International Journal of Circuit, Computing and Networking

**E-ISSN:** 2707-5931
**P-ISSN:** 2707-5923
www.computersciencejournals.com/ijccn
IJCCN 2024; 5(1): 01-11
Received: 02-11-2023
Accepted: 07-12-2023

**Ahmed Al-Ghanimi**
Department of Computer Science, College of Pharmacy, University of Babylon, Babylon, Iraq

**Corresponding Author:**
**Ahmed Al-Ghanimi**
Department of Computer Science, College of Pharmacy, University of Babylon, Babylon, Iraq

# Optimizing English character recognition: Enhanced accuracy through meta-heuristic algorithms

## Ahmed Al-Ghanimi

**DOI:** https://doi.org/10.33545/27075923.2024.v5.i1a.60

**Abstract**
Recognition of symbols and words is important today, and neural networks can be used to detect these symbols and words. The main problem addressed is the requirement for a dependable and effective system that can achieve high accuracy despite the presence of varied font styles and minimal training data. Through the results that we present in this research, we conclude that the results obtained are better than the results reached in previous research methods by combining the Grasshopper Optimization algorithm and the Propeller algorithm. Specifically, compared with existing methods, our method improves the accuracy by about 1.11% to 6.35% for uppercase letters and 3.3% to 7.5% for lowercase letters. We can say that the Grasshopper algorithm helps in identifying the initial ideal areas Also, through an algorithm Propeller, we can improve these areas optimizes, Through this, we can determine the maximum possible degree of accuracy in the neural network. We can say that mixing these systems leads to a more powerful and effective system for character recognition, especially with differences in fonts. Through the results that we were able to obtain, we notice many features and characteristics that can help us solve the problem at hand: Firstly, very high accuracy in recognizing uppercase and lowercase letters in relation to the English language and comparing them with currently available methods. Secondly, a great ability to adapt to various font styles and also the ability to maintain high accuracy with the data specified for training. This proposed method can also recognize the handwriting of small and large letters in the English language, as these results showed great accuracy and efficiency in character recognition. Through this, it can be more reliable in real-world scenarios with different fonts and available data.

**Keywords:** English Alphabet Recognition, Grasshopper Optimization Algorithm, Character Recognition, Pattern Recognition, Optical Character Recognition (OCR)

## Introduction
Recently, it has become necessary to find a strong system for character recognition in light of the digital transformation. Where documents and texts appear in a variety of fonts and styles, This makes it necessary to create systems to recognize them and also. Through this, it can be said that obtaining large sets of training data can be a tremendous task, especially when dealing with non-traditional languages and fonts. In this research, we seek to overcome these obstacles by ensuring high accuracy in character recognition, even with scenarios that contain. Big data for training and different styles [1].
Character recognition is considered a part of one of the basic components of optical character recognition systems, It has a pivotal role in applications ranging from document processing to text extraction in digital environments Therefore, accurately recognizing English letters, both upper and lowercase letters, is a necessary challenge, especially when dealing with non-specific training systems and data. Therefore, through this study, we are developing a system that can recognize letters with high accuracy in order to address these challenges through the collaboration of the neural network and meta-optimization algorithms.
In order to achieve our goal in this article, we use a multi-layer feed-forward neural network, which is a well-established architecture for pattern recognition in its tasks from this, it can be said that success in such networks depends on the precise arrangement of information. The back propagation learning algorithm serves as the main tool for building and training a neural network and can be sensitive to initial information values. In order to increase the accuracy and reduce this sensitivity in character recognition, we are merging the Grasshopper and Propeller algorithms so that we can improve the description.
We test the character recognition system that we developed in this research using a data set

consisting of 26 English letters, whether uppercase or lowercase, and various font styles. After completing the system's work, we compare the results obtained through the followed method and the results obtained from previous methods.

Through this work, we were able to focus on common uses of both the Propeller and Grasshopper algorithms in enabling highly accurate character recognition. Also, through the strength of the system that was built in obtaining high accuracy in recognizing uppercase and lowercase letters with different styles of different fonts, it is a very good contribution to the field of letter recognition.

Through FCN methods, we note that it has a great ability to process images of even random sizes and enables the classification of images according to the required distributions. Given that handwritten word blocks can have varying lengths, FCN methods can offer advantages in such cases. Building on this line of research, the authors of the study [4] have demonstrated that fully convolutional approaches can outperform iterative networks in sequence modelling problems.

During previous periods, character recognition through scanned images was a very complex matter and received great attention through artificial intelligence processing, image processing, and pattern recognition. The appearance of a visual character lacks fixed and universally applicable rules. As a result, it becomes necessary to extract and learn rules from sample data to accurately recognize and classify visual characters [5].

Optical character recognition (OCR) technology was developed to enable machine-readable conversion of printed text into text code. It finds applications in various domains, including aiding visually impaired individuals, interpreting incoming telegram messages, and converting characters into telegram messages. In the 1950s, OCR gained popularity for data entry tasks. Additionally, OCR plays a crucial role in language translation by facilitating the conversion of foreign text into another language. Scanned or photographed handwritten and printed documents cannot be directly utilized as images. Converting such data from image format to a computer-readable format can be expensive for organizations managing large volumes of data. Furthermore, the documents that arise from this procedure could include human mistakes. It is advised to use computer-assisted conversion of such papers to get around these issues and increase efficiency. Due to changes in data formats, traditional methods like optical mark recognition (OMR) sheets and form data have become obsolete [6].

Through this presented research, we divide the work as follows, where the introduction part contains a simple explanation of the topic and its importance, In the second part, we studied some previous research and the problems researchers faced in their research, In the third part, we defined the objectives of our study, In the fourth part, we explained in detail our method of work, and In the last part, we reviewed the results obtained in this study. The goal of the study is to develop a character recognition system, where the Grasshopper algorithm helps identify initial optimal regions, while the Propeller algorithm refines these regions, precisely determining the neural network parameters to achieve maximum accuracy. This combination results in a more robust and efficient character recognition system, especially when dealing with font variations. Now, the main idea of the research is to improve English letter recognition.

**Literature review and problem statement**

Character recognition involves the task of segmenting and identifying characters within an image. The objective of character recognition is to detect and convert characters from the input image into machine-encoded text, such as ASCII. This process increases automation and makes it easier for humans to use computers in various applications. Personality recognition, while a branch of pattern recognition and artificial intelligence in a broader sense, is an interesting and significant subfield. Character recognition has received attention in recent years and many techniques have been investigated such as decision-tree induction, neural networks, and probabilistic and Jacobian distance. This attention is motivated by the large number of applications that character recognition has [7].

