International Journal of Advanced Science and Research
www.allsciencejournal.com
ISSN: 2455-4227
Received: 19-07-2025, Accepted: 19-08-2025, Published: 03-09-2025
Volume 10, Issue 2, 2025, Page No. 1-9

Hyper-heuristic firefly algorithm-based convolutional neural network for handwritten identification system

Locksley EA[1], Olabiyisi SO[1], Ganiyu RA[2], Omidiora EO[2], Taiwo CY[2]
[1] Department of Computer Science, Ladoke Akintola University of Technology, Ogbomoso, Nigeria
[2] Department of Computer Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria

Abstract
The goal of this study is to create a convolutional neural network for a handwritten identification system based on the hyper-heuristic firefly algorithm. Samples of handwriting were gathered from Ladoke Akintola University employees and students in Ogbomoso, Nigeria. The CNN's hyperparameter is chosen by the Hyper Heuristic Firefly Algorithm (HHFA). The MATLAB 2020 environment was used to implement the improved model. False Positive Rate (FPR), accuracy, sensitivity, specificity, precision, and computation time were used to assess the created model. The efficacy of the CNN model based on the hyperheuristic firefly algorithm was evaluated using the paired-sample t-test. To determine the performance differences between the hyper-heuristic firefly method and the current CNN at $P < 0.5$, a hypothesis was established. It was discovered that there is a notable distinction between HHFA-CNN and the existing CNN algorithms. The hyper-heuristic firefly algorithm was therefore suggested as an effective tool in the handwritten identification system.

Keywords: Hyper heuristic firefly algorithm, CNN, handwritten character recognition, hyperparameter optimization, forensic ai, swarm intelligence

Introduction

Handwritten Character Recognition (HCR) allows moving various types of documents into a searchable, editable and assessable data form. The end goal of HCR is to reproduce human literacy in a form that enables robots to read, edit and engage with text at the same speed as humans. Over 50 years of approaches to the discovery of HCR attracted many researchers, and this led to a number of developments in this sphere. New technology developments have received much interest in focusing on handwriting recognition using vision sensors to track the position and movements of 3D fingers moving within the space to write. Nevertheless, due to the overwhelming selection of categories of the characters, the presence of a lot of the same characters and a high variety of handwrites, the performance of HCR has not improved significantly over the past few years (Khandokar *et al.*, 2021) [1, 10].

Identification based on handwritten material is becoming important to forensic investigators and law enforcement, as well as other organisations. Giving credible evidence in a court of law is helpful in solving cases of fraud, forgery, and other criminal activity. One of the primary issues that any handwritten recognition language system should address is the variance in sizes of fonts and letters in handwriting, which varies across different individuals (Sahloul and Suen, 2014) [2]. During data collection, many variants of the handwriting of the same person are observed in studies (Almansari and Hashim, 2019) [3], and this aspect of automated letter identification is a frequent issue (Mustapha *et al.*, 2022) [4]. Moreover, handwriting is very diverse; there are no large publicly available sets, and it takes different forms in different people, which makes it difficult to recognise (Al-Helali and Mahmoud, 2017) [5].

The Convolutional Neural Network (CNN) is a famous deep learning architecture, inspired mainly by the visual perception system of the human brain. CNN studies have rapidly been adopted and reached state-of-the-art performance in a wide range of applications, including image classification, text detection, pose estimation, object tracking, action detection, visual saliency detection, scene labelling, speech processing, and natural language processing. This is attributed to the exponentially growing number of annotated data and the graphics processing unit (GPU) advancements. The CNN architectures may take any form, but they have the same general components. These components consist of three types of layers, namely, convolutional ones, pooling ones, and fully connected ones (Hossain *et al.*, 2021) [6].

One of the most widespread methods to enhance handwriting recognition accuracy uses Convolutional Neural Networks (CNNs). The CNN model consists of several kernels along with nonlinear functions and pooling layers. The methodology addresses the problem of tuning weights and connecting nodes of the neural network (NN) in order to generate an appropriate channel to process both time and space data. However, CNN training is rather a complicated issue. In recent times, optimisation has enhanced the weights as well as the biases of CNN since numerous techniques of optimisation are available through which the weights may be improved and optimised. Another optimisation algorithm is the hyperheuristics Firefly Algorithm (HHFA) (Khosravi and Chalechale, 2022) [7].

This paper aims to develop a hyper-heuristic Firefly Algorithm-based CNN for a handwritten identification system (HHFA-CNN). While the objectives are to formulate a hyperheuristic firefly algorithm to improve on the ordinary firefly algorithm's exploitation and exploration imbalance, integrate the HHFA with CNN to improve general performance, implement the HHFA-CNN on MATLAB 2020a and evaluate and compare the performance of HHFA-CNN with the existing CNN using false positive rate, sensitivity, accuracy, computation time, precision and specificity.

Review of Literature

1. The basic structure of a convolutional network

The states within any convolutional neural network are organised in a grid manner. Given that the value of every feature is determined by a small localised place in the previous layer, these spatial relations are transferred to the following layer. As the process of convolution and the transition to the next layer is highly dependent on those spatial connections, it is essential to keep such connections between the grid cells. The layers of the convolutional network are of a three-dimensional grid where the dimensions include the height, breadth, and depth.

The depth of a convolutional neural network layer is not to be conflated with the overall depth of the network. When a single layer is discussed, the depth is defined as the number of channels of the layer, e.g., the number of main colour channels (like red, blue, and green) in the input picture or the number of feature maps in the hidden layers. It is unfortunate that the number of feature maps in a layer, as well as the number of layers in convolutional networks, are both described as the depth. We will, nevertheless, apply this term in a manner that its context would clarify it (Johnson *et al*., 2015) [12].

