Turkish Journal of Computer and Mathematics Education Vol.12 No.12 (2021), 62-77

Research Article

Social Ski Driver-Jaya Optimization-Enabled Deep Convolution Neural Network for Signature Verification

N. Neelima

Asst. Prof., R.V.R & J.C College of Engineering, Guntur
Email: neelimanalla1979@gmail.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 23 May 2021

Abstract:
With the rapid growth of technology, security plays a very important role for avoiding forgeries and fake. It is also one of the most easily forgeable biometric identity when compared to other biometric features like thumb impression, face recognition etc. Thus, the signature verification is the most important features to check the person authenticity. Therefore, an effective method named Social ski driver-Jaya (SSD-Jaya) optimization algorithm is proposed in this research to verify the signature. The pre-processing is initially done by the input image in which the cropping and the binarization is done. Here, the cropping is performed for resizing the input image with the constant aspect ratio, whereas the binarization is carried out using otsu thresholding for converting the original image into binary image. After that, the feature extraction is carried out based on hierarchical skeleton, yields the addition data for enhancing the accuracy of image matching and it does not require manual intervention. The hierarchical skeleton image is fed to the formation of the graph that indicates the highly significant points in the image. At last, the classification is performed using the developed SSD-Jaya optimization algorithm, which is designed by integrating Social ski driver (SSD) and Jaya algorithm. The performance of the signature verification Proposed models are analyzed depending on the three critical measures: precision, sensitivity, and specificity. the developed model delivers the highest precision of 0.2 and sensitivity of 0.98 with respect to standard deviation

Keywords: Signature verification, Social sky driver, Jaya optimization, Deep Convolution Neural Network, Hierarchical skeleton.

1. Introduction

Now a days, the biometrics is used to identify or authenticate the person in the daily life. Biometrics technology is employed in the wide range of security applications. The main goal is to find the person based on behavioural or physiological traits. In addition, the biometric systems are generally used in two states: identification and verification. For identification, the user may provide the biometric sample, and the main purpose is to find the users enrolled in system [23][11]. Here, the system user claims an identity for providing the biometric sample. Thus, the recognition is carried out using the biological traits measurements, like the face, iris, fingerprint and so on. Later, the behavioural traits, such as handwritten and voice signature is concerned [12] [11]. Handwritten signature is the characteristic of unique human personal for person authentication [24].

The handwritten signature is the type of biometric trait, because of their ubiquitous usage for verifying the person's in legal, financial, and the administrative areas. The major reasons for their widespread usage are that the process to collect the handwritten signatures is non-invasive, and the people are familiar using their signatures in the daily life [13], but handling huge number of signatures is cumbersome. Hence, several algorithms are introduced for dealing the issue of signature-enabled person authentication in several applications, such as crime detection, person verification and identification, and the bank cheque fraud detection and so on [10][11][3]. The method for signature verification differentiates between the persons forged and original signatures, thus rejecting the forged ones and accepting the raw signatures [25].

Signature verification system aims to discriminate automatically if a bio-metric sample is indeed of claimed individual, which means the query signatures are classified as the forgeries or the genuine. The forgeries are broadly categorized into simple, random, and simulated or skilled forgeries. Here, the forgery consists of various semantic meaning the genuine signature from user with different shape. In simple forgeries, the forger has the knowledge of user's name, but failed to consider users signature [1][10]. The forgery signature may have the similarities to genuine signature with their full name or the part of it. In the skilled forgeries, the forger access both the user's signature

and the name, and imitating the user's signature, that results having higher resemblance to genuine signature, and thus very harder to detect. Shah *et.al* [14] presented an approach for signature verification. Here, 15 signatures were used as the input and the feature extraction and the classification steps are carried out for improving the overall strengths.

This paper incorporates the signature method suggested by SSD-Jaya, using a Deep CNN model as the basis. Degreezin points out that in order to carry out the signature authentication process, we must use these steps: This is the primary stage, the second is the post-processing, and the third is the final processing. First, the input picture is resized and the otsu thresholding are performed on it. The image graph is built using a hierarchical structure from which features are extracted. Feature-based picture matching is employed for the signature verification with the use of high-tech graphics processing can be successful Use of the proposed SSD-Jaya-based Deep CNN classifier brings in the signature verification to be completed. Using the method suggested is reliable. Both speed and accuracy are increased.

Thus the document is laid out: The current methods for signature verification are described in section 2, as well as obstacles for potential new methods that remain the impetus for study. The signature-verification plan is presented in section 3, and the results are seen in section 4. Finally, segment 5 comes to an end on this study project.

2. Motivation

The committee discusses the literature review of approaches and their shortcomings in detail. This study uses eight previous signatures to look at the other eight known processes.

2.1 Literature review

Several methods related to signature verification are described, and analyzed as follows**:** Aini Najwa Azmi *et al.*[1] developed an approach for signature verification system (SVS) that employed Freeman chain code (FCC) as the data representation. Initially, the input image was fed to the pre-processing stage where the noise removal, binarization, thinning, and cropping were performed. After that, the feature extraction was carried out based on boundary-enabled style of signature images. Subsequently, the extracted FCC was partitioned to several parts. At last, the Euclidean distance measure and the k-nearest neighbours was introduced for verification. Here, the hybrid verifier was not considered in order to improve the system performance. Luiz G. Hafemann *et al.* [2] presented adversial examples for the online handwritten SVS for characterizing various attacks. In this framework, the type-2 attacks were identified in the entire systems with less knowledge scenario in which the attacker failed to access the signature utilized for training writer-dependent classifiers. Here, the success rate is reduced, but the impact of the physical attacks was not analysed.

Ankan Kumar Bhunia *et al.* [3] developed writer-based signature verification approach where the texture features, like discrete wavelet and the local quantized patterns (LQP) features were used for extracting statistical and transform information from the input images. After that, the score-level classifier was fused with average method for achieving final verification score. The method failed to consider different datasets with the signatures written by various shapes or scripts to enhance the system performance. Antonio Parziale *et al.* [4] presented Stability Modulated Dynamic Time Warping (SM-DTW) that aims to compute the regions of stability and find the difference between the stability regions than the ordinary regions. The method failed to consider larger kinematics to extract the strokes embedded into a motor plan.

Xinghua Xia *et al.* [5] employed discriminative feature selection approach for the online signature verification. To improve the accuracy, the discriminative features were chosen for the verification purpose. The main drawback of this system is decreased efficiency of the online signature verification. Om Prakash Patel *et al.* [6] presented enhanced quantum-enabled neural network learning algorithm (EQNN-S) for verifying the signature. The quantum computing concept was utilized for deciding threshold of neurons and the connection of weights. The boundary parameter was established for determining neuron threshold that assists effective learning. Online signature was not considered for verification.

Marcin Zalasinski, and Krzyszt Cpalka [7] developed partitioning approach for increasing the precision of signature processing. This method employs one-class fuzzy system to evaluate the similarity between reference as well as the test signatures. The population-based algorithm was not considered for the subset selection. Marcin Zalasiński *et al.* [8] developed an method for signature

verification using global features, and is evaluated using the function of genetic algorithm. In addition, the weights were determined based on selected global features and utilized for the classification. Here, the classification was done on the basis of fuzzy one-class classifier.

3. Signature verification using the proposed Social ski driver-Jaya optimization-based Deep convolutional neural network

In this novel strategy part, we are using the proposed SSD-Jaya algorithm to demonstrate how the signature verification algorithm functions. As mentioned above, there are three separate stages of verification that will occur during the signature verification process: preliminary screening, feature extraction, and classfication. Until the input pictures are processed, they are cropped and binarized. Here, cropping is used to make the picture same-size adjustment, while binarization is done using Otsu's thresholding method is performed using Otsu's ratio. After that, the features are drawn from the photographs, which is done in the feature extraction step, hierarchical skeleton. Also, hierarchical attributes including the number of grid vertices, number of linked components, graph features, and unconnected components are used in the function extraction. The output features are forwarded to the signature stage for determining whether or not they are the correct signature is correct. The photos' signatures are identified using a Deep CNN model, which has been trained with the proposal SSD-SSD-Jaya-optimized Jaya's signature matching technique to improve the accuracy of the recognition system. Furthermore, the suggested SSD-Jaya algorithm incorporates the Jaya algorithm.

Figure 1 portrays the signature verification framework using proposed SSD-Jaya-based DeepCNN.

