[1] Dr. K. Tamilarasi *

[2] Mrs. V. Annapoorani

[3] Mrs. T. Nathiya

Optimized SVM Parameters with Googlenet Model for Handwritten Signature Verification

Journal of Electrical Systems

Abstract: - As far as behavioral biometric authentication procedures go, signature verification is at the top of the list. The most popular form of user authentication, a signature is like a "seal of approval" that confirms the user's permission. The primary objective of this verification technique is to distinguish between real signatures and those that have been forgeried by imposters. In order to learn characteristics from the pre-processed real and fake signatures, Convolutional Neural Networks (CNN) were used in this research. The model utilized to train the CNN was the Inception V1 architecture (GoogleNet). To make the network larger rather than deeper, the design makes advantage of the idea of having various filters on the same level. A small number of publicly accessible datasets, including CEDAR, the BHSig260 signature corpus, as well as UTSig, are used to evaluate the suggested approach in this study.

Keywords: Handwritten Signature, Convolutional Neural Network, Inception V1, CEDAR, BHSig260 signature corpus, and UTSig.

I. Introduction

Despite the widespread use of digital signatures for verification in many fields (e.g., property records, agreements among parties, legal certificates, identity cards, etc.), the usage of handwritten signatures is still prevalent. Because a forgery might have serious consequences for the rightful owner, signature verification is of the utmost importance. Consequently, the ability to identify legitimate signatures is crucial for preventing certain types of fraud [1]. The primary uses for a person's distinctive signature are in establishing their identification and verifying the authenticity of important or legally binding documents. You may verify a signature in two ways: statically and dynamically. User acceptability, required security level, accuracy, cost, and execution are the primary characteristics that must be considered while developing a signature verification system [2]. Currently, biometrics is used extensively for identifying and verifying individuals and their signatures on a global scale. Banks as well as other financial along with legal institutions rely heavily on handwritten signatures as a kind of unique identifying human effort. The historical importance of handwritten signatures as a target of deceit, however, has led to their rising value. Determining whether a particular sign is genuine (produced by the stated individual) or counterfeiting (made by somebody else) is the goal of the Sign Detection Systems (SVS). The usage of images of scanned signatures with supplementary documents lacking dynamic data related to the signing process has proven difficult, especially in offline (static) settings [3]. The application of an individual's handwritten signature is among the most used biometric authentication techniques. Signature verification has emerged as a formidable obstacle in biometric technology, exacerbated by both intra- and inter-class variability. In addition, since it is static, an offline handwritten signature may be faked by trained eyes [4]. In biometrics, human signatures play a significant role. Everyone has a unique signature with their own set of characteristics, making this a useful authentication technique in many industries, but notably banking. Human signatures are therefore used for the purpose of authenticating each individual. The issue of verifying handwritten signatures remains unresolved, despite a significant amount of research in the area. One of the primary goals of signature verification is to identify forgeries. Both online and offline methods may be used to verify the signature. Especially when done offline, when the signature's dynamic information is unavailable, this is a time-consuming and laborious process [5, 26]. One of the most helpful biometric technologies for identifying persons within an organization or financial department is handwritten signature verification. The use of neural networks trained with deep learning to enhance photo classification has opened the door for computer vision to be shown in modern research applications via the use of image processing techniques. In order to eliminate the inefficiencies, mistakes, delays, and hassles associated with manual signature verification, a recognition system for automated signature verification must be developed [6]. When an offline handwritten signature is validated by biometric authentication technology. One of the challenges of signature verification is that signatures are time-variant. Online dynamic signatures are one of two main types of signatures.

[1, 3] Assistant professor, Department of ECE, Excel Engineering College, Namakkal, Tamilnadu. [3] nathiyan959@gmail.com

[2] Assistant professor, Department of ECE, Mahendra Institute of Technology. annapooraniv@mahendratech.org

* Corresponding Author Email: tamilarasiphd.23@gmail.com

One that is not in use at the moment is called a static signature. When an offline signature can't be replicated, even by a trained hand, the term intra-personal variability is employed [7]. To safeguard sensitive information that should only be accessible to those with the proper authentication credentials, it is crucial to safeguard one's authenticity in the modern world. The use of fraudulent methods, such as signature forging, to get access to private data is on the rise these days. A number of approaches have been developed to address the signature verification problem, including Machine Learning and Deep Learning, which can verify the authenticity of a signature [8, 27]. In many tasks, deep learning has recently outperformed both people and more traditional, manual approaches. It is difficult to validate the actual limitations of deep learning due to the low quantity of publicly accessible data for some applications, including verification of handwritten signatures. The use of different databases and experimental techniques makes it difficult to evaluate the improvements of new suggested approaches, and there is also a lack of publicly accessible data [9]. Machine learning has seen a dramatic shift due to the fast development of deep neural networks, which has allowed for outstanding progress in many areas [10]. Forensic as well as commercial transactions have been among the most recent to make extensive use of signature verification technologies to confirm the identity of signatories. In most cases, the reliability of the system's authentication is greatly affected by the feature extraction and categorization processes. Signature verification systems have a significant challenge when it comes to feature extraction because of the wide variety of signature types and sample conditions. In terms of distinguishing between real and fake signatures, current signature verification methods show encouraging results. But expert forgery detection as a whole is still not flexible enough to provide satisfactory results. In addition, in order to improve the accuracy of signature verification, the majority of existing methods need a huge number of learning instances [11, 28]. One of the best behavioral biometrics is a handwritten signature. When it comes to controlling access to financial services, conducting criminal investigations, providing legal assistance, etc., it is crucial for identifying and verifying individuals. Because of its prominent usage, the handwritten signature is vulnerable to abuse. To lessen the possibility of signature abuse, verification methods based on deep learning are rapidly gaining popularity. To determine whether a signer is who they say they are or if their signature is fake, signature verification relies on pairwise restrictions [12]. In order to verify signatures or complete other activities, it is now common practice to need to know where a signature appears in documents. While manual processing may be feasible for relatively moderate document loads, it would need substantial personnel and supplies to handle very large document loads.When trying to intelligently find signatures using deep learning approaches, there are a few obstacles: First of all, the signature is often obscured or otherwise interfered with in Chinese papers like files and contracts, which makes it difficult to see the signature. Secondly, real-world contracts and invoices that may serve as training sets are not easy to come by. Third, intelligent locating poses certain challenges due to the signature's lack of characteristics [13, 29]. The literature on handwritten signature verification includes a number of synthetic databases that were created for data-augmentation purposes. These databases used various deep learning methods, bio-inspired methods, neuromotor synthesizers, and Generative Adversarial Networks to generate new specimens and identities. For the purpose of training and developing signature verification systems, these synthetic databases include synthetic specimens of real and forged signatures [14]. The scientific community is divided on the best way to validate the authenticity of handwritten signatures, which are considered biometrics. Many researchers have concentrated on expanding the use of systems based on the analysis and processing of handwritten signatures to new domains, with the past decade seeing significant advancements in the technology's application in management, finance, handling cases, and security [15].