A simple convolutional neural network consists of three layers: a convolutional layer, a pooling layer and an output layer. The pooling layer is occasionally optional. The typical architecture of the convolutional neural network composed of three convolutional layers as shown in figure 1, is perfectly applicable to the task of classification of the handwriting images. This system consists of the input layer, numerous hidden layers (convolutional, normalisation, and pooling repetitions), a fully connected layer, and an output layer (Ahlawat *et al.*, 2020) [13].

Those neurons of one layer are connected to a small number of neurons of the next layer in order to help the upsizing of the better-quality pictures. The pooling/sub-sampling can be used to reduce the size of the input to a squeeze. The input image can be considered a set of small sub-regions, which were denoted as the receptive fields of a CNN architecture. The input layer possesses a convolutional mathematical operation of modelling the response into the next layer. The response is a visual stimulus per se. This is fully described here:

1. Input Layer

In the input layer, the input data is loaded and saved. This layer provides the input image's height, width, and number of channels (also known as RGB information) (Ahlawat *et al.*, 2020) [13].

2. Hidden Layer

The foundation of the CNN architecture is made up of hidden layers. They employ a number of convolutions, pooling, and activation functions in their feature extraction technique. At this point, the distinctive characteristics of handwritten numbers are identified (Ahlawat *et al.*, 2020) [13].

3. Convolutional Layer

The initial layer on the input image is the convolution layer. It is deployed to obtain the character of an image. The *n x n* input neurons in the input layer output *(n-m + 1) x (n-m + 1)* after being merged with an *m x m* filter. Non-linearity was injected through a neural activation function. The receptive field, stride, dilation, and padding are the main elements of the convolutional layer as described in the paragraph below. Animal visual brain is a source of inspiration to CNN computing (Eickenberg *et al.*, 2017) [15]. The part of the brain where the information received by the retina is translated is referred to as the visual cortex. It is responsive to small sub-regions of the input and visual information. Similarly, a CNN calculates the receptive field, a tiny zone of an input picture that can determine a specific network region. It is also used to figure out other CNN properties and is among the essential design specifics of the CNN architecture (Le and Borji, 2018) [14]. It works in the same way that the foveal vision of the human eye presents sharp centre transparency and is the same magnitude as the kernel. The stride, pooling of CNN, the size of the kernel, and depth have impacts on the receptive field (Luo *et al.*, 2017). The terms receptive field (r), effective receptive field (ERF) and projective field (PF) are used in calculating effective sub-regions in a network. Ahlawat *et al.* (2020) [13] define the PF as a number of neurons, which neurons use as targets to send their output and the ERF as the area of the original image, which affects the activity level of a neuron.

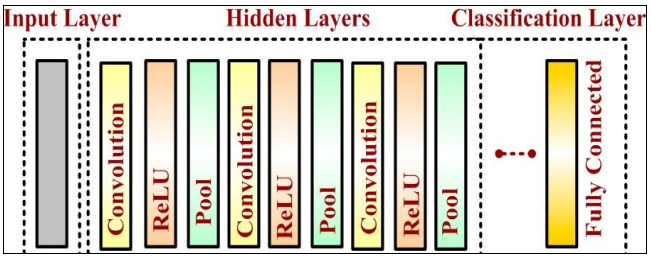

Fig1: Typical Convolutional Neural Network Architecture (Ahlawat *et. al.*, 2020) [13]

Hyper-heuristic firefly algorithm

The hyperheuristic firefly algorithm is a species of optimisation algorithm, in which the concepts of firefly algorithms have been combined with hyperheuristics. To solve complex problems, a family of methods called hyperheuristics automate the selection, combination, or generation of low-level heuristics. Conversely, firefly algorithms form part of swarm intelligence algorithms that help in optimising a problem through emulating the actions of fireflies. The two methods are integrated in the hyperheuristic firefly algorithm, where the hyperheuristic firefly algorithm optimises the problem by using firefly algorithms, optimising the matter with low-level generated and selected heuristics via hyperheuristics. The algorithm builds an initial population of fireflies and then chooses the best according to their fitness with the help of a hyperheuristic approach. New heuristics are then formed with the selected firefly and re-evaluated on the basis of the fitness function. It is iteratively applied, hoping to find a decent solution, and the best heuristics are stored to be applied to the subsequent iteration (Wei *et al.*, 2019).

The hyperheuristic firefly algorithm has been applied in other applications, which include optimisation problems in the engineering, financial and logistics domains. It has been found to succeed and find better answers to complex issues, where more traditional approaches to optimisation may not yield results. However, as in all optimisations, the techniques, the quality of the fitness that is used to rate the potential solutions, and the specific problem in question

define how effective the algorithms perform (Sun *et al.*, 2021).

The HHFA is the combination of hyper-heuristics and the multi-objective firefly algorithm. The hyper-heuristic form of optimisation framework employs both high-level and low-level strategies to enhance the optimisation function of the firefly algorithm. The low-level approach examines the problem and forms the yardstick by which solutions are selected. To increase the number of potential solutions, a single or more solutions are then considered, integrated or changed to form a new set of solutions. According to the principles generated by the low-level strategy, the high-level strategy initiates the heuristic search process to select the solutions out of the set of potential solutions (Aswanandini and Deepa, 2021).