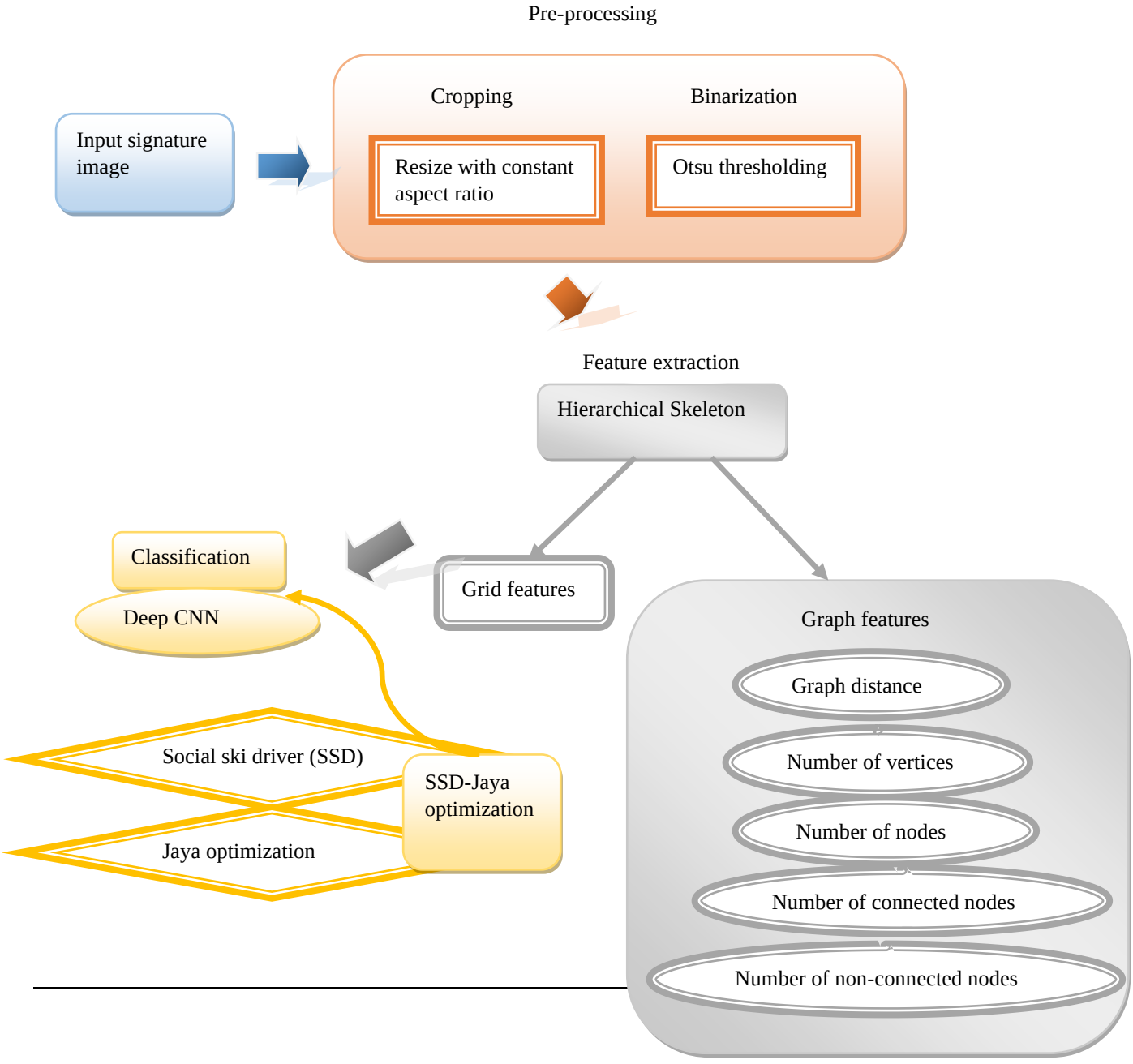


Figure 1. Schematic view of object detection framework using proposed SSD Jaya-based Deep CNN

Assume the database D with d number of signature images, which is given by,

$$D = \{J_d\}; (1 \le u \le d) \quad (1)$$

where, the term d represents the total images and J_d indicate the d^{th} image. Each image J_d is subjected to the pre-processing module, which carries resizing and binarization of images. The size of the database is expressed as $[Q \times Y]$.

3.1. Pre-processing based on resizing and binarization

The initial step involved in the signature verification is the pre-processing of image. The pre-processing step is very necessary for enhancing the quality of image and to verify the signature accurately. Here, the input image J_d is fed to the pre-processing stage where the pre-processing is carried out based on cropping, and the binarization using Otsu's thresholding.

Cropping: Cropping is the initial pre-processing task to remove the outer parts of the image for improving accentuating the subject matter, and framing. Based on cropping, the size of the input image is reduced for effective verification.

Binarization based on Otsu's thresholding: Otsu thresholding is the simple binarization approach, utilized for converting the original image into binary image based on threshold value. Otsu approach consists of two classes of pixels, like background and foreground pixels, and then the optimum threshold is computed to separate two classes. The threshold value ranging from 0 and 1. This process is utilized to perform feature extraction more efficiently. The pre-processed image is denoted as T_c.

3.2 Feature extraction based on hierarchical skeleton:

Once, the input signature image is pre-processed and then, it is required to extract the appropriate features using Hierarchical skeleton [19]. The pre-processed image T_c is fed as input to the feature extraction phase. Here, the more useful and the appropriate features are effectively extracted using Hierarchical skeleton where grid features and graph features, like number of nodes, graph distance, number of vertices, number of connected nodes, and number of non-connected nodes are considered. The feature extraction process enables the proposed optimization algorithm to train the Deep CNN classifier faster. It minimizes the complexity and the interpretation more easily. Moreover, the performance of accuracy in verifying the signature is effectively increased.

a) Hierarchical skeleton:

The hierarchical skeleton is utilized for removing all the skeleton branches of insignificant shape regions based on boundary extraction approach. The main advantage of this method is that there is no need to search the root level when the entire hierarchical levels are arranged with pruning lines. In addition, the hierarchical skeleton uses boundary information for matching the objects by descriptor as they are hierarchically preserved. Thus, the hierarchical skeleton reduces ambiguous and increases the accuracy of single skeleton matching.

Assume the planar shape B and the initial skeleton of shape B is represented as $G^k(B)$, which is generated using max-disc model. In the circle, the centre points that are linked with the shape boundary is represented as, $a \in G^k(B)$ and the contact point of a on shape boundary are termed as generating points and the term k represents the iteration index of Discrete Curve Evolution(DCE) and is decreased till three. The term j signifies one of the following below steps.

- The B^{th} boundary is referred as initial polygon Q^k in which the polygon simplification approach is utilized to simplify the polygon into Q^j.
- Using Q^j, the $G^k(B)$ is pruned to eliminate the skeletal points $a \in G^k(B)$ hence the generating points are remained in same contour, which is the part of shape boundary between two neighbouring vertices of Q^j.

The result obtained from the individual segment is the individual pruned point in accordance with polygon partition. Thus, the un-significant points are eliminated to obtain Q^{jth} simplified polygon. Consider the consecutive pair segments a_1 and a_2 that are connected with the end points $a_1 \cup a_2$. Hence, similar polygon sequence, $Q = Q^k, Q^{k-1}, ..., Q^3$ so that the polygons Q^{k-j} are formed by removing the single vertex r from Q^{k-j+1}. The smallest shape contributed with the use of L^{th} measure is expressed as,

$$L(a_1, a_2) = \frac{\gamma(a_1, a_2) n(a_1) n(a_2)}{n(a_1) + n(a_2)} \tag{2}$$

where, the corner angle with the consecutive pair segments is denoted as $\gamma(a_1, a_2)$, the length function is represented as n. When $\gamma(a_1, a_2)$ is high, the end points $a_1 \cup a_2$ is also increased using L^{th} measure. On the other side, the skeletal matching is done using the distance measure.

