Kufa Journal of Engineering
Vol. 17, No. 3, July 2026, P.P. 601-622
Article history: Received 18 October 2025, last revised 17 December 2025, accepted 4 January 2026

HYBRID DEEP LEARNING-CROW SEARCH ALGORITHM FOR ACCURATE ARABIC HANDWRITTEN DIGIT RECOGNITION

Abtisam Abdulelah Salim Azeez

Ministry of Education, Iraq, Email:abtisama80@gmail.com.

https://doi.org/10.30572/2018/KJE/170334

ABSTRACT

This paper proposes a hybrid deep learning framework for Arabic handwritten digit recognition by optimizing Convolutional Neural Network (CNN) hyperparameters using the Crow Search Algorithm (CSA). Due to the high variability and structural complexity of Arabic handwritten digits, achieving optimal CNN performance requires efficient and automatic hyperparameter tuning. In the proposed approach, CSA is employed to optimize key CNN hyperparameters, including filter size, number of filters, mini-batch size, and learning rate, with the objective of minimizing classification error. The MADBase dataset is used for evaluation, and preprocessing steps such as normalization, image reshaping, one-hot encoding, noise reduction, and data shuffling are applied to enhance training efficiency and model robustness. The CNN architecture is trained using the optimized hyperparameters obtained through CSA iterations. Experimental results show that the proposed CSA-optimized CNN achieves 99% accuracy on the testing set, demonstrating strong generalization capability and stability. The findings confirm that CSA effectively improves CNN performance while eliminating the need for manual hyperparameter tuning, making the framework suitable for other image classification tasks.

KEYWORDS

Arabic handwritten digit recognition; Convolutional neural network; Crow search algorithm; Hyperparameter optimization; Deep learning; Image preprocessing.

1. INTRODUCTION

In today's, in the digital age, recognition of handwritten digits is of extreme importance in several areas of computer vision like banking, educational software, postal automation, smart analysing and document scanning, etc (Chychkarov, Y, 2021).

Although there has been progresses made in English and Arabic handwritten digit recognition, it still remains a challenge because of high inter class similarity, different writing styles, complex shapes and diverse forms. Traditional systems rely on a set of features that have been determined manually (Ahmed, S. 2023).

These systems too fail to produce satisfactory accuracy, as the systems do not contour to high levels of noise and alteration. The results obtained from Convolutional Neural Networks for the extraction of spatial features, as well as the classification of images, is of utmost importance (Joy et al., 2025). However, the effectiveness of CNNs heavily depends on optimal hyperparameter selection—such as learning rate, batch size, and the number of neurons—which is difficult and time-consuming to fine-tune manually (Kusetogullari, H. 2021).

To crush this barrier, we need to use hyperparameter optimization more heuristically than ever. The present work incorporates the Crow Search Algorithm (CSA) with CNN to automate hyperparameter tuning with the goal of improving the model's classification accuracy and efficiency. The proposed CNN-CSA framework for hyperparameter tuning is tested on the MADBase dataset containing Arabic handwritten digits, proving the model's strength and efficacy in tackling the difficulty of this recognition task.

1.1. Literature review

Many other works in the field have proposed increasingly sophisticated methods to boost the recognition of Arabic handwritten digits, seeking to improve the precision of machine learning algorithms in differentiating between various digits.

The study (Alsurori, M. H. 2023, October) examines difficulties in recognizing handwritten characters, words, digits, and the complex cursive style of Arabic writing. Different recognition strategies are incorporated and their accuracies analyzed. The best results were obtained using HOG, LBP, and Gabor feature extraction filters and a k-NN classifier, achieving a remarkable 99.88% accuracy. The study, however, demonstrates reliance on methods of feature extraction that are hand-engineered, thus limiting their effectiveness on the diverse writing styles that may be encountered in practice. This approach provides the basis of a system designed to boost the interaction of users with smart devices and to automate documentation. There is a possibility, though, that it will underperform on complex datasets that are often seen in practice. Because

of these considerations the study encourages the use of advanced machine learning and deep learning techniques on Arabic handwriting recognition.

This article (Khudeyer, R. S., 2023) analyses the integration of deep learning and classical machine learning approaches in the field of Arabic handwriting analysis. To increase learning speed and accuracy, the authors suggest substituting the last layer of the ResNet50 convolutional neural network with hyperplanes of support vector machine or random forest classifier. The approach uses three datasets: AHCD, the Alexa Isolated Alphabet Dataset (AIA9K) and Hijja Dataset. As per the results, the hybrid ResNet50-RF model surpassed the other models improved ResNet50 with accuracy rates of 95%, 99%, and 92.4% for AIA9K, AHCD, and Hijja datasets, respectively, as opposed to achieving 92.37%, 98.39%, and 91.5% accuracy for the original ResNet50. However, the model still depends on the architecture ResNet50 and the specific datasets, and still remains somewhat of a black box. The transfer of the model to other Arabic handwriting styles or more diverse larger datasets is still an open question.

This study (Essam, F., 2023) main objective is to compare the performances of seven distinct Machine Learning algorithms – K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression, Neural Networks, Random forest (RF), Naive Bayes, and Decision Tree – and the AUC, accuracy, F1 score, precision, and recall measures for each of the models. The problem to be solved is multi-class classification of handwritten digits in the range of (0-9). The two datasets to be reviewed were the MNIST database and the Handwritten digits classification (HDC) database. The results of the Neural Networks models were outstanding, although, in some of the model comparisons, the results were fairly similar. Although some of the results were quite positive, the study had some limitations. The major one being the sole focus of the study on some of the more traditional datasets, and that the findings may not be generalized to Arabic handwritten digits in particular. The other major limitation to the study, was that it concentrated more on the relative performances of the classifiers than on any attempt to design a new model for recogniton of the digits, or improve on any of the other major issues a particular dataset may pose.

This study (Mohamed, N., 2023, May) was to analyze the recognition of handwritten digits using machine learning to lessen the negative effects of failures in scenarios such as banking processes. The authors trained SVM, MLP, and CNN model on MNIST (read digits 0-9). The study used results (execution speed and recognition accuracy of the digits) to determine the most appropriate model. The study is aimed at the recognition of handwriting technology to facilitate the processing of handwriting inputs from scanned documents, images, and mobile

phones. However, the study does not analyze the most appropriate recognition of Arabic handwritten digits, and does not consider the use of a combination of deep learning to increase recognition accuracy. The authors of the study also do not analyze the problem of enormous variety of people's handwriting that can reduce the accuracy of the model in practical use.