II. Related Work

The signature is recognized using the deep learning approach in the work of [16] because to its excellent accuracy and lack of preprocessing requirements. The most typical uses for a CNN-based deep learning model are image processing, classification, as well as segmentation. Surpassing other algorithms like KNN and SVM, CNN learns more. In order to enhance categorization, CNN is used in that work. In order to verify digital signatures online, the researchers [17] created a CNN-based machine learning approach. To extract distinguishing characteristics across several scales, convolutional kernels with different values, such as 1×1, 3×3, and 5×5, are used. In order to generate more reliable features, the CNN combines information from the first and intermediate layers. Various geographically linked data from convolutional layers has been mixed using upsampling with bilinear interpolation. The convolutional features have been aggregated using both addition and concatenation approaches. To increase the depth of the convolutional layers, they employ a convolutional transposition approach to construct layers of varying depths. The fully linked layers then use the combined features for categorization and high-level feature extraction. The evaluation of the suggested method included the creation of a bespoke database, its deployment in an android app, and the collection of 985 digital signatures from 197 participants. Thanks to the data augmentation approach, online signature verification no longer needs to deal with insufficient training data. On the custom-built online signature database, the experimental findings reveal that the deep aggregates convolutional characteristics modeling approach achieves an accuracy of 99.32%. Utilizing the Grupo de Procesado Online de Seales, researchers offer a DL-based automated recognition approach in [18]. The synthetic signature dataset, the largest publicly available database of handwritten signatures, was used to classify the signatures of one hundred individuals, with thirty bogus and twenty legitimate signatures per person. The addition of layers such as dense (1024), dense (1), dropouts (0.5),and flatten refines that model. Its foundation is a learning transfer (TL) paradigm from Inception V3 that was fine-tuned from its intermediate architecture. Six well-known TL convolutional neural network designs—EfficientNet, MobileNet, ResNet 101, ResNet 50, and VGG 19—were competitors of the suggested model. The suggested model performs better than the pre-trained models. With similar values of 88%, 88%, and 87% in sensitivity, F1-score, and accuracy, the model likewise outperformed expectations in these areas. For the previously trained models, the corresponding ratings were 80%, 81%, 73%, 77%, and 74%. With an accuracy of 88%, the proposed fine-tuned inception V3 classifies signatures as either real or fraudulent, producing the most precise results. The results of this study will pave the way for an improved network method to be created for offline signature verification using computer vision. I. In this study, we look at the most recent deep learning methods for verifying signatures online and analyze them in detail. ii. It presents and details DeepSignDB, a brand-new public databases for online handwritten signing biometrics. iii. It offers a baseline and a standard experimental procedure that scientists may employ to properly evaluate novel techniques to the state-of-the-art. iv. Time-Aligned Recurrent Neural Networks (TA-RNNs), a new deep learning technique for online handwritten signature verification, is adopted and analysed. This method trains systems to be less susceptible to forgeries by combining Dynamic Time Warping and Recurrent Neural Networks. Their proposed TA-RNN system outperforms the state-of-the-art, achieving outcomes with an EER lower than 2% even when each user just uses one training signature. The paper presents a CNN-based deep learning template for signature verification. [20]. For the sake of that experiment, they retrained the classification layer using back propagation and the idea of transfer learning, and they utilized the feature extraction part of the GoogleNet model to calculate transfer values. The Deep learning model's classification layer underwent retraining using a dataset of 25 classes of trademark images, with 85 signatures per class. A testing dataset consisting of fifteen signatures from every class was used to assess the model's performance after training. The neural network design tested with the signature dataset had an average testing accuracy of 95.2%. Researchers analyze NeuroWrite's data preparation techniques, network architecture, and training methodologies in detail in [21]. On handwritten digit dataset like MNIST, they demonstrate how NeuroWrite may accomplish strong generalization and excellent classification accuracy by applying state-of-the-art approaches. Researchers also investigate possible practical uses of the model, such as automatic postal code identification, signature verification, and digit recognition in digital documents. The versatility and efficiency of NeuroWrite make it a valuable resource for computer vision as well as pattern recognition tasks. That paper provides a comprehensive overview of NeuroWrite, including its architecture, training technique, and evaluation metrics. It shows how NeuroWrite may enhance many applications that need handwritten digit categorization. The results prove that NeuroWrite is a viable method for enhancing deep neural network-based handwritten digit recognition. Preprocessing, multifeature combination, discriminant feature selection via a genetic algorithm utilizing a one-class support vector machine, as well as a one-class learning approach to handle imbalanced signature data are the four main stages of an everyday signature verification system that the paper outlines to address the aforementioned problems. Three signature databases—UTSIG, CEDAR, and SID-Arabic—are used by the proposed technique. In comparison to existing systems, the suggested method achieves better results in terms of error rates, false approval rates, along with false rejection rates, according to the experimental results. Using an extensive set of signature images, the research contrasts and compares many deep learning systems that make use of the Siamese architecture [23]. They use publically available datasets, including the BH-Sig260 signature corpus, the ICDAR 2011 SigComp, the Handwritten Signature database via Kaggle, and CEDAR, to train the models. A sequence of classifiers, such as K-Nearest Neighbours, Gaussian Naïve Bayes, Logistic Regression, and Support Vector Classification computations, are employed to evaluate if the signature is real or fraudulent. The aim of the study in [24] is to increase the accuracy of offline signature recognition. To improve accuracy, the suggested technique uses histogram orientation gradient in addition to signature length normalization. In order to develop the reference structure for future predictions, a CNN-based deep-learning approach is used for verification purposes. The experiments are carried out using a total of 2,000 expertly fabricated signature samples, obtained from 200 different persons, along with 4,000 authentic samples. Individuals in Malaysia have access to that database via its public distribution under the SIGMA name. The experimental data are presented in two types of error: the False Accept Rate, which reached 4.15%, and the False Reject Rates, which reached 1.65%. The total success rate is as high as 97.1%. When compared to the state-of-the-art study that used the same SIGMA database, the suggested method performs better. Their objective in [25] problem is to build a system that uses CNNs and deep learning to determine whether the signature is legitimate or false from a collection of signatures. Signatures fluctuate throughout time due to various behavioral changes, such as age, mental condition, physical health, etc., which is why they utilize CNN and deep learning. Authors need an apparatus that can improve its detection accuracy by learning from various training datasets. Both offline and online signature verification technologies are available for use in authentication processes. The offline signature forgery detection approach is the foundation of their project. These signatures must be captured digitally since they are handwritten on official papers. For that reason, image processing should be considered with that project. In order to improve the project's accuracy, they want to use the offline approaches outlined in a few of articles that deal with signature forgery detection using deep learning models, in addition to the online methods.