3.2.1 Graph construction for input image

This section presents the construction of graph [21] for the input image using the hierarchical skeleton of image for effective image retrieval through feature matching. From the input binary image, the hierarchical skeleton is introduced based on the derived joints and the end points. Hence, the labelling forms an initial step corresponding to the graph formation so that the skeleton pints are labelled at an individual point using the local variations of radial functions. The skeleton point label is on the basis of shape and they are denoted as label 1, 2, 3, and 4.When the radial function increases or decreases, the particular segment attains label-1 however, the sign may change vary between them. The two ends of the radial functions are increased then the shape acquires label-2 with local minima. The label-3 is obtained by the skeletal segment with constant radial function, whereas the label-4 is for point possessing local maximal radial function.

The labelling of individual point related to the skeletal branch is based on time, and the skeletal branch is obtained based on connected points between the end points of medical axis. After that, the labelled points with their corresponding branch are grouped on the basis of labels and its connectivity in the same way the individual group related to similar label connected points are stored in the graph nodes and the individual skeletal branch belongs to more than two nodes. Finally, the edges are provided between the nodes for establishing directed acyclic graph with directed edges based on time and shock points. Therefore, the shock branches are illustrated in graph as nodes and the joint points and the skeleton branches are numbered for plotting the graph followed by labelling individual branches. The label-2 and 4 are subjected to joint points using the connected shock types in which each branch and the joint point are indicated as nodes. The nodes marked as type-4 is ranked as top hierarchy and the branches connecting the nodes produce the descendents. The plotting is repeated till the entire joint points and the branches are plotted. After the development of graph, the features are achieved from the graph.

3.2.2 Generation of feature database

The features are utilized for effective retrieval mechanism by mitigating the time and the complexity associated with image retrieval. The grid features and graph features, like graph distance, number of vertices, number of connected nodes, number of nodes, and the number of non-connected nodes are briefly illustrated below.

a) Graph distance: It is computed by summing the distance between the nodes in graph. The distance of graph ensures the effective retrieval of image from the dataset. The graph distance is denoted as f_1 and the feature dimension is represented by, $[1 \times 1]$.

b) Number of vertices: The vertex is defined as the vertex points in the graph. The vertex points are also called as corner nodes. The number of vertices is denoted as f_2 with dimension $[1 \times 1]$.

c) Number of nodes: The nodes is defined as the key points in the graph for enabling the connection in the graph. Here, the skeleton image branches are indicated as the nodes and they introduce the connection defining the shape of binary input image. The number of nodes is denoted as f_3 of size $[1 \times 1]$.

d) Number of connected vertices: The connected vertex signifies the line that connects more than two nodes in graph and its dimension is represented by, $[1\times1]$. The number of connected vertices is represented as f_4.

e) Number of non-connected vertices: The vertex that are not connected with the other nodes is termed as non-connected vertex. The number of non-connected nodes is denoted as f_5 of size $[1\times1]$.

f) Grid features: In the grid features, the graph of the image is partitioned as grids based on grid lines, and the dimension of grid line is $[16x16\ ]$, whereas the grid-based feature dimension is expressed by, $[1\times(16\times16)]$.

The feature vector is expressed as,

$$F=\{f_1, f_2, f_3, f_4, f_5, f_6\} \tag{3}$$

3.3 Classification using Deep Convolution Neural Network

function extraction is done using the suggested Jaya Deep CNN for the classification. Here, the function vector is added as the input to a Deep Convolutionaluational Neural Network. For effective verification, the Deep CNN is used, and the model-induced biases are set up correctly. Deep CNN's design and the planned SSD moves can be understood as such:

a) Architecture of DCNN

See Figure 2 for the simple design of the Deep CNN. DCNN is used to do the big part of verifying signatures. Often called the Deep CNN, this network contains 3 convolutionalues, rectification, and fully connected layers. Each layer has a different goal, like the function maps in conv layers, classifier layers apply sub-downsample features, and FC layers classify features.

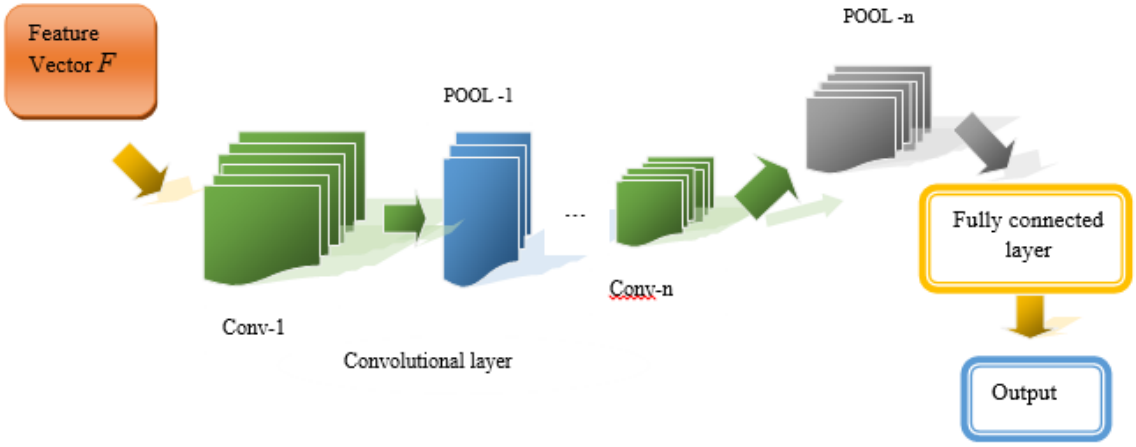


Figure 2. Architecture of DCNN

Conv Layers: Assume the input to DCNN is F and therefore, the output from conv layer is expressed by,

$$\left(H_x^y\right)_{U,V}=\left(E_x^y\right)_{U,V}+\sum_{b=1}^{v_1^{b-1}}\sum_{e=-X_1^r}^{X_1^r}\sum_{t=-X_2^r}^{X_2^r}\left(\gamma_{x,e}^y\right)_{e,t}*\left(F\right)_{U+e,V+t} \tag{4}$$

where, the symbol $*$ signifies the convolutional operator for extracting local patterns from the alternative conv layers, $\left(H_x^y\right)_{U,V}$ signifies the fixed feature map or output of y^{th} conv layers, which is centeredat (U,V). The output from previous $(y-1)^{th}$ layer forms the input to y^{th} conv layer. The term $\gamma_{x,e}^y$ refer to the x^{th} conv layer weight and the bias of x^{th} conv layer are denotedas, E_x^y. Let us consider b , e , and t as the notations of feature maps.

ReLU layer:The output from y^{th} layer is the activation function of preceding $(y-1)^{th}$ layer, and is expressed as,

$$H_x^y = Afn\left(H_x^{y-1}\right) \tag{5}$$

where, *Afn* denotes the activation function.

Fully connected (FC) layers: The features generated byReLU layer are given to FC layer, and the output of FC layer is representedby,

$$R_x^y = \delta\left(H_x^{y-1}\right) \text{with } H_x^y = \sum_{b=1}^{v_1^{b-1}} \sum_{e=-X_1^r}^{X_1^r} \sum_{t=-X_2^r}^{X_2^r} \left(\gamma_{x,e}^r\right)_{e,t} * \left(F\right)_{U+e,V+t} \tag{6}$$

where, $\left(\gamma_{x,e}^r\right)_{e,t}$ denotes the weight.

b) Training of Deep CNN based on SSD-Jaya

The Deep CNN verification training using the suggested SSD-Jaya optimization methodology uses various forms of reinforcement learning. The suggested "SSDaya" algorithm integrates SSD [16] The SSD optimization is designed to reduce the size of the SVM parameters with the aim of enhancing device efficiency in mind. "Finding an optimal or a good answer" is the primary goal of SSD This is an optimal approach for addressing multi-objective problems. Non-linear questions, which are complicated, can also be handled using the approach JAYA uses candidates to decide which approach should be implemented, and is not based on any criteria. the application of the Jaya algorithm is simpler; it operates in one step The Jaya algorithm formula is used to compute the SSD update values. Using the modification would render the solution more effective, and enhance its convergence. The algorithm worked as follows:

Step 1: Initialization: Stage 1 of the proposed SSD-Jaya algorithm is to count all the search agents found by the consumer and store their initial locations in a global array. The whereabouts of the performers is described as, is as,

$$X_v^\tau; \left(1 \le v \le c\right) \tag{7}$$

where, X_v^τ represents the agents position at time τ, and the number of variables are denoted as v. Here, $X \in \left(\gamma_{x,e}^y, \gamma_{x,e}^r, E_x^y\right)$.