In study (Ghanim, M., 2023) evaluated the recognition difficulty associated with Arabic & English handwritten digits using four different machine learning algorithms. MLP, CNN, Random Forest (RF), and CNN-RF (hybrid) models were used to study digits predominantly used in banking and business in the Arab World and evaluated using MADBase database and English digits from the MNIST database. CNN achieved the highest level of success among the four with 99.11% demonstrating deep learning (DL) — especially CNN — strength in converting handwritten (analog) from various sources like paper, screens, and screenshots to machine-readable, text. Yet this study points out challenges of image processing & pattern recognition die to the changing nature of handwritten characters, different sizes and shapes of digits and features in segmentation and extraction thereof. This ultimately bounds the models to contain very little to no generalization.

In study (Prasad, M. L., 2023, November) outlines a technique for recognizing and classifying handwritten digits through the use of different machine learning techniques. The technique separates each character and utilizes a convex hull to determine the features, followed by classification using Support Vector Machines (SVM). The method specifically recognizes and classifies several handwritten characters by overcoming consistency of character formation, different styles of handwriting, and jumbled/disorderly structures. The proposed technique effectively overcomes some of the challenges, however, the method has not fully addressed some of the issues including, poor handwriting and characters being highly overlapped. Lastly, the method has been mostly tested on standard datasets, which limits the method from being used on other handwriting styles and datasets.

in study (Ahmed, S. S., 2023) Addressing the handwritten digit recognition, examines digit counting and classification from 0 to 9 using the EfficientDet-D4 model and a B4 Annotator. This tackles the foundational issues such as varying styles of handwriting and different image noise and artifacts, including varying intensities, blur, and noise. Outstandingly, the model reached 99.83 accuracy on MNIST while obtaining 99.10 in a cross-dataset evaluation on the USPS, which reproduced strong generalization even in the presence of unseen noise and blurring, as well as variations in position and size of the digits. Despite the accuracy of the model, there are concerns in terms of the computational power required for this model, especially in online and real time usage and for bigger datasets. It will certainly work on the

MNIST and USPS datasets, and there are no cases to support its effectiveness on Arabic handwritten digits, and on highly general multilingual datasets to date.

This review paper (Mezghani, A., 2023) captures advancements in machine learning and deep learning concerning the recognition of Arabic handwritten characters. The review presents a comparative and detailed assessment of recognition systems concerning the overall recognition systems, and describes the systems using holistic, analytical, and segmentation approaches. The review attempts a comparative analysis of deep learning and machine learning, and the impact of each on the recognition of handwritten characters. The review consolidates the segments of the recognition system, such as the pre-processing, the feature extraction, and the segmentation steps, and describes the recognition system in its standards, and categorizes them, providing future generations desiring extensive knowledge in the domain of research. The review paper is extensive, and the review paper lacks the construction of a recognition system, and lacks the construction of empirical work. The review papers are primarily theoretical, the review paper, and the framework proposed are intended for the attainment of novel experiments, where the empirical work of the review paper guides, and the theoretical framework proposes valuable lessons.

This study (Fateh, A., 2024) suggests a new framework for recognizing handwritten digits that builds upon current methods, which limit themselves to a single alphabet or a few documents. The framework aims to address 12 languages, each having its own language module fully operating in the latent space of a neural network. The framework incorporates the principles of transfer learning along with a laterally shifted MRA module to reduce the difficulty of recognition and limit the margin of recognition required for the image. The study shows a 0.60 improvement with respect to the previous models for a few languages while maintaining the same level of accuracy and a number of other positive indicators. However, the study still has a number of limitations including the extensive computational power the framework may require given its complexity and multi-language setup and the fact that its performance in the real world with different handwritten documents has not been fully verified outside the documents tested. Also, while the framework does have the ability to perform in a number of languages, it still might require some particular customizations to work with certain blended scripts or for very irregular handwriting.

This research (Mutawa, A. 2024) In an attempt to mitigate the marginalization of the Arabic language, introduces a different deep learning strategy to recognize Arabic handwritten text. The strategy consists of a hybrid of two deep neural networks. The first of which is a ResNet that is used to obtain features of images which is then combined with a BiLSTM and CTC to

be used for sequence modeling. The proposed system surpasses most of the comparable systems with a character error rate and word error rate of 13.2 and 27.31, respectively. However, some limitations of this research should be acknowledged. The model is tested against a small dataset and more research should be done to determine how the model can be used with different handwritten Arabic text styles. The increased structural complexity of the hybrid model is also a problem for the proposed model for systems that require real-time processing.

The variability of handwriting, such as different writing styles, cursive links, and varying forms of characters, is a prominent problem for recognizing Arabic handwritten digits. Numerous classical machine learning techniques depend on bespoke designed features, which often extract complex patterns based on spatial arrangements and are prone to noise. In deep learning, models like CNNs and even hybrid architectures, struggle with generalization across diverse datasets and have to be tuned extensively for hyperparameters to reach the ideal performance goal. For one, the majority of techniques in use do not incorporate metaheuristic optimization algorithms or attention mechanisms which would otherwise better feature selection and recognition accuracy. This results in diminished robustness and practical performance

To address these challenges, the objective of this study is to propose a hybrid deep learning framework as a robust recognition system that optimally integrates a Convolutional Neural Network (CNN) with the Crow Search Algorithm (CSA) for Arabic handwritten digit recognition. The CNN component examines and extracts appropriate levels and strides to construct robust spatial representations suitable for optimal recognition. The Crow Search Algorithm aims to optimize the learning process by tuning hyperparameters such as the learning rate, batch size, and etc, to achieve optimal computational performance. By reducing the need for manual tuning through automated optimization, the proposed model attains superior performance. The integrated approach preserves the strengths of deep learning while reducing the complexity of Arabic handwritten recognition systems and enhancing generalization capability to achieve optimal performance across diverse datasets.

Authors' Contributions:

The author solely designed the research framework, implemented the data preprocessing and CNN model, applied the Crow Search Algorithm for hyperparameter optimization, analyzed the results, prepared figures and tables, and wrote the manuscript.

2. METHODOLOGY

This research paper analyses the use of the Crow algorithm in the hyperparameter optimization of CNN architectures tailored for the recognition of handwritten Arabic digits. Although the characters are handwritten, the increasing complexity and diversity of Arabic characters leads

to the to the necessity of implementing advanced techniques for fine and effective model training. The Crow algorithm serves as an optimization method for improving the learning capabilities of the model, and as such, hones key hyperparameters to include filter size, number of filter, mini batch size and learning rate.