III. Proposed Work

3.1 Architecture Overview

Learned features are used to train a writer-independent classifier in this article. The process begins with extracting features from signatures and creating a feature set. The model is then evaluated using test data. This study use an CNN to learn writer-independent attributes for every user. A convolutional neural network was trained to take advantage of the fact that there are significant differences between the fake and real signatures. There are two categories here: authentic signatures and counterfeit ones. In order to prepare a feature space for a handwritten signature, it is necessary to extract its intrinsic attributes. Therefore, utilizing this feature space, this represents the attributes of each user, the train classifier for both classes with the goal of discovering the distinctive distinctions. As seen in Figure 3, we take into account two preprocessed pictures of the signature and subsequently use an architecture that draws inspiration from the Google Net (Inception V1) architecture.

The characteristics for handwritten signature verification are computed in this study using a revolutionary deep learning technique.

Pseudo-code for training our proposed method

1. Preparing the dataset for analysis.
2. Separating the datasets into subsets of test and train data.
3. Feature generator network training

a. Utilising a source task to train the feature generator network;

b. Putting the trained network to the test to evaluate its performance.

c. Applying the network's weights to a specific job, with or without fine-tuning.

4. Classifier Training:

Making a support vector machine (SVM) for every user class is very important.

- For every kind of signature, create a random collection of forgeries.
- Train the SVM using both the user's actual signature and a pool of randomly generated fake signatures.

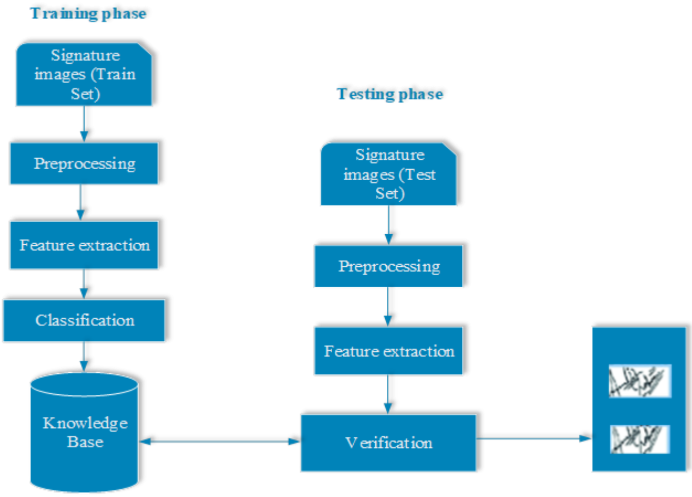


Fig. 1. Proposed Model

3.2 *Image Preprocessing*

Since the signatures used in our tests were previously derived from the original papers, this research does not go into the process of signature extraction. Signatures made by different people will always have some inherent differences in size, pen its thickness, rotation, as well as translation, therefore pre-processing the signature photos is a must. Handshakes, unequal forces, and other outside influences disrupt the signature curve during data collection. Therefore, the signature curve undergoes a pre-processing phase to eliminate noise and smooth it out before feature extraction techniques are used. In GPDS, the inputs to the deep learning network span from tiny signatures measuring 153×258 pixels to huge signatures measuring 819×1137 pixels, with a constant size that allows for substantial form variation in the signatures.

The goal of this step is to rotate a signature picture so that many copies may be made. The sources as well as noised signature pictures were subjected to geometric alteration, which resulted in rotation at certain angles. The angles of spin are 30 degrees, -30 degrees, 20 degrees, -15 degrees, and -15 degrees. The angles chosen to represent the many rotations that may occur while scanning or signing a document. Fig.1 is represent the Proposed Model.