Step 2: Fitness function evaluation: The position of the agents is made dependent on the problem, or the task is to minimize it. In comparison to the minimal, the value of the objective function represents the best solution, which is why the optimum solution is found. There is only one possible source of the mistake.

$$MSE = \frac{1}{Z}\left[\sum_{h=1}^{Z} I_{t\arg et} - R_x^y\right] \tag{8}$$

where, $I_{t\arg et}$, and R_x^y are the estimated and target output of classifier. The term Z denotes the total number of samples.

Step 3: Update the solution based on SSD-Jaya algorithm: After evaluating the objective function, the solution undergoes position update on the basis of SSD-Jaya. The update equation of SSD velocity h_v^τ is given by,

$$X_v^{\tau+1} = X_v^\tau + h_v^\tau \tag{9}$$

$$h_v^\tau = \begin{cases} e\sin\left(m_1\right)\left(\mathrm{B}_v^\tau - X_v^\tau\right) + \sin\left(m_1\right)\left(\mathrm{A}_v^\tau - X_v^\tau\right) \; if \left(m_2 \le 0.5\right) \\ e\cos\left(m_1\right)\left(\mathrm{B}_v^\tau - X_v^\tau\right) + \cos\left(m_1\right)\left(\mathrm{A}_v^\tau - X_v^\tau\right) if \left(m_2 > 0.5\right) \end{cases} \tag{10}$$

Where, the term h_v denotes the velocity of X_v, and the uniformly distributed random numbers are indicated as m_1 and m_2 ranging from 0 to 1. The term B_v represents the best solution of v^{th} agent, and the term A_v indicates the mean global solution for whole population.

Considering the first equation of SSD velocity update equation,

$$h_v^\tau = e\sin(m_1)\mathrm{B}_v^\tau - e\sin(m_1)X_v^\tau + \sin(m_1)\mathrm{A}_v^\tau - \sin(m_1)X_v^\tau \tag{11}$$

Rearranging equation (11), the solution becomes,

$$h_v^\tau = -X_v^\tau\left(e\sin(m_1)+\sin(m_1)\right) + e\sin(m_1)\mathrm{B}_v^\tau + \sin(m_1)\mathrm{A}_v^\tau \tag{12}$$

Then, the equation (12) is modified using the Jaya algorithm to improve the effectiveness of approach and to identify the solutions to several optimization problems. The update equation of the Jaya algorithm is expressed as,

$$X_v^{\tau+1} = X_v^\tau + m_{1v}\left(X_v^{best} - \left|X_v^\tau\right|\right) - m_{2v}\left(X_v^{worst} - \left|X_v^\tau\right|\right) \tag{13}$$

Assuming X_v^τ as positive, and the solution becomes,

$$X_v^{\tau+1} = X_v^\tau + m_{1v}\left(X_v^{best} - X_v^\tau\right) - m_{2v}\left(X_v^{worst} - X_v^\tau\right) \tag{14}$$

$$X_v^{\tau+1} = X_v^\tau\left(1 - m_{1v} + m_{2v}\right) + m_{1v}X_v^{best} - m_{2v}X_v^{worst} \tag{15}$$

$$X_v^\tau = \frac{X_v^{\tau+1} - m_{1v}X_v^{best} - m_{2v}X_v^{worst}}{1 - m_{1v} + m_{2v}} \tag{16}$$

where, the term $X_v^{\tau+1}$ indicates the value of v^{th} variable in $\tau+1^{th}$ iteration, m_{1v} as well as m_{2v} refers to the random numbers ranges from 0 to1 . The best and the worst candidate solution is represented as X_v^{best} and X_v^{worst} .

Substituting equation (16) in equation (12),

$$h_v^\tau = -\frac{X_v^{\tau+1} - m_{1v}X_v^{best} - m_{2v}X_v^{worst}}{1 - m_{1v} + m_{2v}}\left(e\sin(m_1)+\sin(m_1)\right) + e\sin(m_1)\mathrm{B}_v^\tau + \sin(m_1)\mathrm{A}_v^\tau \tag{17}$$

Substitute $X_v^{best} = A_v^\tau$ in the above equation,

$$h_v^\tau = \left(\frac{-X_v^{\tau+1}}{1 - m_{1v} + m_{2v}} - \frac{-m_{2v}X_v^{worst} - m_{1v}A_v^\tau}{1 - m_{1v} + m_{2v}}\right)\left(e\sin(m_1)+\sin(m_1)\right) + e\sin(m_1)\mathrm{B}_v^\tau + \sin(m_1)\mathrm{A}_v^\tau \tag{18}$$

Substitute equation (18) in equation (9), the solution becomes,

$$X_v^{\tau+1} = X_v^\tau - \left(e\sin(m_1)+\sin(m_1)\right)\left(\frac{X_v^{\tau+1}}{1 - m_{1v} + m_{2v}} + \frac{m_{2v}X_v^{worst} - m_{1v}A_v^\tau}{1 - m_{1v} + m_{2v}}\right) + e\sin(m_1)\mathrm{B}_v^\tau + \sin(m_1)\mathrm{A}_v^\tau \tag{19}$$

$$X_v^{\tau+1} = X_v^\tau - \frac{X_v^{\tau+1}}{1 - m_{1v} + m_{2v}}\left(e\sin(m_1)+\sin(m_1)\right) - \frac{m_{2v}X_v^{worst} - m_{1v}A_v^\tau}{1 - m_{1v} + m_{2v}}\left(e\sin(m_1)+\sin(m_1)\right) + e\sin(m_1)\mathrm{B}_v^\tau + \sin(m_1)\mathrm{A}_v^\tau \tag{20}$$

$$X_v^{\tau+1} + \frac{X_v^{\tau+1}}{1 - m_{1v} + m_{2v}}\left(e\sin(m_1)+\sin(m_1)\right) = X_v^\tau - \frac{m_{2v}X_v^{worst} - m_{1v}A_v^\tau}{1 - m_{1v} + m_{2v}}\left(e\sin(m_1)+\sin(m_1)\right) + e\sin(m_1)\mathrm{B}_v^\tau + \sin(m_1)\mathrm{A}_v^\tau \tag{21}$$

$$X_v^{\tau+1}\left(1 + \frac{e\sin(m_1)+\sin(m_1)}{1 - m_{1v} + m_{2v}}\right) = X_v^\tau - \frac{m_{2v}X_v^{worst} - m_{1v}A_v^\tau}{1 - m_{1v} + m_{2v}}\left(e\sin(m_1)+\sin(m_1)\right) + e\sin(m_1)\mathrm{B}_v^\tau + \sin(m_1)\mathrm{A}_v^\tau \tag{22}$$

$$X_v^{\tau+1}\left(\frac{1 - m_{1v} + m_{2v}}{1 - m_{1v} + m_{2v} + e\sin(m_1)+\sin(m_1)}\right) = X_v^\tau - \frac{m_{2v}X_v^{worst} - m_{1v}A_v^\tau}{1 - m_{1v} + m_{2v}}\left(e\sin(m_1)+\sin(m_1)\right) + e\sin(m_1)\mathrm{B}_v^\tau + \sin(m_1)\mathrm{A}_v^\tau \tag{23}$$

$$X_v^{\tau+1} = \frac{1 - m_{1v} + m_{2v}}{1 - m_{1v} + m_{2v} + e\sin(m_1)+\sin(m_1)}\left[X_v^\tau - \frac{m_{2v}X_v^{worst} - m_{1v}A_v^\tau}{1 - m_{1v} + m_{2v}}\left(e\sin(m_1)+\sin(m_1)\right) + e\sin(m_1)\mathrm{B}_v^\tau + \sin(m_1)\mathrm{A}_v^\tau\right] \tag{24}$$

where, $e^{\tau+1} = \beta e^\tau$. Here, α denotes the random vector ranging from $(0,1)$.

$$X_v^{\tau+1}=\begin{cases}\dfrac{1-m_{1v}+m_{2v}}{1-m_{1v}+m_{2v}+e\sin(m_1)+\sin(m_1)}\left[X_v^{\tau}-\dfrac{m_{2v}X_v^{worst}-m_{1v}A_v^{\tau}}{1-m_{1v}+m_{2v}}(e\sin(m_1)+\sin(m_1))+e\sin(m_1)\mathrm{B}_v^{\tau}+\sin(m_1)\mathrm{A}_v^{\tau}\right];m_2\le 0.5\\ \dfrac{1-m_{1v}+m_{2v}}{1-m_{1v}+m_{2v}+e\cos(m_1)+\cos(m_1)}\left[X_v^{\tau}-\dfrac{m_{2v}X_v^{worst}-m_{1v}A_v^{\tau}}{1-m_{1v}+m_{2v}}(e\cos(m_1)+\cos(m_1))+e\cos(m_1)\mathrm{B}_v^{\tau}+\cos(m_1)\mathrm{A}_v^{\tau}\right];m_{2:}>0.5\end{cases} \quad (25)$$

The term X_v^{best} refers to the ability of SSD-Jaya algorithm for better solution, and the term X_v^{worst} signifies the ability of the SSD-Jaya algorithm for avoiding worst solution. The above equation specifies the updated equation of the proposed SSD-Jaya, which in turn used to perform signature verification effectively. By integrating the parametric features of Jaya in SSD ensures the effectiveness of system performance.