Information gathering is the first stage where data is collected, then put on hold, until training becomes necessary. The procedure of data preprocessing is extremely important, and it is powerful since it has the greatest influence on the quality and effectiveness of the finished, generated model. The primary intention behind preprocessing is to customize the data, and change its form so that it becomes more compatible with deep learning models, and thereby, it makes feature extraction easier. This article focuses on simplifying the stages of preprocessing to a few fundamental actions.

To start, we will load and analyze raw and unstructured data. We will make use of diagrams that have images and their captions in CSV files, along with the raw training and testing samples that will be inputted into the model. We will then move on to the normalization and transformation steps. This process involves changing the matrix structure of the data to two-dimensional images and vice versa, so we will transform each image matrix that is an array of 28 pixels in height and 28 pixels in width, each containing a grayscale value, to a 28x28 image. In addition, we will normalize the training pixel values to be between 0 and 1. This will enhance the ability of the deep learning model to identify different features in the training data set, and will accelerate the convergence of the model training.

In order to model the system capabilities to differentiate various patterns, data labels will be processed using one-hot encoding approaches. This technique takes the labels of the data class and changes them into unique vectors, creating a unique class specific vector for each class. This allows the data to be processed in a neural network model. The data is then subjected to some initial exploration to identify outliers or missing values. The data will and has been rotated with random shuffling to improve data quality by reducing noise. This reduces the risk of overfitting and increases the practical utility of the model. The preprocessing stage in this article, data has been loaded and transformations of the structure have been applied, the data has been normalized, and one-hot encoding has been applied to the labels. The out-of-range data is then identified and removed, and the remaining data is opportunistically shuffled. The model will be able to analyze the data and produce high quality valuable patterns with the utmost efficacy.

Afterwards, the CNN structure is built, and the precise locations of the population are determined by applying the Crow algorithm to the wide-ranging CNN hyperparameter

configurations (Al Akabi et al., 2025). Convolutional Neural Networks (CNNs) are sophisticated deep learning networks designed for processing images and spatiotemporal data (Mohammed et al., 2022). CNNs consist of convolution layers that extract features like edges, textures, and patterns, followed by pooling layers to downsample the features and reduce computational cost. Finally, fully connected layers (FC layers) are used to recognize patterns and generate outputs (Nagesh et al., 2025). Activation functions such as ReLU introduce non-linearities, as shown in Eq.1:

$$f(x) = \max(0, x) \tag{1}$$

The entire network adjusts weights during training to minimize error.

On the other hand, the Crow Search Algorithm (CSA), which is based on the intelligent food-hiding behavior of crows, optimizes hyperparameters of CNN. Each crow is a solution (a set of hyperparameters), and he has a memory of the best position. In every iteration, crows move toward the best position of other crows, while trying to avoid being seen. This algorithm efficiently optimizes the filter size, number of filters, mini-batch size, and learning rate to improve the performance of CNN.

Within the CNN process, the first few convolution layers focus on determining essential characteristics from the input. These characteristics are then made more compact with the following pooling layers, and the final output is then generated through the fully connected layers. In the course of training, optimization methods, such as gradient descent, are used in order to reduce error on predictions and to adjust the weights. In this regard, the Crow algorithm optimizes the hyperparameters in a concurrent manner, thus enhancing the accuracy with which CNNs can generalize to the task of recognizing handwritten Arabic digits.

Within this approach, the model's effectiveness will be evaluated based on the CNN error, and the Crow Search Algorithm will stop after a predefined number of iterations, retaining the best hyperparameters. The update rules of the crows' positions are defined as follows (Eqs.2 and 3):

$$X_i^{t+1} \begin{cases} x_i^t + r \,.\, fl_i^t \left(M_j^t - M_i^t\right), if\ r\ \geq AP_j^t \\ random\ position \quad ptherwise \end{cases} \tag{2}$$

X_i^t : $Position\ of\ crow\ i\ at\ iteration\ t$

M_j^t : $Memory\ of\ crow\ jjj\ (best\ stored\ position)$

fl_i^t : $Flight\ length\ of\ crow\ i$

AP_j^t: $Awareness\ probability\ of\ crow\ j$

r: Random number between 0 and 1

$$M_i^{t+1} = \begin{cases} x_i^{t+1}, if\ (x_i^{t+1}) < f(M_i^t) \\ M_i^t, Otherwise \end{cases} \quad (3)$$

Where f (·) is the objective function (e.g., CNN error).

When the algorithm satisfies the stopping condition, the best solution achieved corresponds the to hyperparameters set which minimizes the error of the CNN. This method allows the simultaneous optimization of various hyperparameters of a CNN, increasing the model's accuracy and generalization ability on the classification of handwritten Arabic digits, all without any manual hyperparameter optimization.

Fig. 1 illustrates the layout of the proposed method and the main steps for using the Crow algorithm to optimize the hyperparameters of the CNN.

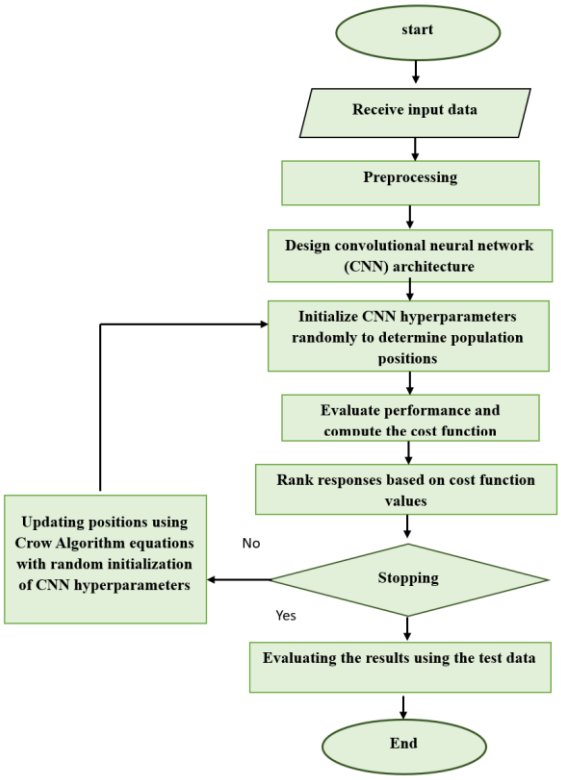

Fig .1. Method of the proposed method

3. RESULT AND DISCUSSION

3.1. Dataset

This research is based on a dataset Modified Arabic Digits Database, popularly used as a benchmark dataset for evaluating machine learning models in Arabic handwritten digit recognition is MADBase. It contains 10,000 test images and 60,000 training images depicting Arabic handwritten digits of 0 to 9. Each image is a 28x28 pixel grayscale digit, which is ideal for a CNN bounded framework. The dataset possesses various styles of handwritten digits, and different types of digits, which makes it very resourceful for training and evaluating recognition