We will review below the steps followed to identify character: First, we do pre-processing in order to improve the input document by eliminating noise and increasing the shape information to the maximum extent. This leads to obtaining a good image with optimal compression. In Section II, we will break down segmentation because it plays a crucial role in influencing text recognition accuracy. It can be said that achieving separation between lines, words, and letters is important for successfully defining a character. The third part of the work, that is, after decorating, extracts the various characteristics and features such as width, height, vertical and horizontal lines, and also recognition of upward and downward strokes from each individual letter. In the fourth part of the work, we detect and classify the clips using the back propagation algorithm, which helps in classification. Final step: The recognized characters are stored in a text format, representing the output of the character recognition process.

OCR research has a rich history that dates back to a time before the advent of early computers. Numerous efforts have been made to enhance license plate OCR systems, but challenges persist due to the dynamic nature of the input set. Traditional OCR systems are typically trained to recognize specific handwriting styles, making them less effective when faced with dynamic variations. Handling texts with multiple fonts, styles, and spacing further compounds the challenge [8].

The paper [9] proposed a combination of RNN-HMM for offline English handwriting detection. They utilized HMM to obtain framework-based labeling and used these labeled frames as input for an RNN, which generated posterior probabilities for HMM propagation and transcript generation. They also introduced a technique that involved scaling LSTM memory cell gates using scalar multipliers in each RNN layer.

In the [10] study, the authors suggested employing the Tesseract OCR engine, an open-source OCR engine, to train a Tamil OCR model. The proposed method involved preparing an educational dataset using computer images of Tamil characters and creating an OCR alphabet for the Tamil script. The training process included the preparation of educational images and character segmentation using a box file creation process. The module was trained on various images of different sizes and font types.

Also, the paper [11], the authors presented an automatic oblique plate detection method. The input image was converted to a binary image, and noise removal was performed as a crucial preprocessing step. Four algorithms were developed for generating encoded text of the section

characters. These algorithms involved vector intersection, zoning, correlation assessment, and a combination of intersection and zoning. MATLAB was utilized for implementing these methods.

The [12] paper introduced a fuzzy logic-based method for license plate identification using scale-invariant feature transform (SIFT). SIFT enabled the identification, extraction, and description of local image features that remained invariant to scaling, rotation, and brightness changes. The method aimed to minimize disruptions caused by noise, occlusions, and clutter during feature comparison.

The authors in [13] proposed a method for Urdu character recognition using point attribute matching. The input image underwent preprocessing and was compared with object patterns stored in a database. These patterns consisted of individual Urdu characters and lines of characters forming Urdu words. Feature points were identified in the target images, and corresponding attributes were extracted to locate the object within the image. The method achieved a 92% detection rate.

The study [14] presented a feature selection method for handwritten character recognition that is ranking-based. This strategy sought to enhance classification performance while lowering the computational cost of classification. It addressed problems with classifier reliance, computational complexity, and analysing feature relationships. The strategy produced feature rankings by combining several univariate measurements with feature-based strategies. A selection of characteristics that would optimize classification results was chosen using greedy search techniques, which significantly reduced complexity with just a little drop in detection rate.

The paper [15] developed a method for character recognition in diverse datasets, focusing on handwritten text and layout characters. The technique used the information energy methodology for line segmentation and the radial part encryption algorithm to find the necessary text in photos. It was created to handle printed and handwritten papers that were written in Arabic, English, and French. The method involves preprocessing to remove noise from closely spaced components, followed by the classification of text and non-text using a cc-based learning strategy and MLP classifier. Following layer separation, block division was used to separate typed from handwritten material using the code book technique.

The authors of [16] study introduced an adaptive transfer classification method for handwritten Chinese character recognition. They proposed a novel neural network structure for HCCR, where large amounts of source domain data were labeled and a small amount of target domain data was used to learn model parameters. The approach incorporated classification-level transfer and assigned different weights based on the usefulness of the source domain data for the classifiers. Experimental results on three datasets demonstrated the effectiveness of the proposed model compared to advanced baselines.

[17] Researchers proposed character recognition using highly complex neural networks. The preprocessing step normalized the input blocks to a standard display, eliminating the need for costly duplicate symbol correction. Character classification was performed implicitly without relying on predefined dictionaries or background information. The focus was on extracting handwriting symbol blocks, and the input was represented as a well-reduced image of a custom one-dimensional symbol sequence.

We note in Reference [18] that he presented an approach to recognize numbers, letters, handwritten English through a neural network. We note that feature extraction techniques and pre-processing techniques were used due to the difference in shape, slope, and size of the individual character. We note that The Feed Forward algorithm was able to provide insights into the performance of the neural network, and then the Back-Propagation algorithm for experimentation and training as well. Geometric features and gradual extraction techniques were used to recognize English numbers and letters using the neural network.

We note in Review [19] that he presented his method of the offline recognition of Gurmukhi characters Through the use of a workbook the k-NN. We note that in this paper he analyzed the effect of combining feature classification and feature extraction techniques using the SVM classifier. It is noteworthy that he used (PCA) to be able to determine the effective characteristics of the document, and he also used steps such as pre-processing, digitization, feature extraction for classification or classification, the use of SVM, and the use of K-NN to achieve high accuracy for this research.

In his paper [20] he presented a method for recognizing handwritten characters in Javanese also based on an artificial neural network. The process involved feed forwarding to obtain the error and subsequently adjust the weights accordingly.

The paper [21] proposed a method for recognizing Tamil handwriting characters based on the closest point of interest. The study focused on image-to-image adaptation through feature analysis, without relying on machine learning approaches. The approach involved making local decisions about the class of the experimental image based on individual characteristics called IP training images. These local decisions were then processed to make an overall decision about the class of the experimental image.

The researchers of study [22] introduced multi-objective optimization for detecting isolated handwritten Indian scripts. They developed an efficient area sampling technique to identify the most informative local areas. The local areas were ranked based on their contribution to detection accuracy in the cross-validation dataset. These rankings guided the proposed algorithm, and the importance of a local area was determined by calculating the negative SVM classifier detection accuracy on the cross-validation dataset while considering all other features except the given one.