Step 4: Compute the feasibility: The fitness of each search agent is calculated as follows: the lower error produces the best possible answer.

Step 5: Termination: The measures mentioned above are replicated before the iteration goal is reached or until a more creative approach is found. Proposed SSD-Jaya-based Deep CNN algorithm: pseudo code

Algorithm 1. Pseudo code of the proposed SSD-Jaya-based Deep CNN

Input: Position of search agents $X_v^{\tau};(1\le v\le c)$
Output: Best position
Procedure:
Begin
Initialize the agent's position and velocities
Assume the fitness function is minimum
While the stopping criteria are failed to met do
For all agents do
Calculate the fitness function using equation (8)
Arranged the agents based on the fitness value
Compute the mean global solution and the previous best solution
Update the agent's location using equation (25)
End if
End for
Check the feasibility of the solutions
Return the best solution
$\tau=\tau+1$
End while
Optimal solution is obtained
End

4. Discussion of results

The segment uses the SSD-Jaya-based Deep CNN and shows how the findings compare to other state-of-of-the-the-the-art approaches in terms of precision, sensitivity, granularity, and specificity

4.1 Experimental setup

The experimentation is done using handwritten-signature dataset [18], which consists of forged and genuine signature of 30 people. Here, each person has 5 Genuine signatures made themselves and 5 Forged signatures made by someone.

4.2 Performance metrics

Analyses, such as the precision, sensitivity, are shown below.

a) Accuracy: Accuracy is defined as the measure of accurateness based on the proposed SSD-Jaya-based Deep CNN, and is expressed as,

$$Accuracy = \frac{X + Y}{X + P + Y + O} \tag{26}$$

To have an accurate view of reality, you need to look at both the true positives and the negatives. The word describes a coin having a high odds of being heads and the possibility of being tails.

b) Sensitivity: Costiveness can be determined by using the expression of sensitivity seen below.

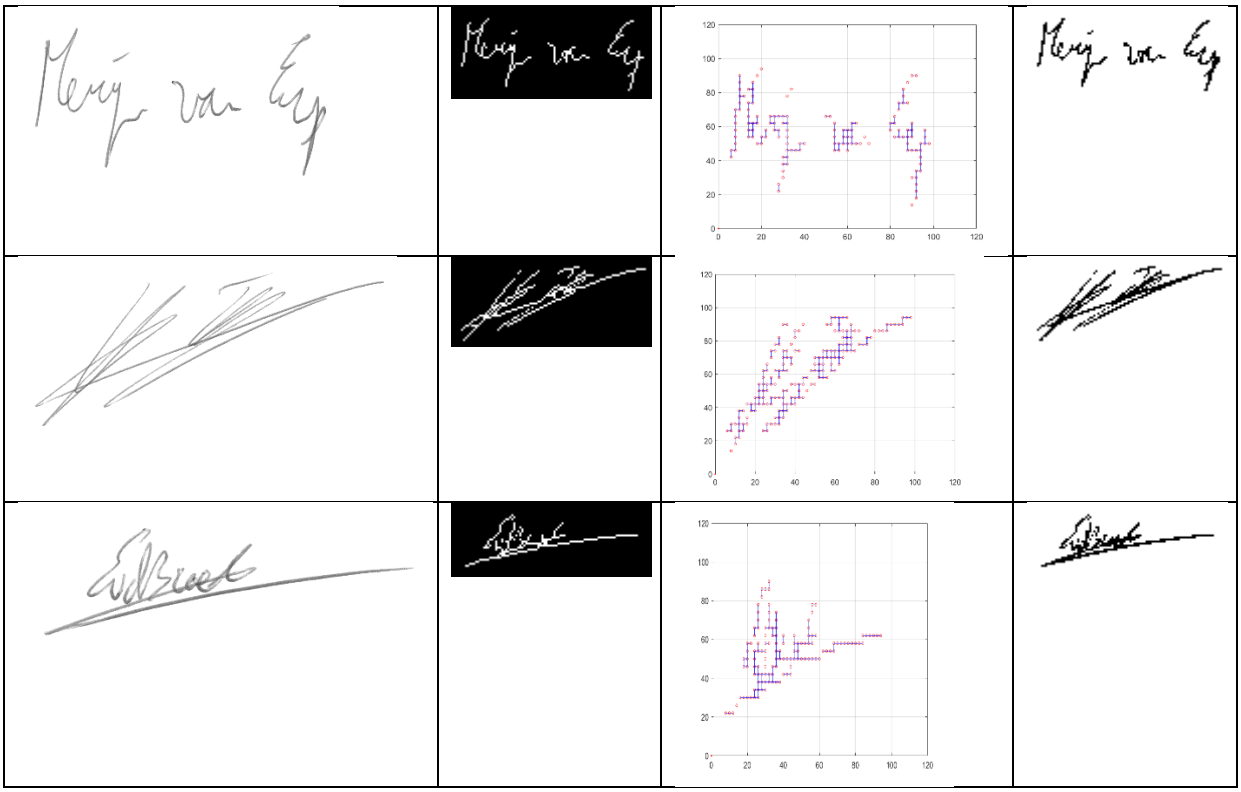
$$Sensitivity = \frac{Y}{P + Y} \tag{27}$$

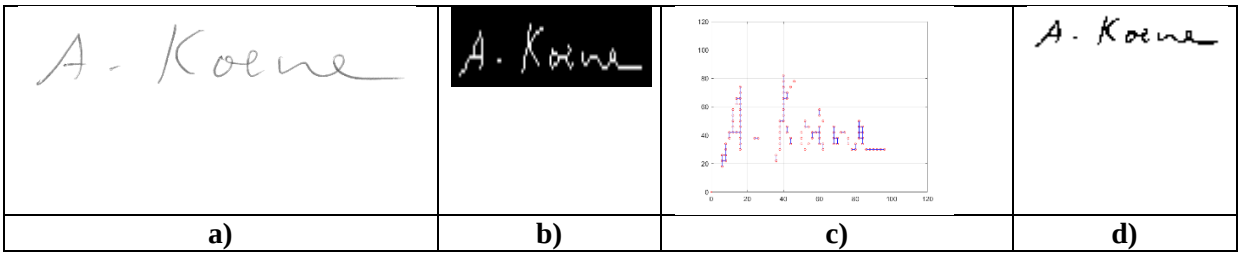
c) Specificity: The specificity is the calculation of the number of false negatives in an experiment is what may often be called the False Negativity. specificity is usually shows up as

$$Specificity = \frac{X}{X + O} \tag{28}$$

4.3 Experimental results

This segment shows the experimental results. 3 are seen in figure 3. An input picture is seen in Figure 3(a) and the skeleton of the figure 3b Notice (The graph in figure 3c is seen as image 2, and the image in figure 3d is labeled as image 2.)





a)	b)	c)	d)

Figure 3. Sample results a) Input image, b) Skeleton image, c) Graph image, d) Threshold image

