in a recognition system. Given the significant challenges in the signature recognition and verification problems, almost all mentioned techniques have concentrated on the performance of the
technique regarding the recognition accuracy and EER, and none of them has given an approximate value for the time complexity of the whole technique in terms of the number of FLOPS or even given a round figure of the time required in the verification or recognition of the unknown signatures. Giving the order of complexity in terms of the Big O notation is not sufficient for fair comparisons since many factors need to be considered, such as parallelization of the technique and the current support of hardware and software available for such parallelization.

- Further enhancements of the proposed technique may consider augmentation using human-like duplicates either through sigma-lognormal based models or deep autoencoders. A significant criterion of our technique is using the feature extractor models (without retraining) with newly enrolled users while maintaining a high recognition rate (above 99%).

VIII. CONCLUSION

In this research we introduced a novel architecture for signature identification using ResNet18 model as a feature extractor and 1-NN with cosine distance as a classifier yielding a highly accurate technique that outperformed the state-of-the-art techniques in both recognition accuracy and random forgery EER, having the following contributions:

- Although using a deep learning-based technique with very small training samples we have got a very high recognition accuracy.
- Although having a smaller number of layers and less computational time in both training and testing, the residual network was one of the best architectures for handwritten signature recognition.
- The features extracted from deep layers before the global average-pooling layer gave more accurate discrimination distances than that extracted after the pooling layer, the fully-connected layer, or the Softmax layer. Hence it is important to search for the optimum layer in the residual network or any other deep network for extracting the best features with high discriminative distances.
- It was proven empirically that the cosine distance is the most accurate discriminating distance when comparing features extracted from the residual networks for handwritten signatures classification.
- The technique was tested on two public datasets yielding 100% accuracy with only three training samples.
- When using only one sample, the proposed technique offered very high accuracy that outperformed the state-of-the-art techniques without using human-like duplicated samples in the training process.
- Concatenating features extracted from 5 models trained on 5 rotated dataset versions decreased the EER of the 1-NN classifier to zero in the three tested datasets and to 0.15 in the aggregated dataset when using only three genuine training samples.
- The initially generated extractor models are sufficiently robust enough to allow enrolling new genuine users while maintaining the recognition accuracy above 99%. Augmenting the signatures using the sigma-lognormal-based generation model can be considered in future work to get 100% recognition accuracy with training samples less than two and three for TASK1 and TASK2, respectively.

ACKNOWLEDGMENT

The authors acknowledge Prof. Abdullah Basuhail who supervises the Thesis of Mr. Hadi and did the proofreading of this article.