The study [23] proposed Bangla handwriting character recognition using a convolutional neural network (CNN) with data augmentation. The model achieved an accuracy of 91.81% on the basic dataset of letters (50-character classes) and further improved performance by increasing the number of images to 200,000 through augmentation. The model was designed for easy testing and collaboration on a web server, utilizing the Google Colab cloud-based web interface for AI experiments.

The paper [24] showed recognition of fixed-size handwritten characters using single-layer feed-forward neural networks. Their proposed system focused on offline recognition of numbers and the Latin alphabet. The query image characters were dynamically resized to 60x40 pixels and sent to the neural networks for identification. The resizing process maintained the character's dimension ratio to preserve the image integrity. The proposed system successfully separated

handwritten text characters from the query image, achieving an accuracy of 95.69%.

Researchers of study [25] addressed the recognition of handwritten Hindi characters using deep learning techniques, specifically convolutional neural networks (CNN) with the RMSprop optimizer and deep neural networks (DFFNN). The framework was trained on a large database of image samples and tested on experimental customer characteristic information index images, yielding highly validated results.

The paper [26] introduced a method for recognizing handwritten Chinese characters based on conceptual learning. They utilized the idea of human learning and developed a calculated model learning approach that could achieve detection with small sample sizes, distinguishing it from existing deep learning models. In order to generate a useful model for stroke-related Chinese characters, the method suggested building a "meta-stroke" combining character impact extraction and techniques library using prior knowledge and Bayesian program learning.

This study examined a number of structural and statistical elements and, on the basis of those qualities, suggested a novel and distinctive viewpoint. Using the Multilayers Perceptron (MLP) and Support Vector Machines (SVM) classifiers, the efficacy of these characteristics has been evaluated in the identification of English handwritten characters. The updated display-based functionality has proven to be quite effective. Results of extensive assessment tests conducted with various attributes or combinations of attributes and categories are displayed. The results of the experiments demonstrate how accurate the detected traits are. This is done by integrating many aspects using artificial intelligence, and through this work we use the MLP neural network to address the problems of character classification. It can be said that the idea of using Grasshoppers it is a tool used for improvement in a mixed architecture to improve performance in the MLP neural network.

## Materials and Methods

### 1. Datasets

The data used in our research was obtained from website the Kaggle [29]. The data used in the research consists of a set of 26 folders containing uppercase and lowercase letters. Each folder contains 3232 letters and also includes more than 14,900 errors, as the data set targeted in the work was diverse as it contains both lowercase and uppercase letters. This is all in the interest of the work so that more results can be obtained. Accuracy and strength for character recognition

### 2. Preprocessing

We improved the quality of the input data before training the network by subjecting it to a series of processing:

1. We started by converting color images to grayscale images so that we could simplify the data and preserve its basic information.
2. We converted grayscale images to 2D images, focusing on them in terms of the shape of the letters by applying thresholding.
3. We went through the process of scaling all images to a fixed size of 15 x 15 pixels, ensuring reduced computational complexity and uniformity for all images.
4. We normalized the values of each pixel to a scale of [1,0], and this led to stability in training and improved convergence in the image
5. We used some blur applications to remove slight differences within the images, which ensures that the image features are better captured.
6. We also subjected the images to flipping and rotation to increase the data set, and all of this goes towards enhancing the strength of the network that was built.

### 3. Optimization Algorithms

Through the work that we have done, we have used two algorithms in order to obtain the best improvement from the meta-heuristic optimization algorithms to adjust the information of the neural network:

Grasshopper An algorithm that mimics the behavior of a grasshopper so that we can explore solutions available in the image and achieve a balance between exploitation and exploration. This helps in identifying ideal areas in the solution space.

Propeller Algorithm This algorithm is inspired by nature and can be used to improve the design of the neural network, thus ensuring high accuracy of the work.

### 4. Neural Network Configuration

The general structure of the neural network consists of three layers. The first layer used contains 225 neurons, that is, 15 by 15 pixels. At the same time, it also contains hidden layers that contain more than 80 neurons. It also contains 52 cells from the output layers. This all represents lowercase letters. And the big one. This includes activation 'tansig' functions for the hidden layers and also 'trainbfg' for the output layer. We used back propagation training algorithms for more than 1000 iterations at a rate of 0.6.

### 5. Experimental Setup

In our work, we tested 80 neurons from hidden layers based on experience and the balance between learnability and complexity.

1. The current rate of convergence that was stable during the back propagation training process was 0.6.
2. Through our algorithm, the grasshoppers algorithm, we were able to explore diverse and good areas in the images.
3. Through our own work, we have conducted approximately 150 iterations to improve the network information for the MLP using our algorithm.
4. We have run our algorithm independently ten times to ensure good results.

### Performance Evaluation

We used the following equation to evaluate the accuracy of English letter recognition (1).

$$\text{accuracy} = \frac{(\text{TP}+\text{TN})}{\text{All samples}} \quad (1)$$

TP: represents predicted values equal to the original value

TN: represents negatively predicted values equal to the original value.

### Comparison with Existing Methods The aim

Through our own work, we compared the method that was built with the currently available methods [29, 30], and SVM

[28], and through this comparison, we demonstrated the effectiveness of our used system based on recognizing uppercase and lowercase English letters.
Through this presented approach that integrates a large set of diverse data, processing techniques as well as optimization algorithms, we can say that it has provided a strong framework for an accurate and effective character recognition system. Through an experimental design, it is built and a rigorous evaluation is conducted to highlight the possibility of the proposed method for application at the present time.

**Details of Proposed Method**
It can be said that the word meta-heuristic refers to a more in-depth degree of investigation that gives a wide range of positives to improvement issues. From previous literature, it can be said that the meta-heuristic algorithm is effective in solving many complex problems
It can be said that the attractiveness of using such algorithms lies in their ability to provide more suitable solutions to problems of large sizes and dimensions in a short period of time. It is noted that the optimization problems that received attention in the field of meta-heuristics had different characteristics
Nature-inspired algorithms divide the search process into two key phases: exploration and exploitation. During exploration, search agents are encouraged to explore the solution space extensively, while during exploitation, they focus on exploiting local regions. Optimization, as a process, involves finding the best values for problem variables to either minimize or maximize an objective function. First, the problem parameters need to be identified, followed by recognizing any constraints imposed on these parameters, which categorizes optimization problems as constrained or unconstrained. Fig. 1, 2 shows the life cycle of the Grasshoppers.