The set of rules that addresses the combinatorial part of the problem, consisting of formulating rules that serve to solve each instance of a situation that is chosen, is covered under the regulations at the low heuristic level. By using one or more solutions and mixing them or modifying them through several search methods, a new group of solutions is formed. The FA-based search method is one of the search techniques applied in this piece to offer new solutions. The strategic policy that operates at a higher level emulates the selection process after the innovative solutions have been created. The high-level strategy automates the heuristic selection, and that is where each heuristic is chosen alone and then applied to the solutions. A heuristic selection process is carried out online to select the heuristics out of the set of heuristics currently being generated by the rules generated by the low-level strategy. The empirical reward and the variables of the confidence level are the main measures that determine the usefulness of the heuristics. The empirical reward is based on the performance in the past, whereas the heuristic frequency construction demonstrates the level of confidence. These two are applied to judge whether the heuristics are suitable and applicable under the current operational state or not. Consequently, the firefly foraging process uses feasible heuristics to the solutions (Yang and He, 2013).

The heuristics are initialised as the population of fireflies x_i (i = 1, 2, z). In the case where the brightest firefly at location x represents the optimum solution, this can be fashioned in the form of a maximisation problem *I(x) ~ f (x)*, where *I(x)* is the light intensity and *f(x)* the fitness function. The error rate will be used as the fitness in this study, and the problem is hence translated into a minimisation problem as given in Equation 1.

$$I(x) = \begin{cases} \frac{1}{f(x)}, & if\ f(x) > 0 \\ 1 + |f(x)|, & otherwise \end{cases} \quad (1)$$

This algorithm is founded on the light intensity *I(x)* and the attractiveness function *(β)*. As determined by Equation 2, as the brightness and appealing power of light raises, the separation *(r)* between the source firefly and destination firefly lowers.

$$I(r) = \frac{I_0}{1+\gamma r^2} \quad (2)$$

Here I_0 denotes the intensity of the source light, and gamma denotes the absorption coefficient. This can be suggested by the Gaussian form as given in Equation 3.

$$I(r) = I_0 \exp(-\gamma r^2) \quad (3)$$

The attractiveness β is proportional to the light intensity seen by the adjacent fireflies. Hence, the attractiveness β is given as proportional to the light intensity of the solutions as described in Equation 4.

$$\beta(r) = \beta_0 \exp(-\gamma r^m) \quad (4)$$

Where β_0 is the attractiveness at r=0 and m is the number of iterations.

In CNN optimisation, the computational complexity has to be lower, and hence the resource utilisation has to be lower. Thus, an expression of attractiveness is changed to a practically applicable form expressed in Equation 5.

$$\beta(r) = \frac{\beta_0}{1+\gamma r^2} \quad (5)$$

The distance between any two fireflies (nodes) i and j positioned at x_i and x_j is denoted as$r_{i,j}$, which is computed as the Cartesian distance measure as defined in Equation 6.

$$r_{i,j} = \sqrt{\sum (x_{i,k} - x_{j,k})^2} \quad (6)$$

Where $x_{i,k}$ and $x_{j,k}$ are the Cartesian points of xi and x j is the number of dimensions. The firefly moves towards the best firefly, and this location is updated after each iteration using Equation 7.

$$x_i = x_i + \beta_0 e^{-\gamma r^2}(x_j - x_i) + \alpha(rand - \frac{1}{2}) \quad (7)$$

Here, rand is the pseudo-random number in the range [0, 1], and α is the step-size control parameter. For optimal outcomes, the values of α and β0 are adjusted to α ∈ [0, 1] and $\beta_0 = 0.2$.

The heuristic is applied to each of the solutions obtained by the firefly algorithm, based on the light intensity and the attractiveness of the firefly. The firefly, which is returned as the global best solution, contains the solution to be applied. The heuristic is applied with the selected solution to form a new set of solutions. At this stage, serial scheduling and double justification are employed. Serial scheduling is used to select the solutions without interleaving the feasible solutions. Likewise, the double rationale is a simple local search technique that searches the solutions with exact shifting to control the search quality. The new solutions are compared, and then they are analysed by their properties. This analysis, in terms of configuration, determines whether to include them in the existing set of solutions or terminate them to accommodate newer solutions from the next iteration (Aswanandini and Deepa, 2021).

After the formation of new solutions by low-level heuristics and the selection by the high-level strategy, they are saved in the non-dominated set of solutions in the archive. The non-dominated sorting procedure is used to classify the archive, creating several levels to retain the newer solutions. The first level is given to the solution with high priority, and the next level will be given to the second-best priority, and vice versa. The HHFA selects solutions from this archive based on the Pareto front and returns the best configuration as the final solution.

2. Related Works

To develop a revolutionary approach to Tamil handwritten character recognition, Lincy and Gayathri (2021) [8] involved two fundamental processes: preprocessing and

recognition. Preprocessing stage includes RGB to greyscale conversion; binarisation of thresholding, picture complementation, morphological operations and linearisation. The picture is linearised and then recognised using an effectively constructed Convolutional Neural Network (CNN). To be more exact, a new Self-Adaptive Lion Algorithm (SALA), as the conceptual enhancement of the traditional Lion Algorithm (LA), tunes the fully linked layer and weights. In terms of specific performance indicators, the work proposed is compared and presented as being better than that of other innovative models.

To implement security of massive data, Aswadinni and Deepa (2021) presented a hyperheuristic artificial neural network using the firefly algorithm. They developed an intrusion-detecting model that could easily categorise network-based intrusions and host-based intrusions with no complexity issues. An optimised deep learning technique was developed (a hyperheuristic firefly technique-based convolutional neural network, HHFA-CNN) to allow IDS to learn effectively. This proposed method was tested based on two data sets, NSL-KDD and ISCX-IDS. The findings indicated that the proposed HHFA-CNN model was more effective than other models which are in use. The accuracy value of the proposed model is 96.6667%.

Khandokar *et al.* (2021) [1, 10] have examined both the ability of CNN to recognise characters in a set of pictures and the level of accuracy in the course of testing and training. CNN can identify the characters by considering the shapes and also comparing them with the characteristic features that render the characters unique. To determine the accuracy of the handwritten characters, the CNN implementation is tried with the NIST dataset. Based on the test results, the accuracy achievable on 200 photographs using a training set of 1000 NIST pictures is 92.91 per cent.