3.3 *Feature Extraction using Inception V1*

Problems requiring categorization, such as signatures, often make use of Convolutional Neural Networks. This situation is well-suited to the CNN architecture because to its reduced amount of trainable parameters. In this case, CNN architecture works well since we may compress the signature images beyond a certain point without losing any of the signature details. The features are obtained locally in an overlapped grid of patches and integrated in non-linear ways in succeeding layers. This architectural type also has certain traits with handmade feature extractors utilized in the literature. Improving CNN classifiers was a major goal of Inception V1. To demonstrate hints of improved performance before its launch, the most popular CNNs just piled convolution layers more and more deeply. Compared to earlier networks, this one is "widers" rather than "deeper" since it uses many concurrent filters on the same level. The writers made sure that the inception module mirrored this. The "naive" inception module is shown below Figure 6. On the same level, three different filter sizes (1×1, 3×3, and 5×5) are used for convolution, in conjunction with maximum pooling. This broader layer's outputs are combined and used as inputs by the subsequent layer. Fig.2. is depicted in Inception Model.

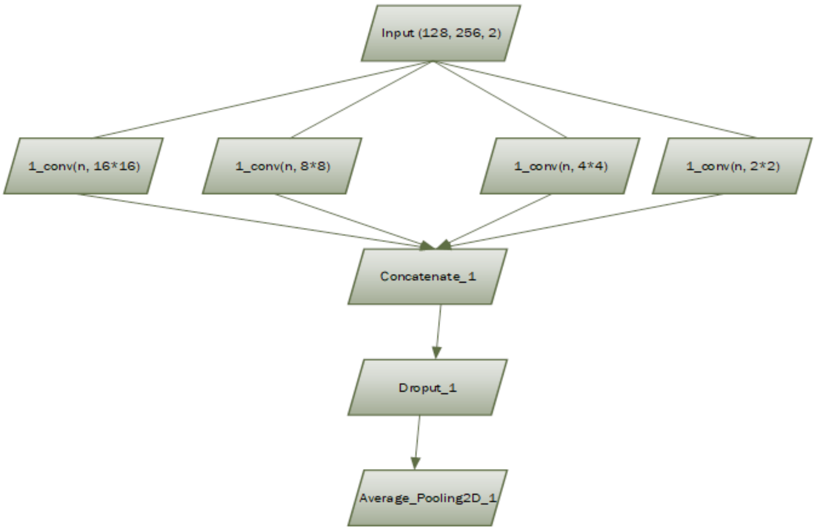


Fig. 2. Inception Model

In 2014, Google researchers presented the design of an advanced convolutional neural network called GoogLeNet (Inception V1). Its cutting-edge detection and classification capabilities helped it win the 2014 ImageNet Large-Scale Visual Recognizing Competition (ILSVRC14). It contains 22 layers total, with 9 of those layers consisting of 'inception modules' that boost computing performance by reducing parameters. Numerous parallel convolutional layers with varying sizes (1×1, 3×3, and 5×5) and pooling processes make up these inception modules. This enables the network to effectively extract global and local properties while capturing data at different sizes. There won't be any computing issues while using a big number of filters. To limit computational complexity and minimize input dimensionality, the design additionally uses a 1 × 1 convolutional layer prior the bigger convolutions. As a result, deeper network designs are possible with fewer parameters. Dropouts along with L2 weight decay are two of the many methods used by GoogLeNet to prevent overfitting. In sum, GoogLeNet is a very effective and precise deep learning architecture; its popularity has prompted the creation of similar frameworks like ResNet. It should be mentioned that GoogLeNet was designed to excel on more complex and expansive datasets like ImageNet. The database utilized for this study contains color images of 1,000 distinct object classes, making it much more visually complicated than the one used. So, it will be interesting to see how GoogLeNet does on this particular dataset.

Feature extraction is a method that takes raw data and uses it to extract useful information in the form of feature vectors. These vectors may then be utilised for processing purposes instead of the raw data itself. This work makes use of a transfer learning model, namely GoogleNet, also known as Inception V1.

One common machine learning technique is transfer learning, which involves taking an existing model and applying it to a new problem. It is capable of applying previously learned material to novel contexts. Transfer learning is a technique in deep learning that involves building an innovative network structure utilizing networks that have already been trained on a big dataset. This new architecture may then be fine-tuned and applied to a different dataset. Given the massive resources needed to train the deep CNN, this generally works out quicker and simpler than constructing and instructing a network from the ground up, and it drastically cuts down on training effort.

The ImageNet competition is a big and difficult picture classification problem, and many deep learning models are trained for it before being utilized for transfer learning. This article makes use of AlexNet architectures that have already been trained. To use AlexNet's well-learned features to a different picture classification job, one removes the network's tail and substitutes it with a different classifier—softmax, for instance—to meet the requirements of the new assignment. The freshly constructed transfer network retains the same structure as the network that was previously trained with the exception of the last two or three layers. Training the newly built network involves keeping weights in the transfer layers frozen. This allows us to apply what we've learnt from the pre-trained datasets to the novel problem at hand.

It accepts H as an input to this portion. The size of the convolution that is frequently employed in the classification of texts is n×w, where w is the width of the convolution filter. Two consecutive convolution procedures with sizes of n×1 and 1×w, respectively, will be used to factorize it in this study. By reducing the network's parameters, factorization alleviates over-fitting and increases the network's depth, which allows for the extraction of nonlinear features compared to simple convolution. What follows is a more in-depth explanation of an asymmetric convolution layer, where the second convolution differs from.