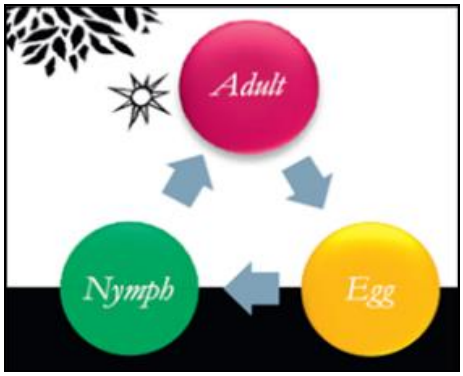


**Fig 1:** Show the life cycle of the Grasshoppers

**Fig 2:** A real Grasshopper

Repulsion occurs at a distance of [0, 2.079] When the Grasshopper is 2.079 units away from the other Grasshopper, there is no attraction and no repulsion of this area is called the comfort zone or comfort distance. Fig. 3 shows a conceptual model of gravity and repulsion and comfort.

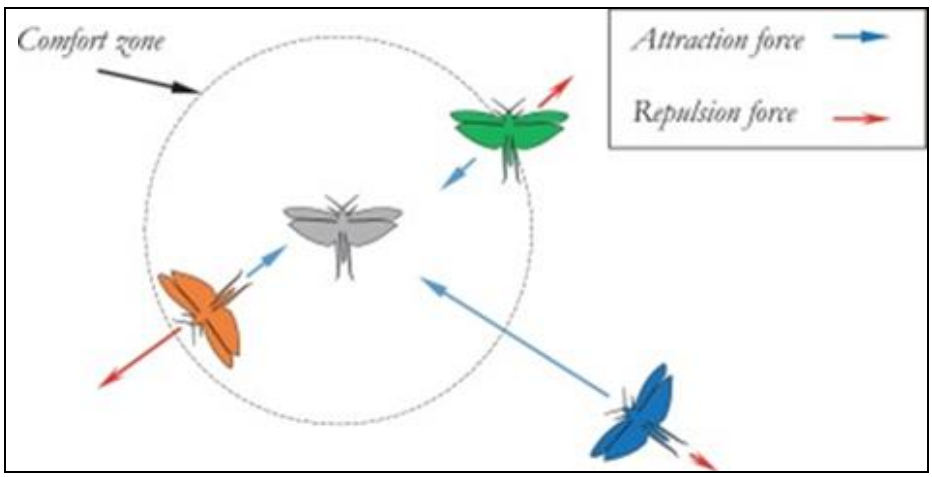


**Fig 3:** Conceptual model of areas of attraction, repulsion and comfort

Character or letter recognition refers to the process of assigning symbolic meaning to objects such as letters, symbols, and numbers depicted in an image. This recognition can be performed either online or offline. One significant challenge faced by existing recognition systems is the requirement for a large number of training samples, leading to extensive computational tasks. Each character necessitates multiple training samples, contributing to the computational load. In this study, our objective is to develop an automatic character recognition system for the English language that achieves high accuracy in recognition while minimizing training and classification time.
To address this, we propose utilizing the Grasshopper optimization algorithm to enhance the weights of a multilayer perceptron neural network during the training and learning process for character recognition. Since typographic characters are written in various fonts and sizes, similar characters may exhibit distinct writing styles.

The research methodology involves identifying the problem's parameters, which, in this case, include the weights and biases for optimization. These parameters are optimized using the Grasshopper algorithm. In the first stage of the work, we create a set of weights and biases randomly using algorithms. The optimal weight and bias outputs are the first values to train the network. Then we repeat the process until we are able to reach the optimal or best solution, and we will notice that the system's work goes through the following steps:

**Step 1:** During this stage, the Grasshoppers are dispersed across the problem space, with each Grasshopper representing a random position corresponding to the weights and biases.

**Step 2:** At this stage, we are in the process of entering data into the network, and it must be noted that the data has been previously modified before entering it into the network.

**Step 3:** At this stage, we review the images for the English letters, and if the image is obtained in color, we will convert it to grayscale images. The characters can be represented as binary values (0 and 1) or as the desired character from the outset. To enhance accuracy, a frame is added to ensure precise execution. The images are then resized to 15 by 15 pixels, significantly reducing training time by minimizing the number of nodes and weights that require updating.

**Step 4:** In this phase, random values between zero and one are selected as coefficients for r1, r2, and r3. These coefficients are utilized in the equations for social interaction (Si), gravitational force (Gi), and horizontal force (Ai) within (2):

$$X_i = r_1 S_i + r_2 G_i + r_3 A_i \quad (2)$$

Where Xi represents the position of the Grasshopper, Si represents social interaction, Gi represents gravitational force, and Ai represents horizontal force.

**Step 5:** The initial values for the gravity intensity (l) and gravity length (f) parameters are selected in this step.

**Step 6:** The values of cmin and cmax, set to 0.0004 and 1 respectively, are considered for the calculation of parameter C. Parameter C represents the balance between exploration and exploitation in the Grasshopper algorithm. Equation (3) calculates parameter C:

$$C = (c_{\max} - 1)\frac{(c_{\max} - c_{\min})}{L} \quad (3)$$

**Step 7:** In this step, the objective function, which represents the network error value, is computed for each Grasshopper in the population. The Grasshopper that exhibits the lowest objective function value is recorded as the best case for each variable.

**Step 8:** This step checks if the weight and bias values are sufficiently optimized. If they are, the algorithm proceeds to Step 10; otherwise, it goes back to Step 5.

**Step 9:** At this stage, the Grasshopper algorithm provides the optimal values for weights and biases, which are then utilized as the initial optimal values in the MLP neural network.

**Step 10:** It was defined the structure of the MLP neural network and select the Back propagation (BP) algorithm to train it. The training dataset consists of the 26 English alphabet characters displayed in five different fonts (Arial, Georgia, Times New Roman, Verdana, and Courier), all with a font size of 12 pt. Each character is represented by a 15x15 pixel box more details about datasets. This means that we encode each character as a vector of length 225 (15*15). The vector is constructed by arranging the pixels from left to right and bottom to top, where white cells are denoted as 0 and black cells as 1. The neural network comprises three layers: an input layer with 225 neurons, a hidden layer with 70 neurons, and an output layer with 52 neurons, accounting for uppercase and lowercase letters. Examples of characters from the database can be seen in Fig. 4, 5. We utilize the "tansig" activation function for the hidden layer and the "trainbfg" function for the output layer. The network is trained for 1000 iterations using a learning rate of 0.8.