Saqib *et al.* (2022) [11] trained a customised CNN model on handwritten images of two different datasets, Kaggle and MNIST, respectively, achieving high accuracy but keeping the size of the model small. The two best of twelve generated models, irrespective of hyperparameters, are recommended so as to determine which models can yield optimal accuracy on a specific dataset. Moreover, both the precision, recall, specificity, and F1 score of the performance matrices of the two proposed models are considered when looking into the classification reports (CRs) of these two proposed models, produced by the confusion matrices (CMs) generated on the given dataset and presented subsequently. In order to reproduce a real-life scenario and enhance the understanding of the output of the models, three more averages of the F measurement (micro, macro, and weighted) are calculated. The model with an optimiser of "RMSprop" performs the best in the digit recognition with a maximum detection accuracy of 99.642 at a learning rate of 0.001 and the model with an optimiser of "ADAM" performs the best in the alphabet recognition with the maximum detection accuracy of 99.563 at a learning rate of 0.00001. In digit and alphabet recognition, the best two models gave weighted F1 and macro F1 of 0.996, 0.998 and 0.997:0.992, respectively.

Methodology

The development of a hyper-heuristic zebra optimisation firefly algorithm-based convolutional neural network for a handwritten identification system involves several steps. Figure 3.1 shows the schematic diagram of the developed model. Here are the steps that were taken into consideration in this study:

1. **Data Collection:** The first step was to collect the handwritten data that was used to train and test the system. This involved collecting data from multiple sources, including public datasets, online sources, and handwritten samples from individuals.

2. **Data Preprocessing:** After the data has been collected, the next step is to preprocess it, prepare it for use in the system. This involved tasks such as normalisation, resizing, noise reduction, and feature extraction.
3. Formulate an Enhanced Hyper-Heuristic Firefly Algorithm (HHFA). This was done to select the best set of hyperparameters for the CNN automatically. The hyperparameters include filter size, number of layers, batch size, and number of filters.
4. Development of an HHFA-based CNN (HHFA-CNN) model for a handwritten identification system. This was used as feature extraction and classification. The essence of this step was to train the CNN using the preprocessed data and the hyperparameters selected by the HHFA algorithm. The trained HHFA-CNN was then tested using a separate dataset to evaluate its performance.
5. The developed technique (HHFA-CNN) was implemented on a MATLAB R2020a environment.
6. The performance of the developed HHFA-CNN model on the handwritten identification system was evaluated using Sensitivity, Specificity, False Positive Rate (FPR), Accuracy and Computational time.
7. T-paired sample test was used to measure the performance of the algorithm

The dataset was acquired from online sources (www.kaggle.com) including public datasets and from individuals at LAUTECH. The dataset contains over **5000** handwritten documents with correspondent images for real and forged signatures. Each image contains about 10 handwritten documents from the same user id. In order to expand the variety of the datasets obtained from the Kaggle repository. For handling handwritten document images and to improve their suitability for accurate character recognition, the intensity levels of the RGB picture are represented by "m-by-n-by-3 arrays of class unit 8, unit 16, single or double," and the RGB image is sometimes referred to as a true colour image. Single or double array values range from 0 to 1, whereas values for 8 units range from 0 to 255. A grayscale digital picture is one in which each pixel's value simply conveys information about intensity. The grey-scale picture is created using several grey hues. By maintaining the luminance, or brightness of the original picture, the true colour image is converted into a grayscale image. The firefly algorithm was used to get the optimal hyper parameter. while the higher-level strategy was used to adaptively adjust the firefly algorithms parameter using no of filter, filter size, batch size, convolutional layers as the hyper parameter to optimize the CNN. The recognition rate was estimated to be 99.3%. the graphical user interphase application was developed with Ladoke Akintola university staff and students handwritten and online samples, the toolboxes used image processing, computer vision and optimization in MATLAB 2020 Convolution half extracts high-level features from the input patterns. It majorly

consists of four essential layers: convolution layer, activation layer, batch normalization layer, and pooling layer. The convolution layer extract features from the input patterns, and it contains many filters with different sizes. The activation layer is a nonlinear layer that consists of varying activation functions used to avert linearity inside the system. Rectified Linear Unit (ReLU) is a commonly used activation function because it gives immediate commutation results and does not endure vanishing difficulties Training a CNN architecture is the most critical thing to define the exact weight for the convolution and classifier halves which cause a reduction in variance between the recognized output labels with actual labels of the dataset during training. Loss and the optimization functions play a significant role in the back-propagation algorithm for training the network. The loss function is the cost function. It is well defined as the difference between predicted x and actual x∗ label. During multi-class classification, the loss function is evaluated. (where S defines the number of categories. In the DL approach, various training optimization functions such as stochastic gradient descent moment, RmsProp, Adaptive Moment Estimation, Nesterov-accelerated Adaptive Moment Estimation, etc., were used. Adaptive movement was effectively used in the DL (Digital Learning) approaches to reduce the loss function. Optimal parameters used in the DL algorithm, like learning rate initialization and scheduled learning rate (SLR), SLR play a significant role in achieving a high recognition rate. Extraction of features from the patterns and tuning the hyper parameters are two techniques for the pre-trained network during transfer learning. During feature extraction, the convolution half network is frozen, and its output is used to train a new dataset. Simultaneously, in fine-tuning the hyper parameters method, early layers of the convolution half are firm, and the remaining layers are trained concerning the new classifier half. Dimension of the new dataset showed similarity to the original dataset used for training, the pre-trained CNN architecture are the two primary factors for choosing the appropriate transfer learning. Four different types of dataset schemes are as follows: Scheme one deals with the new dataset. The initial dataset was smaller in size, so tuning hyper parameters is not the best step to avoid theoverfitting problem in the architecture. Finally, the extraction of high-level features from the pattern is the best choice. Scheme two deals with the enormous number of new datasets considered; however, it is still fewer than the initial dataset; thus, feature extraction and tuning of hyper parameters are combined for better recognition of results.