algorithms. It is especially useful because it captures the differing degrees of variability in the writing of Arabic digits, such as changes in pen thickness, curvature, and orientation, which can vary a lot from person to person. Because of these features, the dataset is useful for determining the generalization and robustness of machine learning models. In this research, MADBase is used to train and test various models while applying the Crow algorithm to tune the hyper parameters of the CNN to improve the model's performance. The dataset MADBase provides significant information on the recognition of handwritten Arabic digits using the methods proposed in this research. (Ahlawat, S., (2020).

3.2. Evaluation Metrics

This article examines the performance of the epoch precision of the CNN model on the handwritten digits using the Crow algorithm with respect to the accuracy, precision, recall, and F1-score metrics. Each of these metrics provides different insights into the classification capabilities of the model. In depth domain analysis, the strengths and the weakness of the model with respect to the performance evaluation metrics is established.

3.2.1. Accuracy

Accuracy, one of the most basic evaluation metrics, refers to the proportion of observed samples that have been correctly identified within the dataset. It can be expressed within the context of the prediction evaluation frame as the number of correct predictions, which include true positives and true negatives, divided over the number of all predictions made within the dataset. Although useful, accuracy as a single descriptor of a model's performance does not account for the possible bias class prediction accuracy observed within imbalanced datasets. Such class predictions can dominate accuracy scores and, as a result, be considered misleading. This is the accuracy formula:

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

3.2.2. Precision

Among all the instances that the model predicted as the positive class, precision measures the percentage of true positive predictions. This metric is particularly useful when the cost of false positives is high, and its focus is on the correctness of positive predictions. In the problem of recognizing handwritten Arabic digits, precision indicates how often the model successfully identifies a specific digit correctly when predicting it. The precision equation is as follows:

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

Precision is particularly important when the goal is to reduce false positives; that is, to ensure that when the model makes a prediction, there is a high probability that the prediction is correct.

3.2.3. Recall or Sensitivity or True Positive Rate

Recall (which some refer to as sensitivity) is defined as the proportion of actual positive samples to all the actual positive cases. This metric describes the extent to which the model retrieves relevant samples of a defined class. When considering the recognition of Arabic digits, recall indicates the extent to which the model successfully identifies a particular digit in the data. High recall scores suggest the model successfully identifies the pertinent cases, sustaining a low false negative rate. Recall is computed as follows:

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

By focusing on identifying all relevant instances, recall improves precision and is particularly important in applications were missing a positive sample—that is, a false negative—may have serious consequences.

3.2.4. F1-Score:

The F1-score is an example of a balanced metric since it includes consideration for false positives and false negatives. It is calculated as the harmonic mean of precision and recall. The F1-score becomes handy when dealing with unbalanced datasets when accuracy by itself is not enough to adequately characterize the model's effectiveness. The F1-score serves the purpose of establishing a balance between precision and recall. The F1-score's best definition is captured by the following equation.

$$\text{F}_1\text{Score} = \frac{2\times(\text{Precision}\times\text{Recall})}{\text{Precision+Recall}} \tag{7}$$

Assessing the overall performance of the Convolutional Neural Networks regarding the recognition of handwritten digit and optimized using the Crow algorithm is best done using the F1-score metric. This is important because it shows the model has precision as well as recall. This particular thesis will analyze in detail the results from the model after classifying it using several different metrics and will prove that the method is indeed reliable and accurate for use in real world situations.

3.3. Simulation results

This segment shall dissect the findings acquired in relation to the suggested approach in a sequential one. The first step is in the employment of the method. In this phase exemplars of the model were utilized to test and refine the algorithm accuracy. First of all, the described model is trained and the images were uploaded, which were both the training images and the unstructured test images. These images were packaged into a collection of HyperText Markup Language accompanied with the files labelled with other metadata, images and accordingly

represented in an unstructured way. In the subsequent step, the data were modified to fit the model and evaluated via normalization. The transformed data comprises 28x28 matrix images where each of the images are monochrome. Each of the pixels' magnitudes also were limited to be between 0 and 1, to train the model better and to yield even better accuracy.

In the next phase, the one hot approach on the labels was performed now, which was necessary for aligning the data with the neural network model. The model's training stages aimed at processing the labels, hence, the unique vector was formulated for each class. The data, examining for any deviations or outliers, were checked to validate the dataset's cleanliness and trustworthiness for the model's training. In addition, the final dataset was reserved which the model trained himself 80 out of the 100 files, the rest for testing the model's performance. The training data was further improved with noise decreasing techniques to provide the model better accuracy against the real world. The model was further confused with noise and improved the data accuracy against the rest of the world. Furthermore, noise was randomly added and the model confused; this further improved the data accuracy against the rest of the globe. Part of the data after preprocessing is illustrated in Fig. 2 to enhance comprehension of the data preparation procedure.

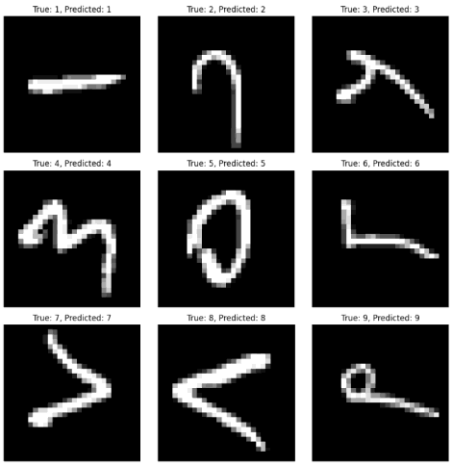

Fig .2. A portion of the data after preprocessing

Now that the preprocessing stage has been completed, the next step in convolution neural networks was done with the following architecture for classification.

1. Input Layer: The model input is a 28x28 grayscale (single-channel) image. This is the standard size for similar datasets, such as MNIST, as it enables the network to pull out meaningful components for a given image.