An asymmetric convolution layer's first convolution with $n \times 1$ size filter $W_j^1 \in \mathbb{R}^{n\times1\times2}$ The input patch H_i, representing the i-th slice throughout periods step dimensions (H(:,i,:)), is subjected to the following operation in order to create a matching local feature:

$$f_{ij}^1 = \max\left(W_j^1 * H_i + b, 0\right) \tag{1}$$

Where subscript $\in \{1, \cdots, m\}$, where b is a bias term, is employed to index the feature map's channels and superscript 1 indicates the initial convolution. The nonlinear activating ReLU is selected. The feature maps that were subsequently obtained are $f^1 \in \mathbb{R}^{1\times l\times m}$ that fulfills

$$f^1(1,:,:) = \left(f_{ij}^1\right)_{l\times m} \tag{2}$$

The asymmetric convolution layer's second convolution, which is of size 1×w and uses a filter, $W_k^2 \in \mathbb{R}^{1\times w\times m}$ It is utilized to convolve with a series of slices, designated as F_i, using a timestamp centered at point i.

$$F_i = \left(f_{i-m/2}^1, \cdots, f_{i-1}^1, f_i^1, f_{i+1}^1, \cdots, f_{i+m/2}^1\right) \in \mathbb{R}^{1\times w\times m}$$

where $f_i{}^1 = f^1(:, i, :)$

The following is the process for producing a local feature when there is no tensor having the same form and i∉{1,⋯,l}:

$$f_{ik}^2 = \max(W_k^2 * F_i + b, 0) \tag{3}$$

where subscript $k \in \{1, \cdots, m\}$, where is used to index the feature map's channels and "2" signifies the second convolution; all other details remain same from the previous description. Thus, tensors constitute the feature maps used in the second stage $f^2 \in \mathbb{R}^{1\times l\times m}$.

3.4 SVM Classification

When it comes to handwritten signature categorization, integrating the characteristics of distinct models may harness the power of numerous pre-trained networks. This approach allows for the concatenation of many pre-trained models, which collect and integrate relevant features before feeding them into an SVM classifier. The result is a final classed output. Here, the SVM classifier receives a mixed set of features taken from the pre-trained networks. Subsequently, the output of the SVM classifier indicates which classes the input signatures belong to.

A popular supervised learning model in machine learning, support vector machines (SVMs) may classify incoming data points according to pre-existing training examples. It uses n-dimensional space having the greatest margin between classes to map data points spatially and attempt to separate every information point of various classes, wherein n is the total amount of features. The hyperplane divides the area into portions, and the samples fit into one of those sections. So, we can tell what group the sample belongs to.

Finding the parameters that minimise the generalisation error or maximise the accurate classification rate is the goal of parameters optimisation for support vector machines (SVM). This is achieved by exploring a small subset of the available values. According to the research, optimising SVM parameters should lead to an accurate classification rates on the training sets of data.

Here we introduce Red Panda Optimisation (RPO), a novel bio-inspired metaheuristic algorithm that mimics the actions of real red pandas. Inspired by two of red pandas' most distinctive natural behaviors—(i) their approach for hunting and (ii) their habit of resting atop trees—RPO was born. Both phases of the suggested RPO method are based on mathematical models of red panda feeding strategies and climbing behaviors: exploration and exploitation. The suggested method avoids the need of a parameter adjustment procedure due to the absence of a control variable in its mathematical modelling, which is its primary benefit.

- Here are the steps involved in the suggested plan.
- Step 1. Bring in the training sets, bat population, and stopping criteria.
- Step 2: Bat population initialization: First, we randomly generate a population of M bats, where M is the total population size. Additionally, we initialise all of the other parameters related to BA.
- Step3: Determine the current global ideal location by measuring the bats in the original population. The SVM classifier's cross-validation accuracy on training data is used here as the fitness value.
- Step4: Use (9-11) to update each bat's location, velocity, and frequency.
- Step5: Pick one ideal solution from the list, and then build a local solution that is close to that one.
- Step6. Determine the fitness score of the revised solution that was created in step 5.
- If you want to refresh each parameter individually, you may do so in Step 7. Otherwise, you can go to Step 8 or Step 9.
- Increase ir and reduce Ai; then, replace the old solution using the new ones. Step 9: Next, compare the population's best answer to the existing global ideal solution and then choose how to renew the best solution.
- Step 10. Keep going until the halting requirement is satisfied.
- At the end of Step 11, we use the parameters for SVM that were determined by the global optimum solution. Fig.3 is illustrated in RPO Algorithm.