The third scheme is about a new dataset with minimal quantity compared to the early dataset and having dissimilar content. Feature extraction is the best choice to deal with the above method for better performance of the pre-trained CNN. The last scheme deals with many new datasets with dissimilar content that need feature extraction, and tuning parameters are the best choice

In the algorithm above HHFA leverages the exploration-exploitation dynamics of the Firefly Algorithm, enhanced with heuristic operators, to automate the selection of hyperparameters such as the number of layers (L), number of filters (F), filter size (S), and batch size (B). The algorithm operates within predefined search bounds and optimizes an objective function based on the recognition rate $(R(\theta))$ of the CNN. By integrating heuristic techniques for refinement, HHFA achieves robust convergence to high-performance hyperparameter configurations, minimizing manual tuning efforts in CNN design.

The initialization phase of HHFA involves generating a population of N fireflies, where each firefly represents a unique CNN hyperparameter configuration (θ_i). These configurations are sampled randomly within their respective bounds, ensuring integer constraints for parameters like L, F, S, and B. The fitness of each firefly is evaluated by training and validating a CNN using its hyperparameters to compute the recognition rate ($R(\theta_i)$). This step ensures that initial solutions are diverse and span the search space effectively, providing a solid.

Algorithm: Hyper-Heuristic Firefly Algorithm (HHFA) for CNN Hyperparameter Optimization

INPUT:

(a) Optimization parameters:

Number of fireflies ($\boldsymbol{N}$), maximum iterations (MaxIter), absorption coefficient ($\boldsymbol{\gamma}$), attractiveness coefficient ($\boldsymbol{\beta_0}$), step size $(\boldsymbol{\alpha})$.

(b) Search bounds for CNN hyperparameters:

$\boldsymbol{L_{min}, L_{max}}$ (number of layers),

$\boldsymbol{F_{min}, F_{max}}$ (number of filters),

$\boldsymbol{S_{min}, S_{max}}$ (filter size),

$\boldsymbol{B_{min}, B_{max}}$ (batch size).

(c) Dataset: Input training and validation dataset for CNN evaluation.

(d) Objective function: Recognition rate $\boldsymbol{R(\theta)}$, where $\boldsymbol{\theta}$ represents the CNN hyperparameters.

Step 1: Initialization

(a) Initialize the positions of $\boldsymbol{N}$ fireflies, where each firefly $\boldsymbol{i}$ represents a CNN hyperparameter configuration:

$$\boldsymbol{\theta_i = \{L_i, F_i, S_i, B_i\},\ i = 1, 2, \ldots, N}$$

randomly sampled within the search bounds.

Where L: Number of layers in the CNN, F: Number of filters per layer, : Filter size (e.g., $\boldsymbol{3 \times 3}$) and $\boldsymbol{B}$: Batch size for training.

(b) Ensure the initialized hyperparameters are integers where applicable (e.g., , $\boldsymbol{F, S, B}$).

(c) Evaluate the objective function $\boldsymbol{R(\theta_i)}$ for all fireflies by training and validating a CNN with $\boldsymbol{\theta_i}$.

Step 2: Main Loop

For $\boldsymbol{t = 1}$ to MaxIter:

Step 2.1: Firefly Movement (Exploration)

(a) For each pair of fireflies i and j :

If $R(\theta_j) > R(\theta_i)$, move firefly i towards firefly j :

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \beta_{ij}\left(\theta_j^{(t)} - \theta_i^{(t)}\right) + \alpha \cdot \text{rand}$$

where:

$$\beta_{ij} = \beta_0 e^{-\gamma \left\| \theta_h^{(i)} - \theta_j^{(j)} \right\|^2}$$ (attractiveness function),

Where β_0 : Base attractiveness of fireflies, γ : Light absorption coefficient, controlling the attractiveness decay with distance, α : Randomization parameter for stochastic exploration. rand $\sim U(-1,1)$ (random perturbation).

Step 2.2: Heuristic Selection and Application (Exploitation)

a. Define a set of heuristic operators $H = \{h_1, h_2, \ldots, h_m\}$, where each heuristic modifies one or more hyperparameters (e.g., fine-tunina L, F, S, B).

b. Select a heuristic h_k using a strategy (e.g., random or performance-based).

c. Select a heuristic h_k using a strategy (e.g., random or performance-based).

d. Apply h_k to refine the updated hyperparameters:

$$\theta_i^{(t+1)} \leftarrow h_k\left(\theta_i^{(t+1)}\right)$$

e. Ensure updated hyperparameters remain within valid bounds and integers.

Step 2.3: Boundary Handling For each $\theta_i^{(t+1)}$:

$$\theta_i^{(t+1)} = \max\left(\min\left(\theta_i^{(t+1)}, \theta_{\max}\right), \theta_{\min}\right)$$

where $\theta_{\min}$ and $\theta_{\max}$ are the bounds for each hyperparameter.

Step 2.4: Evaluation and Update

a. Evaluate $R\left(\theta_i^{(t+1)}\right)$ by training and validating the CNN with $\theta_i^{(t+1)}$.

$R(\theta)$: Recognition rate of the CNN for hyperparameter configuration θ.

b. Update the best solution θ^* :

$$\theta^* = \arg\max_{\theta_i} R(\theta_i), \forall i = 1, \ldots, N$$

Step 3: Convergence

If the stopping criteria are met (e.g., $t =$ MaxIter or negligible improvement in (θ^*)), stop.