2. First Convolutional Layer: This layer employs 32 filters (of 32×3×3) with the ReLU activation function. Edge detection and identifying fundamental patterns in images are the sorts of basic features that are targeted with this size of a filter. The popularity of 32 filters is because it achieves a promising performance with reasonable model complexity.

3. Max-Pooling Layer: The 2x2 pool size of this layer is meant for preserving the essential features of the model while reducing the dimensions, which in turn lowers the risk of overfitting and decreased computational complexity.
4. Second Convolutional Layer: This layer has 64 filters (of size 3x3) that have the ReLU activation function. It allows the model to seize more complex features from the images, together with the increase of filters which deepens the model.
5. Second Max-Pooling Layer: Like the first pooling layer, this second layer improves the usage of the computer by lowering the dimensions.
6. Flatten Layer: This layer converts the two-dimensional output into a one-dimensional format that can be used in the following layers.
7. Dense Layer: 128 units are employed here with the ReLU activation function. A typical choice is 128 units as it provides enough capacity for learning to solve most medium-level tasks without making the model unnecessarily complex.
8. Dropout Layer: A dropout rate of 50% is applied to prevent overfitting. This layer ensures that the network learns more general features, focusing on shared characteristics rather than overfitting to the specific dataset.
9. Output Layer: This layer consists of 10 units with a softmax activation function for multi-class classification. The number of units corresponds to the number of classes (in this case, 10 classes).

As mentioned, the Crow algorithm was employed to optimize hyperparameters such as filter size, number of filters, batch size, and learning rate. The specifications of this algorithm are presented in Table 1.

Table 1. Specifications of the Crow algorithm

Values	Parameters
3-7	Filter size
32-128	Number of filters
16-64	Batch size
0,01 – 0.0001	Learning rate
10	Number of particles
10	Number of iterations

As shown in the table above, the stopping condition of the Crow algorithm is set to 10 iterations. This value was determined based on a sensitivity analysis of the algorithm with respect to the number of iterations. In this analysis, the effect of the number of iterations of the Crow algorithm optimization algorithm on the accuracy of the best-obtained solution was examined. in Fig. 3 On the horizontal axis, the number of iterations (10, 15, and 20) is shown, and on the vertical axis, the best accuracy obtained from running the algorithm is displayed. As observed in the chart, increasing the number of iterations from 10 to 15 and then 20 gradually decreases

the accuracy of the best solution. At 10 iterations, the highest accuracy (above 99%) is achieved, but as the number of iterations increases to 15 and 20, this accuracy drops to around 98.92%. This decrease in accuracy with increasing iterations can be attributed to several factors. One possible reason is premature convergence of the algorithm. At the beginning of the execution, the algorithm reaches more optimal solutions, but with additional iterations, it may get trapped in local optima and fail to find better results. In this situation, the algorithm is no longer able to explore new optimal points or improve previous outcomes. Additionally, increasing the number of iterations may reduce the diversity within the Crows population (solvers). At the start, the crows population has higher diversity and can explore different regions of the search space, but as iterations increase, this diversity diminishes, and the algorithm tends to move toward points that are not functionally optimal. Moreover, more iterations may lead to resource exhaustion and wasted computational time without yielding significant improvements in solution quality. Based on the results of this analysis, it can be concluded that using 10 iterations as the stopping condition is optimal, as the accuracy reaches its highest point at this stage. Beyond this, increasing the number of iterations reduces accuracy. Therefore, employing 10 iterations as the termination point of the algorithm provides the best balance between accuracy and computational time.

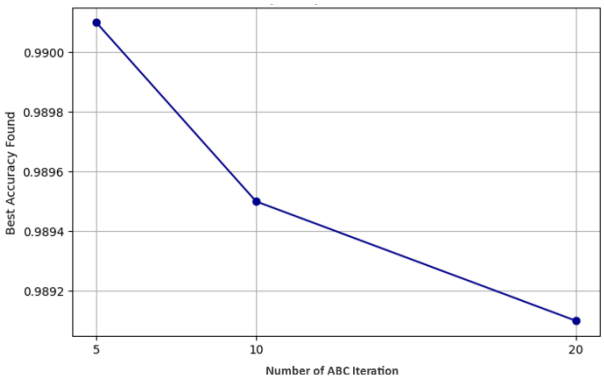

Fig. 3: Sensitivity analysis of the algorithm with respect to different numbers of iterations

Finally, the optimal values for the hyperparameters—filter size, number of filters, batch size, and learning rate—were determined by the Crow algorithm as 3×3, 32, 64, and 0.001, respectively. Fig. 4 illustrates the model's accuracy progression during the training process for each epoch, both on the training and validation datasets. The horizontal axis represents the number of epochs, while the vertical axis shows the accuracy levels. As observed, the training accuracy increases rapidly from the very beginning and quickly approaches high values close to 100%. The validation accuracy follows a stable trend aligned with the training accuracy and remains near the training level until the end of the training process.

A key and valuable observation from this curve is that there is no significant gap between

training and validation accuracy. This clearly indicates that the model has not overfitted; in cases of overfitting, training accuracy would reach high levels while validation accuracy would drop or remain at lower levels. In this chart, the validation accuracy is not only close to the training accuracy but also exhibits appropriate stability, with minor fluctuations across epochs being natural and expected.

This strong performance of the model can be attributed to the optimal tuning of hyperparameters achieved using the Crow algorithm. Selecting the optimal number of filters, learning rate, kernel size, number of layers, and other critical parameters enabled the network to learn effectively from the data without becoming memorization-prone or overfitting the training samples. In other words, the model has successfully learned meaningful and generalizable features, achieving strong and reliable performance on both seen (training) and unseen (validation) data. In summary, the curve in Fig. 4 confirms that after effective optimization using the crow algorithm, the convolutional neural network not only achieves near 100% accuracy but also maintains a balance between performance on training and validation datasets, preventing overfitting and demonstrating a fully stable, precise, and generalizable model.