REFERENCES

[1] D.-Y. Yeung, H. Chang, Y. Xiong, S. George, R. Kashi, T. Matsumoto, and G. Rigoll, “SVC2004: First international signature verification competition,” in Proc. ICBA. in Lecture Notes in Computer Science, vol. 3072, Berlin, Germany: Springer, 2004, pp. 16–22, doi: 10.1007/978-3-540- 25948-0_3.
[2] M. Diaz, M. A. Ferre, D. Impedovo, M. I. Malik, G. Pirlo, and R. Plamondon, “A perspective analysis of handwritten signature technology,” ACM Comput. Surv., vol. 51, no. 6, pp. 1–39, Feb. 2019, doi: 10.1145/3274658.
[3] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, “A survey of deep learning and its applications: A new paradigm to machine learning,” Arch. Comput. Methods Eng., vol. 27, no. 4, pp. 1071–1092, Sep. 2020, doi: 10.1007/s11831-019-09344-w.
[4] R. Ghosh, P. Kumar, and P. P. Roy, “A Dempster–Shafer theory based classifier combination for online signature recognition and verification systems,” Int. J. Mach. Learn. Cybern., vol. 10, no. 9, pp. 2467–2482, Sep. 2019.
[5] K. Bibi, S. Naz, and A. Rehman, “Biometric signature authentication using machine learning techniques: Current trends, challenges and opportunities,” Multimedia Tools Appl., vol. 79, nos. 1–2, pp. 289–340, Jan. 2020.
[6] S. Minae, A. Abdolrashidi, H. Su, M. Bennamoun, and D. Zhang, “Biometrics recognition using deep learning: A survey,” 2019, arXiv:1912.00271. [Online]. Available: https://arxiv.org/abs/1912.00271
[7] A. Forouzandehe, A. A. Hemmat, and H. Rabbani, “Online handwritten signature verification and recognition based on dual-tree complex wavelet packet transform,” J. Med. Signals Sens., vol. 10, no. 3, p. 145, 2020.
[8] S. Dutta, R. Saini, P. Kumar, and P. P. Roy, “An efficient approach for recognition and verification of on-line signatures using PSO,” in Proc. 4th IAPR Asian Conf. Pattern Recognit. (ACPRI), Nov. 2017, pp. 882–887.
[9] Y. Jia, L. Huang, and H. Chen, “A two-stage method for online signature verification using shape contexts and function features,” Sensors, vol. 19, no. 8, p. 1808, Apr. 2019.
[10] M. Diaz, A. Fischer, M. A. Ferrer, and R. Plamondon, “Dynamic signature verification system based on one real signature,” IEEE Trans. Cybern., vol. 48, no. 1, pp. 228–239, Jan. 2018.
[11] V. Chandra Sekhar, P. Mukherjee, D. S. Guru, and V. Pulabaigari, “Offline signature verification based on writer specific feature selection and fuzzy similarity measure,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops, Dec. 2019, pp. 88–95.
[12] A. Jain, S. K. Singh, and K. P. Singh, “Handwritten signature verification using shallow convolutional neural network,” Multimedia Tools Appl., vol. 79, pp. 19993–20018, Jul. 2020.
[13] C. S. Vorugunti, V. Pulabaigari, P. Mukherjee, and A. Sharma, “DeepFuseOSV: Online signature verification using hybrid feature fusion and depthwise separable convolution neural network architecture,” IET Biometrics, vol. 9, no. 6, pp. 259–268, Nov. 2020.
[14] W. Rawat and Z. Wang, “Deep convolutional neural networks for image classification: A comprehensive review,” Neural Comput., vol. 29, no. 9, pp. 2352–2449, Sep. 2017.
[15] M. S. Ebrahim and H. K. Abadi, “Study of residual networks for image recognition,” 2018, arXiv:1805.00325. [Online]. Available: http://arxiv.org/abs/1805.00325
[16] G. Abosamra and H. Oqaibi, “An optimized deep residual network with a depth concatenated block for handwritten characters classification,” Comput. Mater. Contin., vol. 68, no. 1, pp. 1–28, 2021, doi: 10.32604/cmc.2021.015318.
[17] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark analysis of representative deep neural network architectures,” IEEE Access, vol. 6, pp. 64270–64277, 2018.

[18] N. Bhatia and Vandana, “Survey of nearest neighbor techniques,” 2010, arXiv:1007.0085. [Online]. Available: http://arxiv.org/abs/1007.0085

[19] L. Jiang, Z. Cai, D. Wang, and S. Jiang, “Survey of improving K-nearest-neighbor for classification,” in Proc. 4th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD), 2007, pp. 679–683.

[20] A. Khan, A. Sohail, U. Zahoor, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” Artif. Intell. Rev., vol. 53, no. 8, pp. 5455–5516, 2019.

[21] H. Oqaibi and G. Abosamra, “FCITSig dataset,” KING Abdulaziz Univ., Jeddah, Saudi Arabia, Tech. Rep., 2021. [Online]. Available: http://www.kau.edu.sa/GetFile.aspx?id=306609&fn=FCITSigDATA.zip

[22] Z. Ye, A. Gilman, Q. Peng, K. Levick, P. Cosman, and L. Milstein, “Comparison of neural network architectures for spectrum sensing,” in Proc. IEEE Globecom Workshops, Dec. 2019, pp. 1–6.

**GIBRAEL ABOSSAMRA** received the B.S., M.S., and Ph.D. degrees in electronics and communication engineering from Cairo University, Egypt. He is currently working as an Associate Professor with the Faculty of Computing and Information Technology, King Abdulaziz University. He has published numerous journal articles. His research interests include machine learning, deep learning, genetic algorithms, pattern recognition, and image processing.

**HADI OQAIBI** received the bachelor’s and master’s degrees in computer science from King Abdulaziz University (KAU), where he is currently pursuing the Ph.D. degree in on-line signature verification and recognition with the Department of Computer Science, Faculty of Computing and Information Technology. He is currently a Lecturer with the Faculty of Computing and Information Technology, KAU.

---

**VOLUME 9, 2021**