OUTPUT: Return the optimal CNN hyperparameters θ^* and the corresponding recognition rate $R(\theta^*)$.

Results and Discussions

Performance of the CNN technique

Model	Class	Thres hold	TP	FN	FP	TN	FPR (%)	SPEC (%)	SEN (%)	PREC (%)	ACC (%)	Time(sec)
CNN	ALL	0.24	2817	143	150	2810	5.067568	94.932432	95.16892	94.944388	95.050676	211.428176
CNN	ALL	0.36	2816	144	148	2812	5	95	95.13514	95.006748	95.067568	210.184459
CNN	ALL	0.5	2815	145	146	2814	4.932432	95.067568	95.10135	95.069233	95.084459	209.96572
CNN	ALL	0.8	2814	146	143	2817	4.831081	95.168919	95.06757	95.164018	95.118243	211.019358

Table 2: Performance of the HHFA-CNN technique

Model	Class	Thres hold	TP	FN	FP	TN	FPR (%)	SPEC (%)	SEN (%)	PREC (%)	ACC (%)	Time(sec)
HHFA-CNN	ALL	0.24	2864	96	104	2856	3.513514	96.486486	96.75676	96.495957	96.621622	166.478938
HHFA-CNN	ALL	0.36	2863	97	101	2859	3.412162	96.587838	96.72297	96.592443	96.655405	171.737974
HHFA-CNN	ALL	0.5	2862	98	98	2862	3.310811	96.689189	96.68919	96.689189	96.689189	182.355653
HHFA-CNN	ALL	0.8	2861	99	96	2864	3.243243	96.756757	96.65541	96.753466	96.706081	183.29289

From Tables 1 and 2, four significant thresholds were noted during the course of running the algorithms: 0.24, 0.36, 0.5, and 0.8. From Table 1 above, it was observed that the CNN algorithm has true negative and true positive values of 2810 and 2817, respectively, at the 0.24 threshold. True positive and true negative are values that measure the accuracy of the forged and original datasets, respectively. At 0.36 threshold, the true positive value was 2816, while the true negative value was 2812. We observed high figures for both the forged and the original data sets. The same applies to the thresholds of 0.5 and 0.8. Also, the false positive rate, which is supposed to measure how the model misclassified both forged and the original data set, was low; this showed that the model was reliable. The sensitivity value at 0.24 threshold was very high, 95.17(%), accuracy was 95.05(%), while the time spent was 211 seconds; these results showed that the model was able to give an accurate prediction and hence reliable.

Table 2 shows the performance of the HHFA-CNN technique. At a threshold of 0.24, the true positive and true negative values were 2864 and 2856, respectively. The false positive rate was 3.514%. The values for specificity, sensitivity, Accuracy, and specificity were 96.49%, 96.76%, 96.62%, and 96.50%, respectively. The result showed that the HHFA-CNN model was reliable. At a threshold of 0.8, the false positive rate was 3.24%, indicating that the level of wrong classification was very low. The accuracy and sensitivity values were 96.71% and 96.66%, respectively, indicating that the model was reliable. It was observed that the HHFA-CNN algorithm yielded better results than the CNN algorithm.

Table 3: Threshold 0.24

Model	Class	Thres hold	TP	FN	FP	TN	FPR (%)	SPEC (%)	SEN (%)	PREC (%)	ACC (%)	Time(sec)
CNN	ALL	0.24	2817	143	150	2810	5.067568	94.932432	95.16892	94.944388	95.050676	211.428176
HHFA-CNN	ALL	0.24	2864	96	104	2856	3.513514	96.486486	96.75676	96.495957	96.621622	166.478938

Table 4: Threshold 0.36

Model	Class	Thres hold	TP	FN	FP	TN	FPR (%)	SPEC (%)	SEN (%)	PREC (%)	ACC (%)	Time(sec)
CNN	ALL	0.36	2816	144	148	2812	5	95	95.13514	95.006748	95.067568	210.184459
HHFA-CNN	ALL	0.36	2863	97	101	2859	3.4121, t62	96.587838	96.72297	96.592443	96.655405	171.737974

Table 5: Threshold 0.5

Model	Class	Thres hold	TP	FN	FP	TN	FPR (%)	SPEC (%)	SEN (%)	PREC (%)	ACC (%)	Time(sec)
CNN	ALL	0.5	2815	145	146	2814	4.932432	95.067568	95.10135	95.069233	95.084459	209.96572
HHFA-CNN	ALL	0.5	2862	98	98	2862	3.310811	96.689189	96.68919	96.689189	96.689189	182.355653

Table 6: Threshold 0.8

Model	Class	Thres hold	TP	FN	FP	TN	FPR (%)	SPEC (%)	SEN (%)	PREC (%)	ACC (%)	Time(sec)
CNN	ALL	0.8	2814	146	143	2817	4.831081	95.168919	95.06757	95.164018	95.118243	211.019358
HHFA-CNN	ALL	0.8	2861	99	96	2864	3.243243	96.756757	96.65541	96.753466	96.706081	183.29289