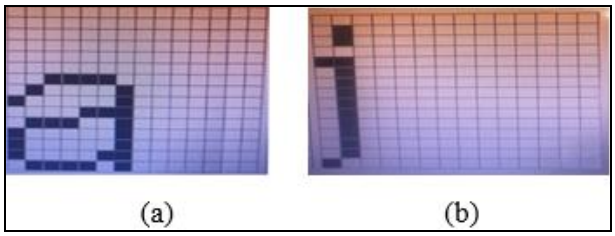


**Fig 4:** The main signature: a – present a letter with arial font while; b – shows j letter with font Georgia

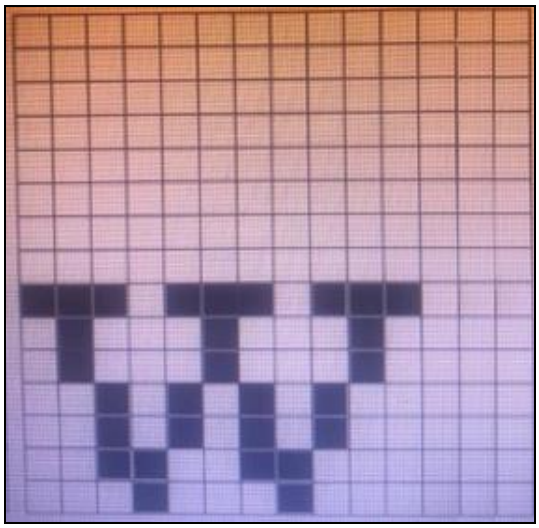

**Fig 5:** W character with Times New Roman font

**Step 11:** In this step, the training error using the Back propagation (BP) algorithm is calculated.

**Step 12:** At this stage, the weights and biases of the network are updated based on the calculated training error.

**Step 13:** During training, you will repeat the process until the difference between the average of the existing line that we obtain from two successive bags reaches a very small

value. If we do not obtain this small value, we will return to the algorithm to step No. 11.

**Step 14:** In this step, during the implementation phase, the converted binary images are compared with the data that was trained so that we can evaluate the network's performance.

**Choosing Impact the Optimization Process (Experiments)**

It is expected that the parameters specified in the neural network will greatly affect the improvement processes on the method that we have presented or proposed. We will mention the reasons behind this and what effects it will have on the improvement process:

**Learning Rate**

The font size determines the learning rate at which the optimization algorithm updates the weights during the training period. Through this, a higher education rate may lead to an acceleration in convergence, but instability and may cause overshooting. Also, a lower education rate can lead to slower convergence, but there may be an easier path to improvement.

**Impact on Optimization Process**

- If the learning rate is high, this will lead to rapid convergence, and this is considered a good thing, but at the same time it may cause a reduction or decline in the improvement process, especially if the font size is very large.
- If the education rate is low, this will lead to slow convergence, but this will prevent overshoot and instability during improvement.

**Activation Functions**

Through the inputs of a neuron, the outputs of its activation functions can be determined. Different activation functions may affect the model's performance in terms of speed and complexity of functions during the training period.

**Impact on Optimization Process**

- Tansig considers one of the common tests for activation functions for hidden layers. Through this test, we can help prevent disappearances in the network's gradients, but the network may suffer from a simple problem, which is the disappearance of the gradient, and this may lead to a slow training process.
- Activation functions can be used in the output layer, where we are considered important, as tansig is considered the hidden layer, and trainbfg is considered the output layer. Through these layers, it is possible to ensure that the output activation is compatible with the nature of the network being used. For example, softmax is considered one of the multi-class classification tools.

**A. Other Hyperparameters**

At the same time, other information can also be chosen with great care, for example the number of iterations for both Grasshopper and the neural network. The quality of improvement, network weights and biases, may be greatly affected by the number of iterations of the proposed Grasshopper algorithm.

**Impact on Optimization Process**

- The number of iterations affects the exploitation and exploration of our algorithm, and if a small number of iterations are used, this may cause the space not to be fully or correctly explored. At the same time, if more iterations are used, it may lead to exploring a larger area, but without convergence occurring.
- The process of repetition in training for the proposed algorithm affects the quality of education for our network in training. Also, if the repetitions are insufficient, the performance of our system will be below the acceptable level, but if more repetitions are made, it may lead to much better results.

From what was mentioned, it can be said that the improvement process is directly affected by parameters such as the learning rate, the number of hidden cells, and the activation functions associated with them. Balancing these parameters and fine-tuning them through experimentation and validation on separate datasets or cross-validation can help you find the optimal configuration that leads to high recognition accuracy and efficient training. It's also advisable to monitor the convergence behavior and validation performance during training to identify any potential issues with your chosen parameters.

**Datasets and Pre-processing**

**A. Datasets**

The dataset used for evaluation is obtained from Kaggle [29], but specific details about the dataset's nature, characteristics, variations, and preprocessing steps are not provided in the information presented. However, I can offer some general insights into what such a dataset for English character recognition might entail:

**Dataset Characteristics**

1. **Size:** The dataset likely consists of a collection of images, each representing individual English letters (both uppercase and lowercase).
2. **Number of Samples:** It would likely contain multiple samples for each English letter to provide sufficient training and testing data.
3. **Variations:** The dataset might include variations in fonts, styles, and sizes of the English letters to simulate real-world scenarios where characters can be written in different ways.

**B. Pre-processing**

It is necessary to have pre-processing in preparing the data that will be fed into the neural network, especially when dealing with images, because pre-processing of this data may improve the quality of the input data, remove irrelevant information and noise as well as create the appropriate arrangement and format that enables Neural network of learning. We will mention some of the pre-processing steps on the images that will be fed into the neural network and the role of these steps in improving the network's accuracy.

**1. Grayscale Conversion**

In most networks of this type, we convert the input images to grayscale images as a first step because grayscale images are on only one channel, which leads to reducing the dimensions of the data and computational complexity. Through this step, we will simplify the processing processes

for the neural network and humans as well, and at the same time we will preserve the basic contrast information that helps us distinguish between letters.

**2. Binarization**
In the second step of the work, we convert the grayscale image threshold into a two-dimensional image. Through this, the pixels will be either white or black. This step will greatly simplify the input processes, and this work will also focus the attention of the neural network on the shapes of the characters themselves.