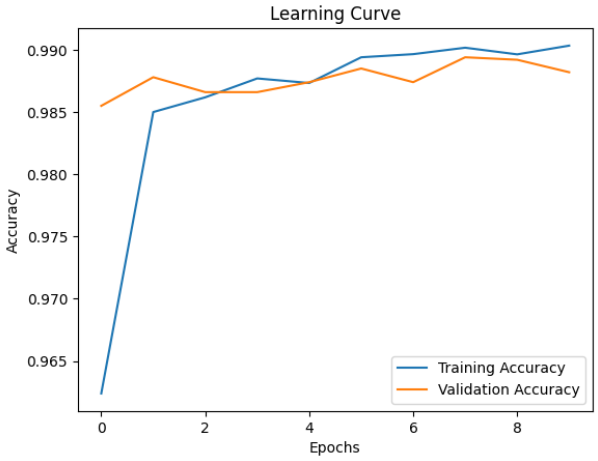

Fig. 4. Learning curve of the CNN model after hyperparameter optimization using the Crow algorithm

3.3.1. Analysis of the results for the training and testing datasets based on the confusion matrix

One of the important tools for evaluating performance in classification tasks is the confusion matrix, which provides an organized and understandable means to visualize and interpret the model's prediction results. This matrix consists of four main components: FP, FN, TP, and TN. True Negatives represent instances where the model correctly predicted the absence of a class, while True Positives indicate instances where the model correctly predicted the label of a class. Not predicting an actual case result into False Negatives, while predicting an actual case incorrectly results into False Positive. This is essential in defining additional measurements of

precision, such as accuracy, recall, and F1-score, since they provide more pointers than just basic accuracy. This is critical for ascertaining behaviour of class imbalances. The confusion matrix is more valuable in handwritten digit recognition as it shows which digits tend to get mistaken for one another the most. For example, more mistakes may happen with some Arabic digits that look alike or when the cursive writing of digits is cursive. On the other hand, assessing True Positive and False Negative across various classifications gives a clearer perspective concerning the model and its shortcomings. Fig.5 and 6 provide confusion matrices of the model for both training and testing datasets to demonstrate the classification performance in a more tangible manner. These matrices yield insights into how well the model generalizes and enable us to interpret the performance gap in training and unseen testing sets. This careful analysis in real world situations is to make sure the final model is accurate as well as dependable for all digit classes.

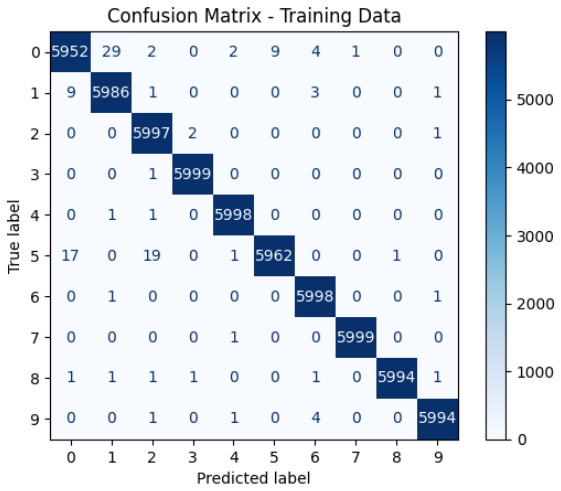

Fig. 5. Confusion matrix of the training dataset for handwritten digit recognition

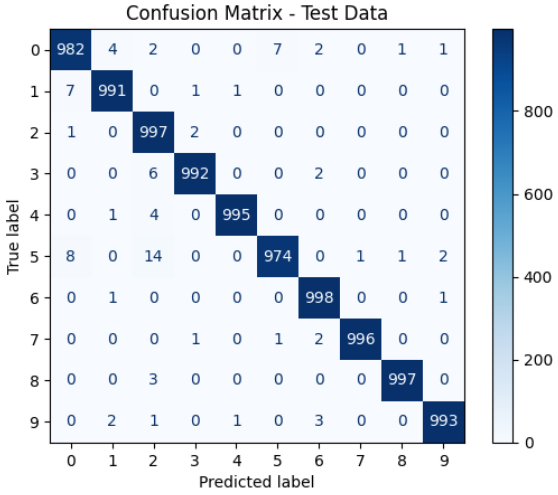

Fig. 6. Confusion matrix of the test dataset for handwritten digit recognition

3.3.2. Analysis of the Test and Training Dataset Results Based on the Curve

The ROC curve is pivotal in measuring the performance of classification algorithms, particularly in binary classification skirmishes (though can also be used in multiclass contexts). The ROC curve assesses the True Positive Rate (TPR) also referred to as Sensitivity or Recall, as compared to the False Positive Rate (FPR) on varying threshold values. The FPR is the fraction of negative samples incorrectly predicted as positive while the True Positive Rate is the proportion of positive samples that the model successfully predicted. The ROC curve indicates the model's discriminatory power of the classes in binaries by assessing the equilibrium across various thresholds between TPR and FPR. Models that have ROC curves close to the upper left corner of the graph, meaning a high true positive rate and low false positive rate across all thresholds, show the best model performance on the plot. Also the curve

gives rise to its another important feature the Area Under the Curve (AUC) which is always between 0 and 1. AUC values of 0.5 means model performance is equally to random guessing while 1 means the model is perfectly classifying the data, delineating between classes with no mistakes.

Each class in a multi-class problem can potentially have its own ROC curve. For instance, in recognizing handwritten Arabic digits, a model is evaluated over all the digits. For this research, we constructed the ROC curve to assess the performance of the Crow algorithm to CVNN model on the test dataset. The ROC curve for the test dataset sheds light on the model's overfitting and underfitting abilities and reveals the certainty that the model classifies the data. This model's overfitting to the test dataset is visually presented in Fig. 7, which graphically summarizes the model performance over the test set. The ROC curve shows that the model captures several true positive factors with very few false positive factors for most underlying numeric classes. This confirms that the model reliably marks the right and wrong predictions. Furthermore, the Area Under the Curve is nearly 1, which confirms that the model recognizes Arabic handwritten digits AUC almost equals to 1.