Tables 3, 4, 5, and 6 showed the comparison of HHFA-CNN with the existing CNN algorithm. It was noted from Table 3 above that at the threshold of 0.24, the true positive values of both CNN and HHFA-CNN algorithms were 2817 and 2864, respectively, while the true negative values were 2810 and 2856, respectively. This implied that the model was able to make predictions for both forged and original data sets. It means the model is reliable. Considering all the figures for the two algorithms at the threshold of 0.24, it was observed that the figures for specificity, sensitivity, precision, and accuracy showed a significant improvement between CNN and HHFA-CNN. HHFACNN performed better than the CNN model. The false positive rate, which indicates how the model misclassifies the original and forged handwritten samples, was lower in HHFA-CNN than in CNN, approximately 3.514 and 5.068, respectively. Likewise, the time spent running the model is considerably lower in HHFA-CNN (166.5 s) than in the CNN (211.4 s) algorithm. It can therefore be concluded that the HHFA-CNN algorithm performed better than the existing CNN algorithm. Observing the models across the different thresholds, as shown in tables 4, 5, and 6, revealed improvements in specificity, sensitivity, precision, and accuracy. Sensitivity was used to measure the actual positive datasets (forged datasets) that the model correctly identified. High sensitivity indicated that the model was good at identifying forged datasets. Accuracy measures the proportion of correct predictions of the forged and the original datasets out of all predictions in the model. High accuracy, as observed in all tables 3,4,5, and 6, indicated that the model is making large proportions of correct predictions.