4.4 Performance analysis

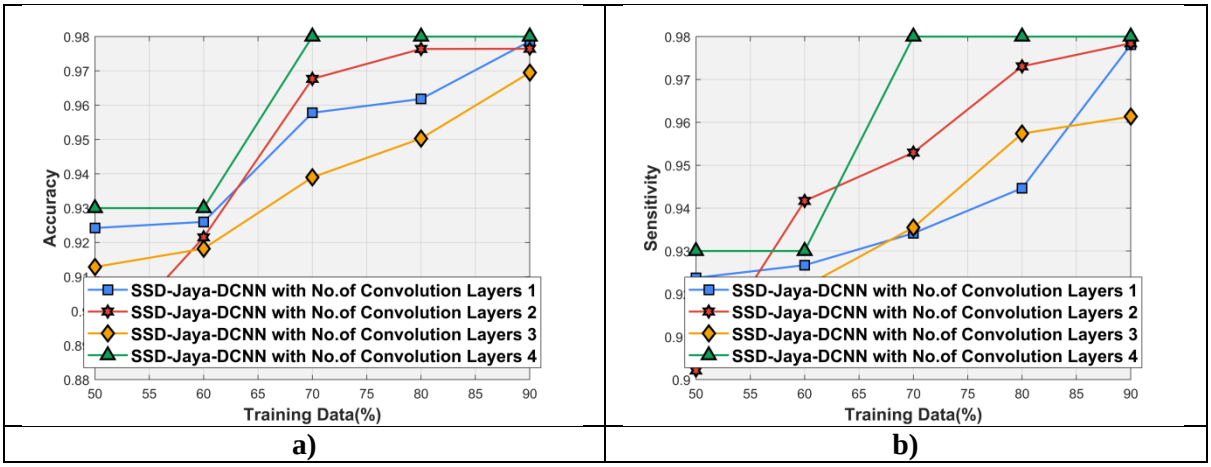
In this part, we vary the training examples percentage to see how it impacts model efficiency and K-Fold. It is calculated using the dimensions.

4.4.1. Analysis with respect to training data percentage

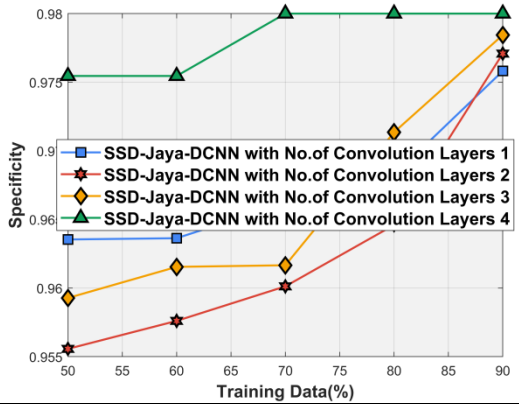
In Figure 4, we see how the model has been built out with various convolutionalent layers. The graph in Figure 4 depicts the analysis dependent on various training percentage. when 50% of training is complete, the values of the model proposed with layer 1, 2, and 3, respectively, are 0.24, 0.886, and 0.98. model results using the first four convolutional layers are 1.92, 0.21, 0.92, and 0.98. Similarly, for 70% training results, the 0.967 value of the 0.57 for SSD Jaya CNN with conv layer 1 is equivalent to 0.957 with the 1,0.9, and the 0.9 and the 0.38 for SSD Jaya CNN with conv layer 2 are almost the same, although the 0.67 and 0.3 values for SSD Jaya dependent CNN are vastly different. computed by the construct with the first, second, third, and fourth convolutional layers, respectively

In figure 4b, the study is dependent on various training data sensitivity. Training output for the model of 1, 2, 3, and 4 convolutional layers amounts of data is 0.23, 0.9, and 0.06 respectively Data percentage is calculated with layers 1, 2, 3, and 4 for this approach: 0.934, 0.952, and 0.98 For 80% training results, the sensitivities are 0.944, 0.73, and 0.98, which come from Jaya Deep CNNs using 1, 2, 3, and 4 respectively. The suggested model computes the percentages of the training data as 0.978, 0.2, 0.2, and 0.1, respectively.

More details and metrics (specificity) were needed for training with different percentages as shown in Figure 4c. the accuracy of the suggested model of convolution layer 1, 2, and 3, respectively. Specificity values of 0.967, 0.96, and 0.98 are calculated for the final model while the training data ratio is 70% For 80% training results, the specificity values are 0.968, 0.964, and 0.98 for Jaya Deep CNN; so conv layer 1 is 0, then conv layers 2 and 3 have values of 0.964, and 0.98, and the 4 is 0.71. When the training data percentage is 90, the resulting model specificity values are 0.975, 0.9, 0.98, and 0.98, the suggested model will be optimal for just the first three convolutional layers.



a)	b)



c)

Figure 4. Performance analysis in terms of training percentage with convolution layers (a) accuracy, (b) sensitivity, (c) and specificity

4.6 Comparative techniques

The methods, like KNN [1], SVM [16], and the ANN [17] are utilized for the comparison with the developed SSD Jaya-based DCNN for the analysis.

4.6.1 Analysis of the signature verification with respect to training data percentage

The precision, sensitivity, and specificity comparison is shown in Figure 5. The figures in Figure 5a demonstrate the effect of training for various data percentages. The training data percentage for the Jaya-based DCNN was 0.886, 0.61, and 0.92. For the training results, the latest strategies, including K Nearest Neighbour Vector, Support Vector Machine, and Artificial Neural Networks have a recorded accuracy of 0.889, 0.88, and 0.61. The built model gained the overall accuracy of 0.92 Similarly, when the training data went to 80%, the approaches, KNN, and SVM, both achieved a slightly better level of accuracy than before development; namely, they were around 0.9 and 0.8, respectively. when the training data percentage is 90, the K-Nearest-Neighbors, Support Vector Machines, and artificial neural networks reach their predicted values of 0.948, 0.47, and 0.97;

in a more intense shade (Figure 5b), as in this comparative study When the training data percentage is set to 50, the values that are found by the K-Nearest Neighbour, Support Vector Machine, and generalization models are 0.888, 0.72, and 0.22. Additionally, for training data of 60%, techniques like KNN and SVM possess 0.888 and 0.72, which are comparatively less sensitive than Jaya-based DCNN. Aproceeding the same training data acquisition, the DCNN finished with the calculated SSD specificity of 0.22 Similarly, as the training data improved to 70%, the approaches, KNN, and SVM, reached the midpoint, were at 0.32, and got better; the evolved process, on the other hand, had a final sensitivity of 0.72. Sensitivity values calculated by KNN, SVM, and the proposed model are 0.36, 0.82, and 0.72 respectively for 80% of training results.

The data in Figure 5c is defined in terms of its specificity. The latest methods, including KNN and SVM, have a predictive power of 0.31 and 0.9, while DCNN holds a lower specificdicsoricty of 0.3.9. Jaya acquired the precision of the evolved neural network at 0.970. For example, the trained set rose method (with 60% of the sample) has a value of 0.31, the SVM (with 9% of the instances) and the evolved method (with 9% of the sample) has a value of 0.970. When the training data percentage is 0.7, the KNN, SVM, and the alternative model's suggested values are 0.96, it has a precision of 0.1 when the training data percentage is held at 80, the SVM, KNN, and the other two models have accuracy values of 0.95, 0.66, and 0.97.