**3. Resize**
I resize the image to a fixed size, and this is necessary, as through this process all images can be made to have the same dimensions. This is essential for a neural network because it prefers to deal with fixed sizes.

**4. Normalization**
The process of equalizing the pixel values of the images that we have changed their size, such as one and zero, prevents the sensitivity of the neural network in terms of the difference in the density of all the pixels, and this will make the process more convergence and stable.

**5. Noise Reduction**
Distortion reduction techniques, such as Gaussian filtering or blurring, can help get rid of differences and defects in the image that may negatively affect our work. Removing or reducing noise helps focus the neural network on only the main aspects of the work.

**6. Data Augmentation**
Applying different transformations to raw data or images, such as rotating, flipping, and scaling operations, is called augmenting the data. By synthesizing data to increase diversity in the training data set, this will strengthen the network's resistance to the different variables that it will encounter during image recognition, and thus this will lead to improved generalization and also prevent over-specialization.

**Importance of Pre-processing for Recognition Accuracy**
Through pre-processing, we can improve the recognition accuracy through the following:

1. Highlighting the main features through pre-processing. This will help the neural network to focus on the main features better.
2. Removing noise and irrelevant details. Through this, pre-processing can ensure greater focus on the work.
3. Changing the size of the input images ensures that the neural network has a consistent format, which leads to more reliable and stable learning.
4. By increasing the data, the data set used for training can be expanded, and this will cause the neural network to deal with more different data, and this will lead to improved performance and improved learning.
5. Changing the images to grayscale and also changing the size in relation to the dimensions without sacrificing the main information and converting the images to binary makes training more efficient.

By incorporating pre-processing steps into the neural network, it leads to learning relevant features and also adapting to different styles of handwriting. This will lead to achieving higher recognition accuracy on both test and training datasets. Because prior operations will have the greatest benefit in improving the network's performance because they will get rid of all impurities, and this will lead to the network focusing on the image only without taking into account the differences and noise present in the training data.

**Research results of the proposed method**
**1. Creating a multi-layer feed-forward neural network to recognize input characters**
The three unique layers of the neural network were arranged hierarchically within an elaborately constructed structural framework. The first layer is considered the information input point. It contains more than 225 neurons, each of which represents one pixel in a network of 15 to 15. After that, we provide the basic inputs in order to get more processors through the input layers that we created, and after that we use the middle layer, which is also considered a hidden layer, and it is a necessary layer for the transformative calculations of the network. This layer can be used, it consists of more than 80 cells. Its function is activation, 'tansig', to help facilitate complex non-linear transformations. After that, we create 52 neurons from the output layers, and it is considered an interface for making the final decision. In the end, the representation of the neuron through the output layer will be either a lowercase or uppercase letter, which represents a wide range of letters that were entered into the system. The application of activation allowed the output layer 'trainbfg' to draw conclusions and inferences more accurately in neural network calculations. Through the final results obtained, we recognize the letters more accurately, which confirms the success of the system with very high accuracy.

**2. Apply the Back-Propagation Learning (BP) Algorithm**
We have used back propagation learning algorithms, which are considered one of the respected and reliable artificial intelligence techniques, throughout the training period in the neural network. We have improved the performance of the neural network and its internal parameters have been adjusted more and more accurately during the course of this iterative process consisting of more than 1000 iterations. For us to achieve a compromise between accuracy and learning speed, the learning rate criterion was 0.6, which is an important parameter as it represents the size of the weights during each iteration. We designed our system to enable us to confirm the ability of the neural network to extract complex relationships and patterns included in the training data. By running the back propagation algorithm, we have a large number of iterations, and through the neural network, we notice that the learning process has become comprehensive thanks to the cooperation between the back propagation learning algorithms and the carefully chosen learning rate. Through this process, the neural network allowed for adjusting and improving the entering data to increase accuracy and thus increase the accuracy of recognition. On the letters.

**3. Include methods for optimizing meta-heuristics**
Through our advanced system Grasshoppers, the strategy used for advanced improvement is explained and included

in a hybrid architecture that we carefully designed to improve the overall performance and increase the accuracy of the MLP. By combining Grasshoppers for Improvement and the MLP structure, a new structure was created that aims to provide efficient and accurate results and is able to adapt to all entered data. Note that the interaction between Grasshoppers and MP brought a new set of improvements to previously existing methods, and this opened the door for the neural network model to be more accurate and reliable. We also note that it provided great hope for progress in artificial intelligence in the near future.

## 4. Assess and contrast with current techniques

Through our method, we conducted a simulation using 26 letters of the English language, both lowercase and uppercase, so that we could evaluate our working system. Additionally, we compared our method with the SVM approach, as mentioned in [28]. The dataset of English letters was obtained from the kaggle site [29] which is help in machine learning create predictive models that recognize alphabets with a variety of fonts. A personal inspiration was to determine whether if machine learning models could learn to predict fonts that are largely distorted but still recognizable by humans (ex. Sans Forgetica – RMIT). Some of the characteristic of this dataset are:

- This collection includes 26 folders (A–Z) that each include 3232 pixel-sized grayscale rendered alphabet characters utilising more than 14900 fonts;
- Additionally, a numpy binary file is offered for simple data loading;
- A demo for loading the numpy binary file into arrays of picture pixels and labels can be found in the kernel section titled "Loading Character Dataset".

In our study, the regulatory parameters were set as follows: 80 hidden layer neurons, a training rate of 0.6, 100 Grasshoppers in the population, and 150 iterations for optimizing the MLP neural network parameters using the Grasshopper algorithm. It's worth noting that we performed independent implementations of the Grasshopper algorithm 10 times. The accuracy of letter recognition was determined using the accuracy criterion, as described in (4):

$$accurcy = \frac{\left(T_P + T_N\right)}{\text{All samples}} \quad (4)$$

Where $T_P$ indicates that the predicted values are equal to the original value and $T_N$ indicates that the negatively predicted values are equal to the original value. In Figure 6 shows the structure of the number of neurons in the layers.