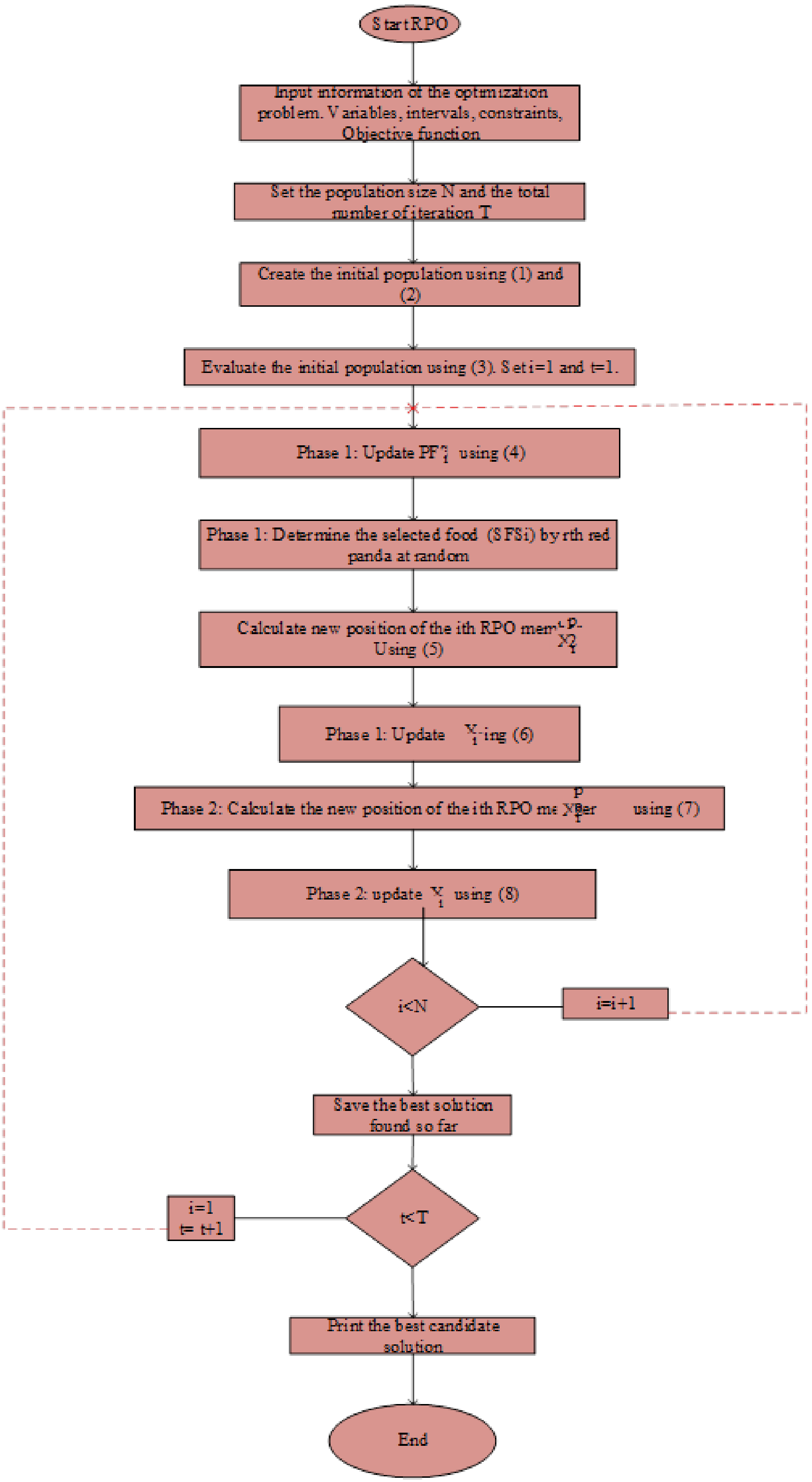

Start RPO
Input information of the optimization problem. Variables, intervals, constraints, Objective function
Set the population size N and the total number of iteration T
Create the initial population using (1) and (2)
Evaluate the initial population using (3). Set i=1 and t=1.
Phase 1: Update PF_i^t using (4)
Phase 1: Determine the selected food (SFSi) by rth red panda at random
Calculate new position of the ith RPO mem X_i^{P1} Using (5)
Phase 1: Update X_i ing (6)
Phase 2: Calculate the new position of the ith RPO me X_i^{P2} er using (7)
Phase 2: update X_i using (8)
i<N
i=i+1
Save the best solution found so far
t<T
i=1
t= t+1
Print the best candidate solution
End

Fig. 3. RPO Algorithm

IV. Results & Discussion

Here we'll go over the basics of the evaluation metrics, including recall, cross-validation, precision, and F1 scores, among others, and how we use them to judge the quality of our work.

4.1 Dataset

- CEDAR: It has the official stamp of 55 signatories hailing from a wide range of professional and social backgrounds. We take into account 24 forgeries and 24 real signatures for each user.

- BHSig260: The Hindi and Bengali languages are represented in this collection of signatures. There are 260 signatories altogether, including 100 from Bengali and 160 from Hindi. For every user, there are 24 authentic signatures and 30 fake ones.
- UTSig: With 8280 signatures in total, the UTSig collection is comprised of signatures signed in Persian language. A total of 115 people have had their signatures recorded. There are 27 real signatures and 45 fake ones for every person.

4.2 Experimental Setup

The computational setup for the aforementioned investigations is as follows.

- Intel Skylake, with 8 virtual CPUs and 32 GB of RAM, is the platform for the system configuration. The GPU used is the NVIDIA Tesla V100. All of the aforementioned settings are live in GCP.
- Systems Requirements: Debian GNU/Linux 9.7
- Python Scripting in Jupyter Notebook, Numpy, Scikit-learn, OpenCV, Tensorflow, and Keras are all part of the software package.

Machine learning models' parameter settings are determined by trying out various values and picking the ones that work best via a trial-and-error process. You may find the optimal parameter values for several models in Table 1.

Table 1: Variables' Configuration.

Models	Variables' Configuration
K-NN	$k = 3$
SVM	
CNN	Optimizer- Adam (Stochastic Gradient Descent) Loss: Categorical Crossentropy Batch Size: 32 Steps per Epoch: 622 Number of Epochs: 15 Validation Steps: 20
VGG-16	thirteen convolutional layers five max-pooling layers three dense layers 20 epochs, with a batch size of 128
GoogLeNet	Optimizer- Adam (Stochastic Gradient Descent) Loss: Categorical Crossentropy Batch Size: 32 Steps per Epoch: 622 Number of Epochs: 10

ResNet-50	Optimizer- Adam (Stochastic Gradient Descent) Loss: Categorical Crossentropy Batch Size: 50 Steps per Epoch: 50 Dense Layer Activation Function: Softmax Number of Epochs: 30 Early Stop Criteria: 15 Epochs

Using the K-fold cross validation method for the CNN model, we were able to further generalize the results. The literature indicates that 10 is the standard value for K. Table 2 displays the outcomes of K-fold validation, with a significance level of K=10.

Table 2: K-Fold validation results ($K = 10$)

Folds	Accuracy in %	Recall	Precision
Fold I	99.299	0.992	0.995
Fold 2	99.366	0.993	0.993
Fold 3	99.400	0.994	0.995
Fold 4	99.466	0.994	0.994
Fold 5	99.266	0.992	0.992
Fold 6	99.500	0.995	0.995
Fold 7	99.466	0.994	0.995
Fold 8	99.400	0.993	0.994
Fold 9	99.566	0.995	0.995
Fold 10	99.566	0.995	0.995
Mean	99.430	0.993	0.994
Standard Deviation	0.09712	0.00095	0.00096

4.3 Performance Analysis

After the training process was complete, the computing technique was tested on the test dataset. We used the AUC-ROC curve values, accuracy, precision, recall, and f1-score to find out how well the model worked. The following is an exhaustive investigation of the assessment criteria employed in this research. In the above ideas and formulae, TP stands for true positives, FN for fake negation, TN for true negatives, with FP for false positives.
Accuracy (Acc.)
This metric determines how many accurate predictions there were out of all the right projections. It may be seen in Equation (4):

$$\text{Accuracy } = \frac{TP+TN}{(TP+TN+FP+FN)} \tag{4}$$

Precision (Prec.)