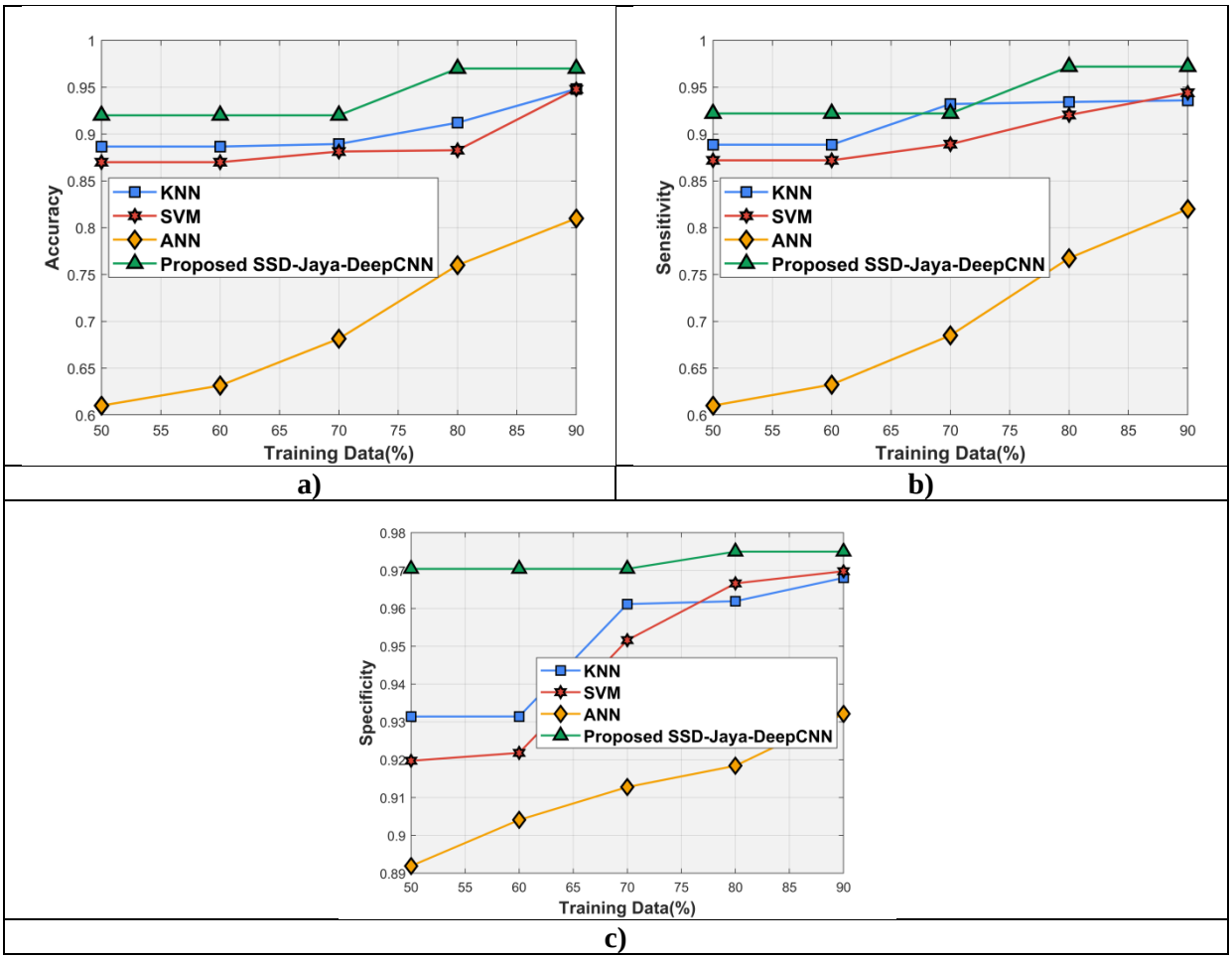


Figure 5. Comparative analysis in terms of training percentage (a) accuracy, (b) sensitivity, (c) and specificity

4.6.2 Analysis of the signature verification based on Hierarchical skeleton features

Figure 6 depicts the study in terms of precision, sensitivity, and specificity as applied to signatures. The figure in this diagram (shown in Section 6) shows the results for various percentages of training data set. Skeleton +KNN trained on existing data results, 0.33, SVM training, and 0.936, and Hierarchical +KNN (0.33/0.36 = 0.916) results, have an accuracy of 0.66. for the training results, there are techniques like Hierarchical KNN, Hierarchical SVM, and Whole skeleton, with a relative accuracy of 0.666, respectively While the SS Jaya-based Hierarchical skeleton DCNN's accuracy was similar to that of 0.98 Similarly, as the training results improved to 80%, the Jaya-based DCNN achieved a precision of 0.733, 0.988, and the D Hierarchical DCNN had a superior performance of 0.98. Training data percentage is 0.7 for KNN, 0.833 for SVM, and 0.98 for HierarchicalDCNN and HierarchCNN was used to estimate a prediction.

It is in Figure 6 where the comparative study of sensitivity is seen (Figure 6b that presents the sensitivity analysis). When the training data proportion is 0.5, the Jaya-based DCNN sensitivities are 0.882, 0.35, and 0.96, respectively. According to 60% of the training results, the current strategies, Hierarchical skeleton has 0.884 sensitivity, which is considerably lower than its Jaya-based cousin, Hierarchical Skeleton +DCN. The SSD Hierarchical skeleton, which uses the Hierarchical algorithm with a word distance-based neural network has the same intrinsic sensitivity of 0.66 Since the training data are increased to 70%, the processes, Hierarchical skeleton +KNN, Hierarchical skeleton +S 0.899, and "SVC" have reached their respective accuracies of 0.899, 0.938, and 0.97, while the "SVC" +SR system is at 97. Here, it can be said that the sensitivity values computed using the data from

skeletal +KNN, the data from the factorialK, the hierarchical skeletal +ANN, and the factorial hierarchicalS method are 0.13, 0.66, and 0.97.

You will see the precision of the findings in Figure 6c. The training data accuracy of S +KNN, SVM, and Hierarchical S+ANN have a score of 0.9 against Hierarchical HDSDCNN, which is lower than its precision, of 0.952 while training the Hierarchical Hierarchical +SSD and the SSD Jaya-based DCNN on the same training results, they were able to achieve a precision of 0.975. Similarly, as the training data were 60%, the processes, hierarchy dependent DCNN, hierarchy based KNN, and Skeletal hierarchy +ANN have a specificity of 0.26, and Skeletal +SVM has a specificity of 0.9.77. When the training data percentage is set to 70, the calculated hierarchy specificities are 0.27, 0.9 and 0.98, respectively for Hierarchical skeleton+ANN, Hierarchical skeleton+KNN, and Hierarchical+ANN, and skeleton-based DCNN. The precision values, calculated by SKELE +KNN, SLE +SVM, and HSAK-based DCNN, are 0.34, 0.55, and 0.9, respectively.

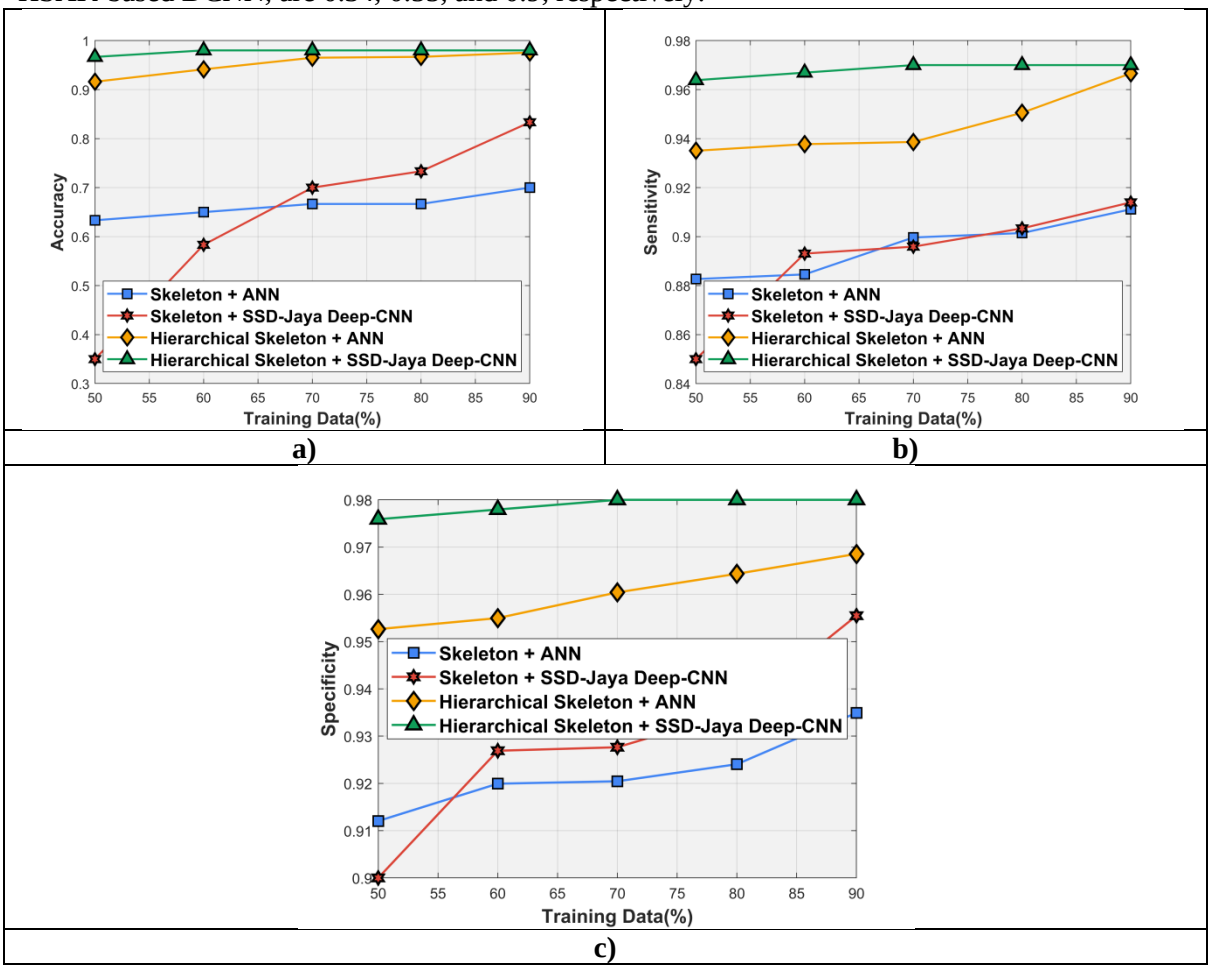


Figure 6. Comparative analysis with respect to features in terms of training data percentage