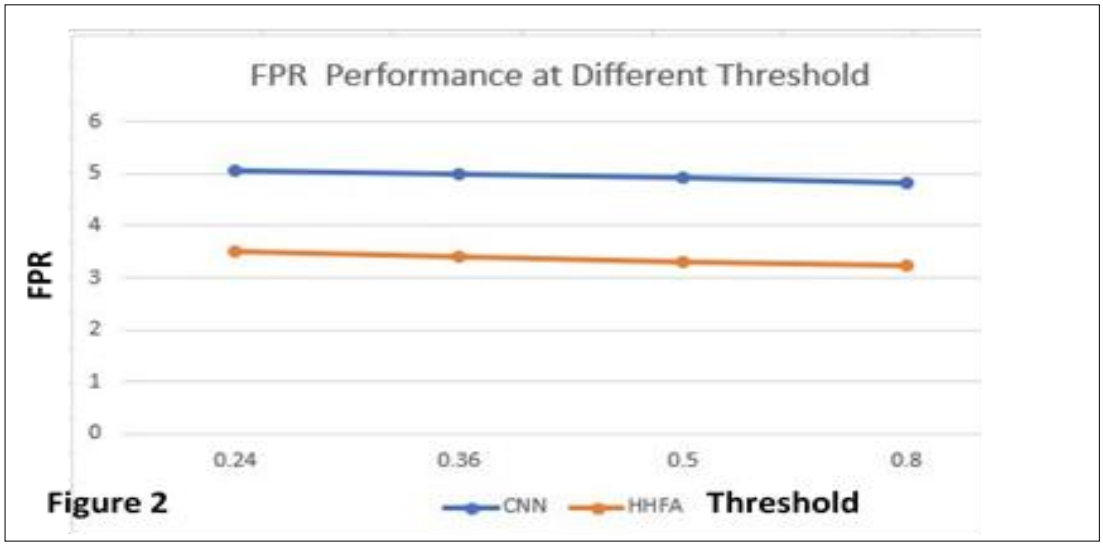

Figure 2

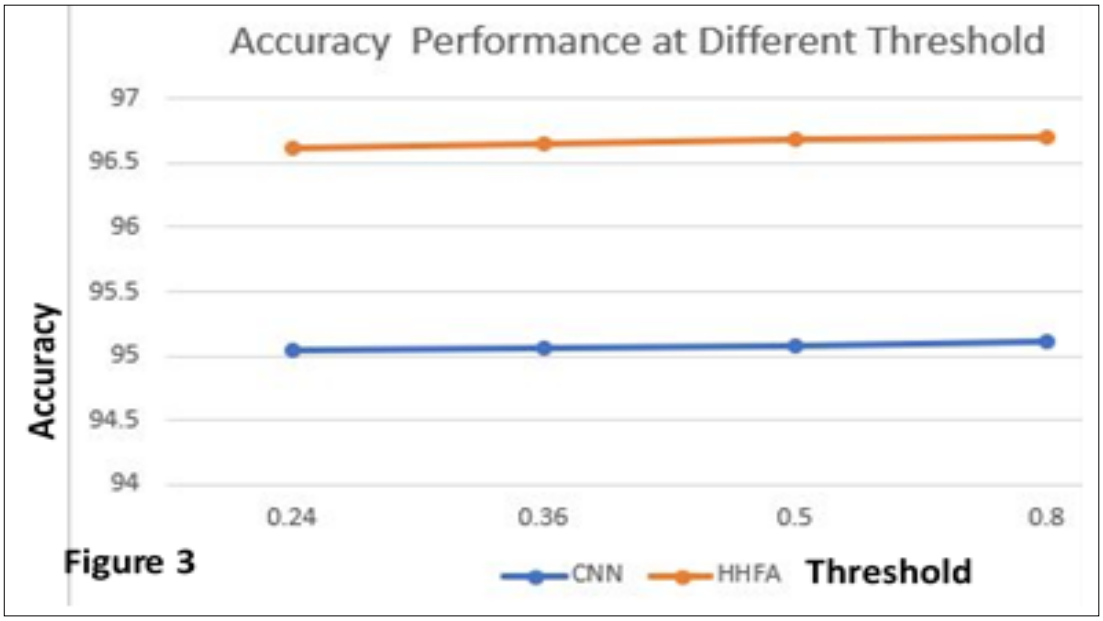

Figure 3

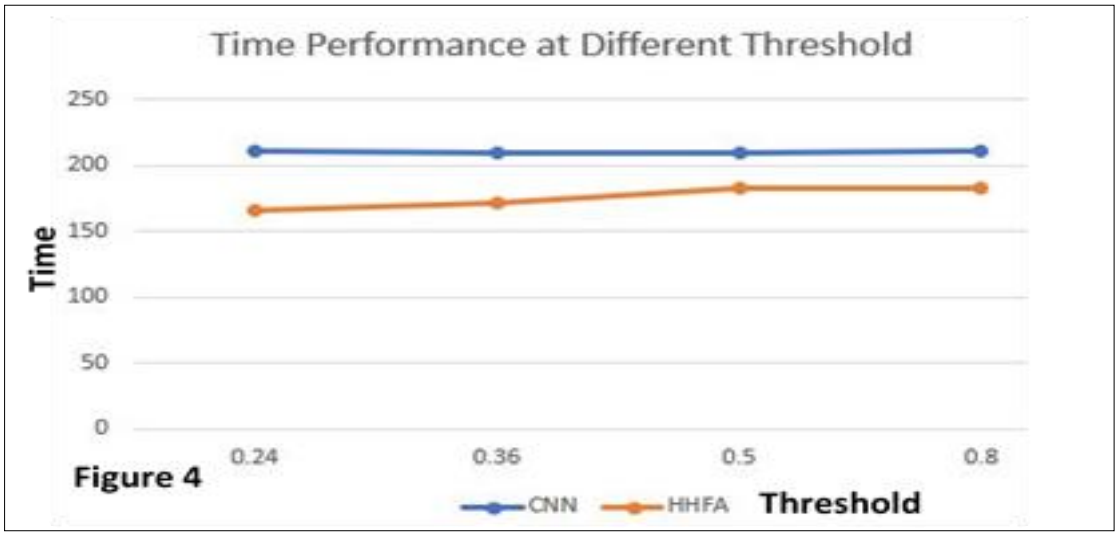

Figure 4

Figures 2, 3, and 4 show the plots of the threshold against the false positive rate, accuracy, and time. It can be observed that the HHFA-CNN shows a significantly better performance than the CNN algorithm. The false positive rate showed a lower value in HHFA-CNN than the CNN model, and the time spent running HHFA-CNN was lower than the time spent in the CNN algorithm.

Table 7: Summary of the t-test result of HHFA-CNN and CNN result

Parameter	t	Degree of Freedom (df)	p-value	Comment
FPR	-124.009	3	0.000	Significant
Sensitivity	1221.455	3	0.000	Significant
Precision	121.181	3	0.000	Significant
Accuracy	203.942	3	0.000	Significant
Computation Time	-8.139	3	0.004	Significant

Hypothesis was postulated that

H0: there is no significant difference between HHFA-CNN and CNN

H1: there is a significant difference between HHFA-CNN and CNN

From the above, it could be observed that the t value calculated was less than the t value from the table, hence the H0 was rejected while H1 was accepted, indicating that there is a significant difference between HHFA-CNN and the CNN algorithm

Conclusion

The results obtained in this research showed that the HHFA-CNN had an improved recognition rate compared to the CNN algorithm in all instances of the original and forged datasets, using different thresholds. It provided evidence of the importance of applying optimisation algorithms to find optimal parameters of a convolutional neural network architecture.

References

1. Khandokar I, Hasan M, Ernawan F, Islam S, Kabir MN. Handwritten character recognition using convolutional

neural network. Journal of Physics: Conference Series,2021:1918(4):042152.
2. Sahloul A, Suen C. Off-line system for the recognition of handwritten Arabic character. Fourth international conference on computer science & information technology, 2014, 227–244.
3. Almansari OA, Hashim WN. Recognition of isolated handwritten Arabic characters. 7th international conference on mechatronics engineering, ICOM, 2019, 1–5.
4. Mustapha IB, Hasan S, Nabus H, Shamsuddin SM. Conditional deep convolutional generative adversarial networks for isolated handwritten Arabic character generation. Arabian Journal for Science and Engineering,2022:47:1309–132.
5. Al-Helali BM, Mahmoud SA. Arabic online handwritten recognition AOHR. A survey. ACM Computing Surveys,2017:50:1–35.
6. Hossain MT, Hasan MW, Das AK. Bangla handwritten word recognition system using convolutional neural network. 15th International Conference on Ubiquitous Information Management and Communication IMCOM, 2021, 1–8.
7. Khosravi S, Chalechale A. Chimp optimization algorithm to optimise a convolutional neural network for recognising Persian/Arabic handwritten words. Mathematical Problems in Engineering, 2022.
8. Lincy RB, Gayathri R. Optimally configured convolutional neural network for Tamil handwritten character recognition by improved lion optimisation model. Multimedia Tools and Applications,2021:80:5917–5943.
9. Aswanadini R, Deep C. Hyper heuristic firefly algorithm based convolutional neural networks for big data cyber security. Indian Journal of Science and Technology,2021:14:2934–2945.
10. Khandokar I, Hasan M, Ernawan F, Islam S, Kabir MN. Handwritten character recognition using convolutional neural network. Journal of Physics Conference Series,2021:1918(4):042152.
11. Saqib N, Haque KF, Yanambaka VP, Abdelgawad A. Convolutional-neural-network-based handwritten character recognition: An approach with massive multisource data. Algorithms,2022:15(4):129.
12. Johnson J, Karpathy A, Fei-Fei L. Densecap Fully convolutional localisation networks for dense captioning. IEEE Conference on Computer Vision and Pattern Recognition, 2015, 4565–4574.
13. Ahlawat S, Choudhary A, Nayyar A, Singh S, Yoon B. Improved handwritten digit recognition using convolutional neural networks CNN. Sensors,2020:20(12):3344.
14. Le H, Borji A. What are the receptive, effective receptive and projective fields of neurons in convolutional neural networks arXiv, 2018.
15. Eickenberg M, Gramfort A, Varoquaux G, Thirion B. Seeing it all Convolutional network layers map the function of the human visual system. Neuroimage,2017:152:184–194.