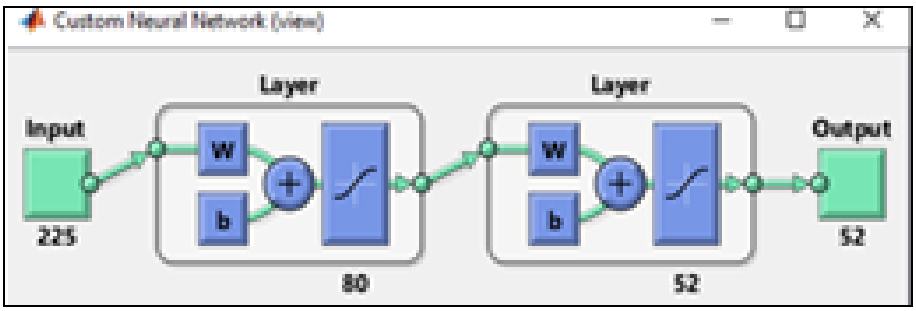


**Fig 6:** Multilayers Perceptron network structure

The next table shows the values obtained from the proposed method and the SVM method [28] and several other methods that have solved this problem. It should be noted that the results in Table 1 are the average obtained from lowercase or uppercase letters.

**Table 1:** Through this table, we compare the results obtained with previous research

| Method name | Capital Alphabet | Small Alphabet |
|---|---|---|
| [29] | 88.46 | 86.00 |
| [30] | 93.7 | 90.2 |
| SVM [28] | 95.74 | 92.19 |
| Proposed method | 94.81 | 93.5 |

Mentioned in the above table and how the proposed method compares to them in terms of accuracy improvements:

### 1. Method [29]

This method achieved an accuracy of 88.46% for capital alphabet recognition and 86.00% for small alphabet recognition. While the specifics of this method are not provided in this context, the proposed method demonstrated a significant enhancement in both uppercase and lowercase recognition. For capital letters, the proposed method improved accuracy by approximately 6.35%, showcasing its superiority. This suggests that the proposed optimization approach outperforms the methodology employed in [29] for English alphabet recognition.

### 2. Method [30]

Method [30] attained an accuracy of 93.7% for capital alphabet recognition and 90.2% for small alphabet recognition. In comparison, the proposed method showcases its effectiveness by achieving higher accuracy in both categories. For capital letters, the proposed method surpasses [30] by 1.11%, indicating an improvement in recognition performance. We note from this article that his method excels by 3.3%, which confirms its ability to increase accuracy for different types of uppercase and lowercase letters.

### 3. SVM [28]

I noticed that the SVM algorithm, which is considered one of the widely known machine learning algorithms, achieved an accuracy rate of 95.74% for recognizing capital letters and the alphabet, and it also achieved a rate of 92.19% for recognizing small letters. Although the accuracy obtained with our proposed method is lower than the SVM method, it remains a competitive method, as a percentage of 93.5%

was obtained for lowercase letters and 94.81 for uppercase letters. However, although it has lower accuracy, it can be considered as a proposed alternative method to the algorithm. SVM.

**5. Analyze and clarify the results**

From the results obtained, we note that our method obtained an accuracy of 94.81% in recognizing capital letters and obtained an accuracy of 93.5 in recognizing small letters in English letters, but although it has somewhat lower accuracy, this method maintains recognition skills. High, while being an effective alternative to the SVM algorithm.

In the end, we say that using methods such as [29, 30] whose algorithms produced varying results for recognizing letters in terms of accuracy, in contrast to the algorithm that was built, which obtained consistently greater accuracy for recognizing small and large letters. Therefore, the results obtained can be compared with the widely known result of the SVM method, and it may have computational advantages due to its good optimization. This comparison highlights how well the suggested approach works to increase identification accuracy and how it may be a promising approach to problems with English letter recognition.

**6. Discussion of results of the study to optimizing English character recognition**

The discussion section aims to provide a comprehensive analysis of the obtained results in the context of the outlined research tasks. The structure follows the chapter outline, emphasizing the key components that contribute to the novelty and significance of the study.

The interpretation of results is crucial to understanding the implications of the study. Each task's outcome is discussed with explicit references to relevant figures, tables, and formulas. For instance, the achieved accuracy rates (as illustrated in Table 1 and (3)) are examined in light of the proposed method's efficiency in character recognition. Figures such as Fig. 6 depicting the neural network structure are referenced to explain the architectural considerations influencing recognition accuracy.

The distinctive features of the proposed solutions are highlighted, offering benefits that distinguish them from existing approaches. A comparative analysis is presented, contrasting the study's results with alternative solutions. For example, the method's superior performance, as evidenced by the substantial improvement over [29, 30], underscores the advantages of the proposed optimization approach. The discussion extends to computational benefits and novel optimization strategies, positioning the study as a promising alternative to the widely used SVM approach [28].

The discussion addresses how the proposed solutions effectively address the problematic aspects identified in Section 2. Through careful argumentation, the evidence base is presented, demonstrating how the study successfully fills the identified research gap. Specific references to figures and tables provide concrete support for the claim that the problem articulated in Section 2 has been mitigated. This achievement is attributed to the integration of Grasshopper and Propeller optimization algorithms, contributing to the robustness of the character recognition system.

We will conclude the discussion by exploring ways of future development. We suggest that future research should focus on the proposed method of adapting to diverse writing fonts and font styles.

**Conclusions**

1. Through our research, we have developed or built a multi-layer feed-forward neural network in order to be able to recognize letters.
2. To build a multi-layer neural network, we used BP learning algorithms to enable it to recognize characters.
3. We use the Grasshoppers method, an optimization tool described in a hybrid architecture to improve the performance of MLP.
4. At this stage, we measured the accuracy of our work and compared it to currently available works. In this study, we aim to improve the accuracy of letter recognition, especially uppercase and lowercase letters in the alphabet. By applying a carefully developed methodology, we have achieved remarkable success. The proposed method showed a significant improvement in accuracy, as evidenced by the recognition rates achieved. For uppercase letters, our method improves accuracy by approximately 6.35%, demonstrating its superiority over previous methods. This indicates that the optimization strategy that has been suggested is superior to the techniques used in [29]. Additionally, the approach performed 3.3% better in tiny alphabet letter recognition than the methods reported in [30]. The remarkable qualitative measure of higher recognition rates highlights the efficacy of our methodology.

Was a secondary objective comparing the performance of our technique with the popular Support Vector Machine (SVM) methodology, as stated in [28]. Although SVM attained competitive accuracy, our suggested method demonstrated its efficacy as a substitute. The recommended technique maintained good recognition skills, with a qualitative indication of 94.81% for capital letter recognition and 93.5% for minuscule alphabet recognition, despite a minor decrease in accuracy when compared to SVM. This comparison shows that, given its benefits in terms of innovative optimization tactics and computing efficiency, our method can be a viable solution for character recognition challenges.