(a) accuracy, (b) sensitivity, (c) and specificity

4.7. Comparative discussion

4.7.1 Analysis in terms of training data percentage

Comparative discussion of the KNN, SVM, and ANN, and the proposed SSD in terms of precision, and sensitivity is presented in Table 1. proposed S2D3DCNN has a limit of 0.73 with respect to parameters, while current KNN, SVM, and ANN provide appropriate accuracy levels of 0.48, 0.9, and 0.86 To improve system sensitivity, a value of 0.72 was assigned to the proposed SSD, as against current value of 0.36, 0.44, and 0.9, respectively. The suggested SSD approach results in a higher precision than the current approaches of 0.975, the SVM 0.969, and the ANN, respectively.

Table 1 Comparative analysis with respect to training data percentage

Methods	Accuracy	Sensitivity	Specificity
KNN	0.948	0.936	0.968
SVM	0.947	0.944	0.969
ANN	0.86	0.82	0.932
Proposed SSD Jaya-based DCNN	**0.973**	**0.972**	**0.975**

5. Conclusion

This paper proposes a signature-based DCNN that is built around the goal of building a Jaya signature. First, the input picture is processed using resizing and the Otsu thresholding techniques. The data is sent to the function extraction stage prior to pre-processing. This skeleton-based attribute extraction system gets the best features for signatures. Since pixel values are used for features extraction, detection rate is increased. If the characteristics have been classified, the analysis is performed by using a deep CNN. For speed and flexibility, the deep CNN was trained using the SSD algorithm. The SSD-Jaya optimization is very much like the Jaya algorithm; it basically the weighting function of the classifier is built on top of the prior work done by the SSD algorithm and improved. With regard to the extracted results, the proposed signature authentication model is better than current techniques. the model's accuracy is estimated using the captured signature's sensitivity and specificity are used to determine its level of effectiveness This results show both the greatest possible level of precision (0.972) as well as well as the maximum possible sensitivity (0.975).

References

[1] Aini Najwa Azmi, Dewi Nasien, and Fakhrul SyakirinOmar,"Biometric signature verification system based on free man chain code and k-nearest neighbor", Multimedia Tools and Applications, vol.76, no.14, pp.15341-15355, 2017.

[2] Luiz G. Hafemann, Robert Sabourin, and Luiz S. Oliveira,"Characterizing and evaluating adversarial examples for Offline Handwritten Signature Verification", IEEE Transactions on Information Forensics and Security, vol.14, no.8, pp.2153-2166, 2019.

[3] Ankan Kumar Bhunia, Alireza Alaei, and Partha Pratim Roy,"Signature verification approach using fusion of hybrid texture features", Neural Computing and Applications, pp.1-12, 2019.

[4] Antonio Parziale, Moises Diaz, Miguel A. Ferrer, Angelo Marcelli,"SM-DTW: Stability Modulated Dynamic Time Warping for signature verification", Pattern Recognition Letters, vol.121, pp.113-122, 2019.

[5] Xinghua Xia, Xiaoyu Song, Fangun Luan, Jungang Zheng, Zhili Chen, and Xiaofu Ma, "Discriminative feature selection for on-line signature verification", Pattern Recognition, vol. 74, pp.422–433, 2018.

[6] Om Prakash Patel, Aruna Tiwari, Rishabh Chaudhary, Sai Vidyaranya Nuthalapati, Neha Bharill, Mukesh Prasad, Farookh Khadeer Hussain, and Omar Khadeer Hussain, "Enhanced quantum-based neural network learning and its application to signature verification", Soft Computing, vol.23, no.9, pp.3067-3080, 2019.

[7] Marcin Zalasinki, and Krzysztof Cpalka,"A new method for signature verification based on selection of the most important partitions of the dynamic signature", Neurocomputing, 289, pp.13-22, 2018.

[8] Marcin Zalasiński, Krzysztof Cpałka and Leszek Rutkowski, "Fuzzy-Genetic Approach to Identity Verification Using a Handwritten Signature", in Advances in Data Analysis with Computational Intelligence Methods, Springer, Cham, pp. 375-394, 2018.

[9] Donato Impedovo and Giuseppe Pirlo,"Automatic Signature Verification: The State of the Art", IEEE transactions on systems, man, and cybernetics—part c: applications and reviews, vol. 38, no. 5, September 2008.
[10] D. Impedovo, G. Pirlo, M. Russo, "Recent Advances in Offline Signature Identification", in proceedings of 14th International Conference on Frontiers in Handwriting Recognition, 2014.
[11] Luiz G. Hafemann, Robert Sabourin, and Luiz S. Oliveira,"Offline Handwritten Signature Verification-Literature Review", in proceedings of Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA), pp. 1-8, 2017.
[12] Anil K. Jain, Arun Ross, and Salil Prabhakar, "An Introduction to Biometric Recognition", IEEE transactions on circuits and systems for video technology, vol. 14, no. 1, January 2004.
[13] RejeanPlamondon, and Sargur N. Srihari, "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey", IEEE transactions on pattern analysis and machine intelligence, vol. 22, no. 1. January 2000.
[14] Alaa Tharwat, and Thomas Gabel, "Parameters optimization of support vector machines for imbalanced data using social ski driver algorithm", Neural Computing and Applications, pp.1-14, March 2018.
[15] R. Venkata Rao,"Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems", International Journal of Industrial Engineering Computations, vol. 7, pp. 19–34, 2016.
[16] Jamileh Yousefi, "Image Binarization using Otsu Thresholding Algorithm", University of Guelph, Ontario, Canada, 2011.
[17] Cemil Oz, "Signature Recognition and Verification with Artificial Neural Network Using Moment Invariant Method ",in proceedings of International Symposium on Neural Networks, pp. 195-202, May 2005.
[18] Handwritten-signature dataset, "https://www.kaggle.com/divyanshrai/handwritten-signatures", accessed on January 2020.
[19] Cong Yang, Oliver Tiebe, Kimiaki Shirahama, and Marcin Grzegorzek, "Object matching with hierarchical skeletons", Pattern Recognition, vol.55, pp.183-197, 2016.
[20] FengbinTu, Shouyi Yin, Peng Ouyang, ShibinTang,Leibo Liu, and Shaojun Wei, "Deep Convolutional Neural Network Architecture with Reconfigurable Computation Patterns", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.25, no.8, pp.2220-2233, 2017.
[21] H. Zaboli ; M. Rahmati, " An Improved Shock Graph Approach for Shape Recognition and Retrieval", in Proceedings of the First Asia International Conference on Modelling & Simulation (AMS'07), pp.27-30 March 2007.
[22] Gopichand G, Sailaja G, N. Venkata Vinod Kumar, T. Samatha,"Digital Signature Verification Using ArtificialNeural Networks", International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, vol.7, no.6S, March 2019.
[23] Margit Antal , LaszloZsolt Szabo, and Tunde Tordai,"Online Signature Verification on MOBISIG Finger - DrawnSignature Corpus", Mobile Information Systems, pp.15, 2018.
[24] Alona Levy, Ben Nassi, Yuval Elovici, and Erez Shmueli,"Handwritten Signature Verification Using Wrist-Worn Devices", in Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, September 2018.
[25] Luiz G. Hafemann, Robert Sabourin, Luiz S. Oliveira, "Learning Features for Offline Handwritten Signature Verification using Deep Convolutional Neural Networks", Pattern Recognition, vol.70, pp.163-176, 2017.