The reliability of classification accuracy as a measure of overall simulation outcomes has been shown by several situations. An example of this would be a scenario where the distribution of classes is not uniform. If we assume that all data is of the highest quality, it is ridiculous to expect a higher accuracy rate. Contrarily, precision suggests that inconsistencies may be achieved by reusing the identical tool, such when inspecting the same component. The term "precision" describes one such statistic:

$$\text{Precision } = \frac{TP}{(TP+FP)} \tag{5}$$

Recall

Remembered is another important indicator; it represents the fraction of the initial dataset that the algorithm correctly categorizes. The calculation for recall is as:

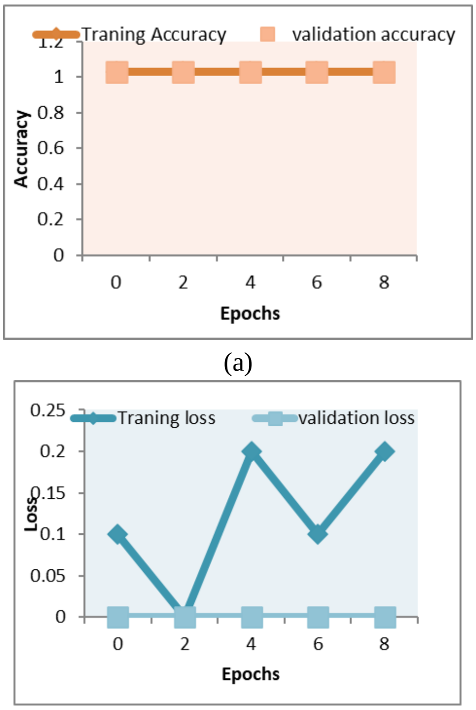
$$\text{Recall } = \frac{TP}{(TP+FN)} \tag{6}$$

F1-Score

The F1-score is a well-liked metric that integrates recall and accuracy assessments. To get the F1score, one must:

$$\text{F1 } - \text{ score } = \frac{2\times(\text{ Precision } * \text{ Recall })}{(\text{ Precision } + \text{ Recall })} \tag{7}$$

Accuracy, Precision, Recall, as well as F1-score are shown in Table 3 for each model that was considered. The proposed model outperforms both deep learning (CNN and ResNet-50) and basic machine learning (SVM) models, according to the results (Table 3).



(a)

(b)

Fig. 4. Training and validation accuracy comparison

Figure 4 shows the accuracy graph and Figure 4 shows the loss chart for the suggested model. The suggested model is simpler than ResNet-50 and CNN, which are much more complicated. Although CNN and ResNet-50 are deeper architectures with a greater ability to learn complicated patterns, they may need larger datasets or more hyperparameter tweaking to work well on smaller datasets. Due to their increased complexity, these models may have significantly less accuracy since they are unable to efficiently learn from the dataset's limited information.

Table 3: Overall Comparison Results for Dataset 1

Models	Training Accuracy	Testing Accuracy	Precision	Recall	F1-Score
K-NN	98.5%	97.266%	0.98	0.98	0.98
SVM	98.45%	99.266%	0.99	0.99	0.99
GoogLeNet	99.766%	99.522%	0.998	0.997	0.9978

VGG-16	96.49%	94.059%	0.9856	0.996	0.9625
CNN	99.133%	97.469%	0.9923	0.99	0.991
ResNet-50	98.5%	97.256%	0.997	0.984	0.99

Table 4: Overall Comparison Results for Dataset 2

Models	Training Accuracy	Testing Accuracy	Precision	Recall	F1-Score
K-NN	97.5%	98.266%	0.98	0.98	0.98
SVM	99.45%	99.366%	0.99	0.99	0.99
GoogLeNet	99.765%	99.522%	0.998	0.9976	0.9978
VGG-16	96.499%	96.059%	0.9746	0.9496	0.9625
CNN	99.133%	97.569%	0.9923	0.99	0.991
ResNet-50	97.5%	95.56%	0.987	0.984	0.985

Table 5: Overall Comparison Results for Dataset 3

Models	Training Accuracy	Testing Accuracy	Precision	Recall	F1-Score
K-NN	98.45%	98.266%	0.98	0.98	0.98
SVM	98.45%	99.66%	0.99	0.99	0.99
GoogLeNet	99.66%	99.22%	0.998	0.9976	0.9978
VGG-16	97.399%	97.059%	0.9756	0.9496	0.9625
CNN	98.133%	98.469%	0.9923	0.99	0.991
ResNet-50	97.5%	98.256%	0.987	0.984	0.985

V. Conclusion

Surprisingly, the suggested model outperformed the competition on every criterion. F1 score, recall, accuracy, and precision were among these measures. Our study's results have significant practical implications. The digitalization of physical records, document scanning, and ID card identification are just a few examples of how accurate handwritten signatures may automate and streamline processes. These applications may be made much more accurate and reliable by using the latest developments in CNN architectures and deep learning. Although we have learned a lot from our study, there are still many ways to improve and expand upon it. It is possible to go more into hyper parameter tweaking and architectural optimization as means of further refining and optimizing ResNet-50 and CNN. To make better use of pre-trained models, it might be worthwhile to look at transfer learning from bigger datasets. It is possible to attain even greater precision by using assembling approaches that integrate the qualities of various models. This allows us to take use of their complimentary skills. Further information on how deep learning models make decisions may be gleaned from studies that concentrate on interpretability. A better comprehension of the method of classification might be achieved using techniques that allow for the visualization of learnt features or the identification of significant areas in input photos. Final thoughts: our results have real-world applications, and next steps for research include improving and enhancing current models, using transfer learning from bigger datasets, investigating assembling methods, and developing interoperability-focused approaches.