This section will examine and elucidate the findings that were discovered and referenced in the preceding section. Crucially, the suggested technique not only achieved better identification accuracy but also showed tolerance to sparse training data and typeface alterations. Grasshopper and Propeller optimization methods are used to create an efficient character recognition system that works well in a range of scenarios. Because this approach retains excellent accuracy even while dealing with changes in lines and patterns, its practical applications are evident. This qualitative metric is very useful in real-world applications where character recognition has to function consistently in various settings.

To sum up, the study's goals were effectively achieved, resulting in improved character recognition accuracy, competitiveness in comparison to current techniques, and practical applicability in scenarios with font changes. These results highlight the great potential of our suggested

approach to solving character recognition problems and further this subject.

**References**


1. Ptucha R, Such FP, Pillai S, Brockler F, Singh V, Hutkowski P. Intelligent character recognition using fully convolutional neural networks. Pattern Recognition. 2019;88:604-613.
2. Jaderberg M, Simonyan K, Vedaldi A, Zisserman A. Synthetic data and artificial neural networks for natural scene text recognition. arXiv preprint arXiv:1406.2227; c2014.
3. Poznanski A, Wolf L. Cnn-n-gram for handwriting word recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; c2016. p. 2305-2314.
4. Bai S, Kolter JZ, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271; c2018.
5. Araokar S. Visual character recognition using artificial neural networks. arXiv preprint cs/0505016; c2005.
6. Suriya S. Intelligent character recognition system using convolutional neural network. EAI Endorsed Transactions on Cloud Systems. 2020;6(19):166659.
7. Rao NV, Sastry ASCS, Chakravarthy ASN, Kalyanchakravarthi P. Optical character recognition technique algorithms. Journal of Theoretical & Applied Information Technology, 2016, 83(2).
8. Vasudeva N, Parashar HJ, Vijendra S. Offline character recognition system using artificial neural network. International Journal of Machine Learning and Computing. 2012;2(4):449.
9. Doetsch P, Kozielski M, Ney H. Fast and robust training of recurrent neural networks for offline handwriting recognition. In: 2014 14th. International Conference on Frontiers in Handwriting Recognition. IEEE; c2014. p. 279-284.
10. Liyanage C, Nadungodage T, Weerasinghe R. Developing a commercial grade Tamil OCR for recognizing font and size independent text. In: 2015 Fifteenth International Conference on Advances in ICT for Emerging Regions (ICTer). IEEE; c2015. p. 130-134.
11. Farhat A, Al-Zawqari A, Al-Qahtani A, Hommos O, Bensaali F, Amira A, *et al*. OCR based feature extraction and template matching algorithms for Qatari number plate. In: 2016 International Conference on Industrial Informatics and Computer Systems (CIICS). IEEE; c2016. p. 01-05.
12. Bedruz RA, Sybingco E, Quiros AR, Uy AC, Vicerra RR, Dadios E. Fuzzy logic based vehicular plate character recognition system using image segmentation and scale-invariant feature transform. In: 2016 IEEE region 10 conference (TENCON). IEEE; c2016. p. 676-681.
13. Khan WQ, Khan RQ. Urdu optical character recognition technique using point feature matching; a generic approach. In: 2015 International Conference on Information and Communication Technologies (ICICT). IEEE; c2015. p. 01-07.
14. Cilia ND, De Stefano C, Fontanella F, di Freca AS. A ranking-based feature selection approach for handwritten character recognition. Pattern Recognition Letters. 2019;121:77-86.
15. Kaur G, Singh V, Chawla SK, Bhasin M. A survey on character recognition from handwritten documents.
16. Zhu Y, Zhuang F, Yang J, Yang X, He Q. Adaptively transfer category-classifier for handwritten Chinese character recognition. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Cham; c2019. p. 110-122.
17. Such FP, Peri D, Brockler F, Paul H, Ptucha R. Fully convolutional networks for handwriting recognition. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). IEEE; c2018. p. 86-91.
18. Murugan RT, Ramani R. Novel Approaches to Handwritten English Character Alphabet and Number Recognition Based on Neural Networks. IJIRMPS, 2019 May-Jun, 7(3).
19. Garg A, Jindal MK, Singh A. Offline handwritten Gurmukhi character recognition: k-NN vs. SVM classifier. International Journal of Information Technology. 2021;13(6):2389-2396.
20. Setiawan A, Prabowo AS, Puspaningrum EY. Handwriting Character Recognition Javanese Letters Based on Artificial Neural Network. IJCONSIST Journals. 2019;1(1):39-42.
21. Ashlin Deepa RN, Rajeswara Rao R. A novel nearest interest point classifier for offline Tamil handwritten character recognition. Pattern Analysis and Applications. 2019;23:199-212.
22. Gupta A, Sarkhel R, Das N, Kundu M. Multiobjective optimization for recognition of isolated handwritten Indic scripts. Pattern Recognition Letters. 2019;128:318-325.
23. Chowdhury RR, Hossain MS, Islam RU, Andersson K, Hossain S. Bangla Handwritten Character Recognition using Convolutional Neural Network with Data Augmentation. In: 2019 Joint 8th. International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR). IEEE; c2019. p. 318-323.
24. Yousaf A, Khan MJ, Javed N, Ibrahim H, Khurshid K. Size invariant handwritten character recognition using single layer feed forward back propagation neural networks. In: 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). IEEE; c2019. p. 01-07.
25. Reddy RVK, Babu UR. Handwritten Hindi character recognition using deep learning techniques. Int. J Comput. Sci. Eng.; c2019.
26. Xu L, Wang Y, Li X, Pan M. Recognition of handwritten Chinese characters based on concept learning. IEEE Access. 2019;7:102039-102053.
27. Saremi Sh, Mirjalili Sa, Lewis A. Grasshopper Optimization Algorithm: Theory and application. Elsevier; c2017.
28. Katiyar G, Katiyar A, Mehfuz S. Off-line handwritten character recognition system using support vector machine. American Journal of Neural Networks and Applications. 2017;3(2):22-28.
29. Alphabet Characters Fonts Dataset [Internet]. Kaggle. Available from: https://www.kaggle.com/thomasqazwsxedc/alphabet-characters-fonts-dataset.
30. Nasien D, Haron H, Yuhaniz SS. Support Vector Machine (SVM) for English handwritten character recognition. In: 2010 Second International Conference on Computer Engineering and Applications. IEEE, 2010, 1.