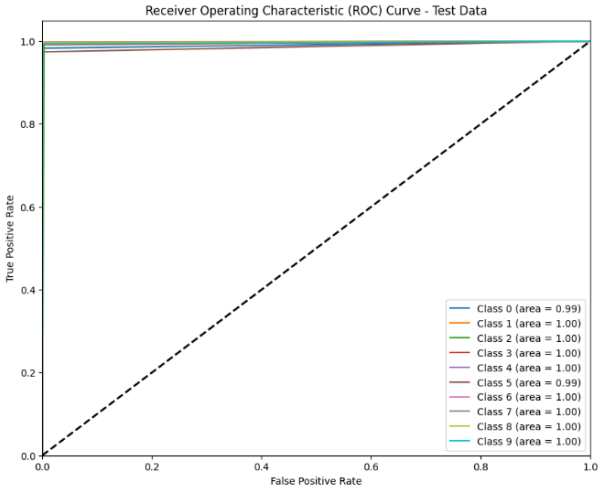

Fig. 7. ROC Test of the Dataset for Handwritten Digit Recognition

3.3.3. Evaluation of Training and Test Dataset Results Based on Introduced Metrics

This section discusses results of classifying the digits based on the CNN model optimized through the Crow algorithm for assessing handwritten digit recognition and determining accuracy, recall, and the F1 score. These metrics effectively outline how the model classifies various digit classes for training and testing datasets. Model results are shown in Fig. 8 and 9 as bar graphs corresponding to the training and testing datasets. From the chart in Fig. 8**,** the model which used the training set was able to achieve an accuracy of 100%, recall of 100%, and an F1 score of 100%. These results demonstrate that the model has learned to identify the key features of Arabic handwritten digits. Conversely, the model results maintained their

performance on the test datasets as shown in Fig. 9 and earned an accuracy of 99%, recall of 99%, and an F1 score of 99%. This result indicates that the model was able to generalize the learned information effectively. This fact demonstrates the effectiveness of the CNN model that has been modified with the Crow algorithm to increase its performance on Arabic handwritten digit recognition for used in real-world scenarios.

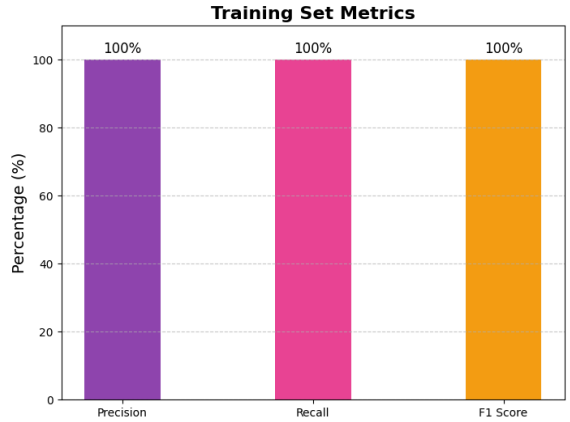

Fig. 8. Performance chart of handwritten digit recognition for the training dataset.

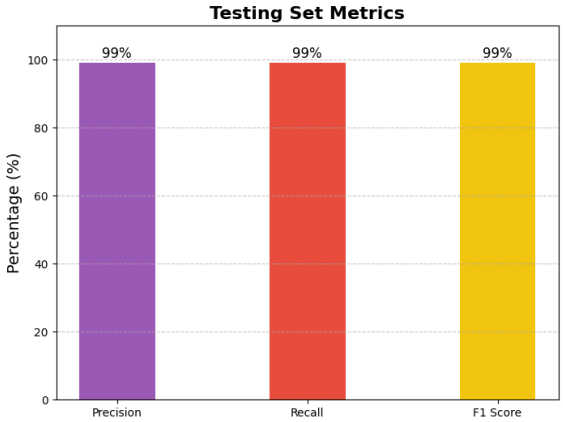

Fig. 9. Performance chart of handwritten digit recognition for the testing dataset.

3.4. Comparison

A multitude of techniques have been explored in the area of Arabic handwritten digit recognition employing a range of classical machine learning and deep learning approaches. This section juxtaposes the results of our proposed model, a CNN deep learning framework, optimized through the Artificial Crow algorithm, which attained 99.3% accuracy, as well as other studies employing varied techniques for digit classification.

In reference (Ghanim, M. F., 2023).

four models — Random Forest (RF), Multilayer Perceptron (MLP), CNN, and a hybrid CNN-RF model — were evaluated for recognizing Arabic and English handwritten digits using the MNIST and MADBase datasets. Among these models, the standalone CNN provided significantly better performance than the others, achieving an accuracy of 99.1%. While the RF and MLP models performed reasonably well, they faced challenges when dealing with the complexity of unstructured image data, and even the hybrid CNN-RF model could not reach the accuracy of the pure CNN.

Reference (Alkhateeb, J. H. 2020). proposed a CNN-based approach for Arabic handwritten digit recognition that achieved an accuracy of 94.3% using the MADBase dataset. The author noted that fine-tuning hyperparameters such as padding and stride could further improve model performance. Although the results of this study were promising, no algorithm was implemented for hyperparameter optimization, and parameters like filter size and learning rate were manually

adjusted. Our proposed method, i.e., CNN optimized with the Crow algorithm, overcomes this limitation by employing the algorithm to automatically tune hyperparameters, providing optimal performance for the model without manual intervention.

In summary, comparisons with these studies indicate that the CNN optimized with the Crow algorithm not only outperforms traditional machine learning methods and manually tuned CNNs but also shows significant improvement in accuracy compared to other advanced models Table 2. While CNNs are inherently well-suited for handwritten digit recognition tasks, combining this architecture with the Crow algorithm in our proposed method provides an additional advantage, as it optimizes the CNN structure according to the specific characteristics of the MADBase dataset. This optimization has resulted in a high accuracy of 99.3%, making our model one of the most competitive methods for Arabic handwritten digit recognition.

Table 2. Comparison of Our Proposed Method with Previous Studies

ACCURACY (%)	DATA SET	METHOD	REFERENCES
97.1	MADBase dataset	Random Forest (RF), Multi-layer Perceptron (MLP), Convolutional Neural Network (CNN), and a hybrid CNN-RF	(Ghanim et al., 2023)
94.3	MADBase dataset	CNN-based approach	(Alkhateeb, 2020)
99	MADBase dataset	CSA-optimized CNN model	Proposed Method

4. CONCLUSION

Recognizing hand written digits is a difficult problem in computer vision and machine learning in general and it is challenging with complex scripts like the Arabic language. Conventional machine learning models may have difficulty when dealing with unstructured image data, and need to develop higher forms of representation in order to make sense of the data. A good and reliable system for digit recognition is important, and necessary in many practical applications ranging from making handwritten documents to scanned forms to enhancing machine print in automated.

The paper thus proposes a model in which CNN hyperparameters are optimized through the use of an Crow algorithm. Whereas it is true that image-based tasks are better performed with CNNs, their performance largely depends on the right selection of hyperparameters such as the number of filters, filter size, learning rate, and mini-batch size. The Crow algorithm integrated into the system automatically selects proper hyperparameters so that manual fine-tuning will not be involved. This proposed model is tested over the MADBase dataset containing Arabic handwritten digits to test how much recognition accuracy can be improved if a CNN is optimized by Crow algorithm.