References

[1] Huang, F., & Lu, H. (2023). Multiscale Feature Learning Using Co-Tuplet Loss for Offline Handwritten Signature Verification.

[2] Hashim, Z., Ahmed, H.M., & Alkhayyat, A.H. (2022). A Comparative Study among Handwritten Signature Verification Methods Using Machine Learning Techniques. Scientific Programming.

[3] Muhtar, Y., Muhammat, M., Yadikar, N., Aysa, A., & Ubul, K. (2023). FC-ResNet: A Multilingual Handwritten Signature Verification Model Using an Improved ResNet with CBAM. Applied Sciences.

[4] Patil, P., & Patil, B.V. (2021). A Review - Signature Verification System Using Deep Learning: A Challenging Problem. International Journal of Scientific Research in Science and Technology, 295-298.

[5] Mosaher, Q.S., & Hasan, M. (2022). Offline Handwritten Signature Recognition Using Deep Convolution Neural Network. European Journal of Engineering and Technology Research.

[6] Bhavani, S.D., & Bharathi, R.K. (2023). A multi-dimensional review on handwritten signature verification: strengths and gaps. Multimedia Tools and Applications, 83, 2853-2894.

[7] Sangeetha, M., & Ramjee, D.M. (2023). Enhancing Handwritten Signature Verification With Siamese And Convolutional Neural Networks.

[8] Upadhyay, R.R., Mehta, R., & Singh, K.K. (2023). Multi-dilation Convolutional Neural Network for Automatic Handwritten Signature Verification. SN Computer Science, 4.

[9] Albasu, F.B., & Al Akkad, M.A. (2023). Exploiting Deep Learning Techniques for the Verification of Handwritten Signatures. Intellekt. Sist. Proizv..

[10] Fathy, M.A., Mohamed, A.E., & Salem, S.A. (2023). HSCM: A New Framework For Handwritten Signature Verification Using ConvMixer. 2023 5th Novel Intelligent and Leading Emerging Sciences Conference (NILES), 356-361.

[11] Kumar, A., & Bhatia, K. (2022). Offline Handwritten Signature Verification Using SVM and LBP.

[12] Tariq, U., Hu, Z., Tariq, R., Iqbal, M.S., & Sadiq, M. (2023). High-Performance Embedded System for Offline Signature Verification Problem Using Machine Learning. Electronics.

[13] Ahmed, S.S., Mehmood, Z., Awan, I.A., & Yousaf, R.M. (2023). A Novel Technique for Handwritten Digit Recognition Using Deep Learning. Journal of Sensors.

[14] (2023). ENHANCING SIGNATURE VERIFICATION USING CONVOLUTIONAL NEURAL NETWORKS FOR FORGERY DETECTION.

[15] Muhtar, Y., Kang, W., Rexit, A., Mahpirat, & Ubul, K. (2022). A Survey of Offline Handwritten Signature Verification Based on Deep Learning. 2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML), 391-397.

[16] Korade, P.M. (2023). Handwritten Signature Verification using Deep Learning. INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT.

[17] Leghari, M., Memon, S., Dhomeja, L.D., Jalbani, A.H., & Chandio, A.A. (2022). Online signature verification using deep learning based aggregated convolutional feature representation. J. Intell. Fuzzy Syst., 43, 2005-2013.

[18] Sharma, N., Gupta, S., Mehta, P.S., Cheng, X., Shankar, A., Singh, P., & Nayak, S.R. (2022). Offline signature verification using deep neural network with application to computer vision. Journal of Electronic Imaging, 31, 041210 - 041210.

[19] Priya, K.S., Ebenazer, M.P., Felshia, M.T., & Ruth, M.E. (2022). AUTOMATIC MALWARE SIGNATURE CLASSIFICATION USING DEEP LEARNING.

[20] Pokharel, S., & Shakya, S. (2021). Deep Learning Based Handwritten Signature Recognition.

[21] Asish, K., Teja, P.S., Chander, R.K., & Hema, D.D. (2023). NeuroWrite: Predictive Handwritten Digit Classification using Deep Neural Networks. ArXiv, abs/2311.01022.

[22] Abdulhussien, A.A., Nasrudin, M.F., Darwish, S.M., & Alyasseri, Z.A. (2023). A Genetic Algorithm Based One Class Support Vector Machine Model for Arabic Skilled Forgery Signature Verification. Journal of Imaging, 9.

[23] K, M., R Bhat, A., Nerella, P., Baburaj, P., & K S, S. (2021). A Comparative Study of Transfer Learning Models for Offline Signature Verification and Forgery Detection. Journal of University of Shanghai for Science and Technology.

[24] Alkattan, Z.M., & Aldabagh, G.M. (2022). Offline Signature Biometric Verification with Length Normalization using Convolution Neural Network. Baghdad Science Journal.

[25] Subramaniam, M., Teja, & Mathew, N.A. (2022). SIGNATURE FORGERY DETECTION USING MACHINE LEARNING..

[26] T, M., K, T., S, M., P, V., U, M., & S, M. (2023). AES Algorithm for the Next Generation of 5G Network Encryption Standards. 2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), 1-6.

[27] Tamilarasi, K., Shalini, B., Soundarya, M.M., Sivapavina, J., & Sowmiya, E. (2022). Automatic Environment Protection Using Mixed Signal Processor For Covid-19. 2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC), 1392-1396.

[28] Tamilarasi, K., & Kalyani, S.N. (2020). Design and implementation of deep learning strategy based smart signature verification System. Microprocess. Microsystems, 77, 103119.

[29] Tamilarasi, K., & Kalyani, S.N. (2017). A survey on signature verification based algorithms.