It attained 100% accuracy on the training data and 99% on the test data, hence an indication of great generalization ability. Such high accuracies validate the model's competence in classification for digit recognition from Arabic handwriting with rather optimal conditions to avoid any overfitting risk-verified explicitly through performance proximity between training and testing sets. The proposed method is advantageous from several perspectives because Crow algorithm hyperparameter optimization is enhancing not only model performance but also making the hyperparameter selection ready and costless. That automatic optimization allows this model to be applied to different datasets and domains without intensive manual interventions. Also, the crow algorithm lets the CNN fine-tune to the particular details of the data getting high exactness with no trial-and-error changes needed. The way our plan can keep both accuracy and speed makes it a useful addition in handwritten digit spotting and hints at possible use for other image-based grouping jobs.

5. REFERENCES

Ahlawat, S., Choudhary, A., Nayyar, A., Singh, S. and Yoon, B. (2020) 'Improved handwritten digit recognition using convolutional neural networks (CNN)', Sensors, 20(12), p. 3344 https://doi.org/10.3390/s20123344.

Ahmed, S.S., Mehmood, Z., Awan, I.A. and Yousaf, R.M. (2023) 'A novel technique for handwritten digit recognition using deep learning', Journal of Sensors, 2023(1), p. 2753941 https://doi.org/10.1155/2023/2753941.

Al Akabi, H. and Al-assadi, T. (2025) 'Classification of fault signals based on DCT and deep learning', Kufa Journal of Engineering, 16(4) https://doi.org/10.30572/2018/KJE/160413.

Alkhateeb, J.H. (2020) 'Handwritten Arabic digit recognition using convolutional neural network', International Journal of Communication Networks and Information Security, 12(3), pp. 411-416.

Alsurori, M.H., Mohsen, A.A., Al-Badani, G., Al-Nahari, M., Faisal, T. and Alkasem, S. (2023) 'Review on Arabic handwritten recognition using deep learning and machine learning', in (2023) 3rd International Conference on Emerging Smart Technologies and Applications (eSmarTA), IEEE, pp. 1-8. 10.1109/eSmarTA59349.2023.10293409.

Chychkarov, Y., Serhiienko, A., Syrmamiikh, I. and Kargin, A. (2021) 'Handwritten digits recognition using SVM, KNN, RF and deep learning neural networks', CMIS, 2864, pp. 496-509. 10.32782/cmis/2864 44.

Essam, F., Samy, H. and Wagdy, J. (2023) 'Mlhandwrittenrecognition: Handwritten digit recognition using machine learning algorithms', Journal of Computing and Communication, 2(1), pp. 9-19. 10.21608/jocc.2023.282076.

Fateh, A., Birgani, R.T., Fateh, M. and Abolghasemi, V. (2024) 'Advancing multilingual handwritten numeral recognition with attention-driven transfer learning', IEEE Access, 12, pp. 41381-41395. 10.1109/ACCESS.2024.3378598.

Ghanim, M.F., Mohammed, A. and Sali, A. (2023) 'Arabic/English handwritten digits recognition using MLPs, CNN, RF, and CNN-RF', Al-Rafidain Engineering Journal (AREJ), 28(2), pp. 252-260. https://doi.org/10.33899/rengj.2023.138592.1236.

Hussien, A.G., Amin, M., Wang, M., Liang, G., Alsanad, A., Gumaei, A. and Chen, H. (2020) 'Crow search algorithm: theory, recent advances, and applications', IEEE Access, 8, pp. 173548-173565. 10.1109/ACCESS.2020.3024108.

Joy, P. and Jebadurai, I.J. (2025) 'Detection of hair fall and scalp disorders through ML and image processing', Kufa Journal of Engineering, 16(4) https://doi.org/10.30572/2018/KJE/160406.

Ketkar, N. and Moolayil, J. (2021) 'Convolutional neural networks', in Deep learning with Python: learn best practices of deep learning models with PyTorch, Berkeley, CA: Apress, pp. 197-242. 10.1007/978 1 4842 5364 9_6.

Khudeyer, R.S. and Almoosawi, N.M. (2023) 'Combination of machine learning algorithms and ResNet50 for Arabic handwritten classification', Informatica, 46(9) https://doi.org/10.31449/inf.v46i9.4375.

Kusetogullari, H., Yavariabdi, A., Hall, J. and Lavesson, N. (2021) 'DigitNet: a deep handwritten digit detection and recognition method using a new historical handwritten digit dataset', Big Data Research, 23, p. 100182. https://doi.org/10.1016/j.bdr.2020.100182.

Li, Z., Liu, F., Yang, W., Peng, S. and Zhou, J. (2021) 'A survey of convolutional neural networks: analysis, applications, and prospects', IEEE Transactions on Neural Networks and Learning Systems, 33(12), pp. 6999-7019. 10.1109/TNNLS.2021.3084827.

Meraihi, Y., Gabis, A.B., Ramdane-Cherif, A. and Acheli, D. (2021) 'A comprehensive survey of Crow Search Algorithm and its applications', Artificial Intelligence Review, 54(4), pp. 2669-2716. 10.1007/s10462-020-09911-9.

Mezghani, A., Maalej, R., Elleuch, M. and Kherallah, M. (2023) 'Recent advances of ML and DL approaches for Arabic handwriting recognition: a review', International Journal of Hybrid Intelligent Systems, 19(1-2), pp. 61-78. https://doi.org/10.3233/HIS-230005.

Mohamed, N., Josphineleela, R., Madkar, S.R., Sena, J.V., Alfurhood, B.S. and Pant, B. (2023) 'The smart handwritten digits recognition using machine learning algorithm', in (2023) 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), IEEE, pp. 340-344.

Mohammed, H.A., Kareem, S.W. and Mohammed, A.S. (2022) 'A comparative evaluation of deep learning methods in digital image classification', Kufa Journal of Engineering, 13(4) https://doi.org/10.30572/2018/KJE/130405.

Mutawa, A.M., Allaho, M.Y. and Al-Hajeri, M. (2024) 'Machine learning approach for Arabic handwritten recognition', Applied Sciences, 14(19), p. 9020. https://doi.org/10.3390/app14199020.

Prasad, M.L., Srinivasulu, C., Yashaswini, G. and Rakshitha, K. (2023) 'Decoding digits: advancements in handwritten digit recognition with machine learning algorithms', in (2023) International Conference on Recent Advances in Science and Engineering Technology (ICRASET), IEEE, pp. 1-6. 10.1109/ICRASET59632.2023.10420054.