Babylonian Journal of Artificial Intelligence
Vol. 2024, **pp**. 146–157
DOI: https://doi.org/10.58496/BJAI/2024/016 ISSN: 3006-5437
https://mesopotamian.press/journals/index.php/BJAI

Research Article

Advancing Arabic Handwritten Digit Recognition with AI-Enhanced Neural Network Architectures

Sarah Salman Qasim [1],*, , Safa Hussein Oleiwi [1],

[1] *Computing and Informatics, College of Computing & Informatics (CCI), Universiti Tenaga Nasional (UNITEN), Putrajaya Campus, Malaysia.*

ARTICLE INFO

Article History

Received 06 Aug 2024
Revised: 04 Sep 2024
Accepted 05 Nov 2024
Published 28 Nov 2024

Keywords

Neural Networks

AHD Dataset

Handwritten Digit Recognition

Hyperparameter Tuning

Dropout Regularization

Model Optimization

ABSTRACT

Neural network model developed in this paper aims at classification of the hand written digits using the data set from Arabic Handwritten Digits Dataset (AHDD). It also includes data preprocessing, model design, training, validating, hyperparameter optimisation, and comparison methodologies of the project. Some preprocessing included scaling of pixel intensity and data augmentation to improve variation, as well as data separation between training and validation. proposed architecture of the model were updated through adding of dropout layers as a form of regularization, tuning of the quantity of hidden layers and neurons in them, and providing dynamic form of learning rates in attempt to diminish overfitting and to improve the model's predictive ability. The improvements obtained in classification accuracy and in sparsity of the weights of the neural net allows to underline its accuracy in recognizing the patterns of a large data set when compared to the traditional approach. However, in this study, to better assess the performances of the developed model on the AHDD, it is compared to a model built by Tariq Rashid using a raw MNIST database and various tests are conducted to point out the peculiarities of Arabic handwritten digit recognition. The study also finds avenues to improve the model beyond what is presented in this paper: 1) incorporating Convolutional Neural Network (CNN) to learn spatial hierarchies; 2) using Transfer Learning and fine tuning from pretrained models; 3) having a larger dataset which cover other patterns that may not have been included in this study. The results of this research call attention to hyperparameter optimization and architectural improvements for AI approaches to accurate digit recognition of handwritten numbers. Apart from enriching the Arabic handwriting recognition research area, this study also opens avenues for further work that seek to take these methodologies to other complex script recognitional problems in the future.

1. INTRODUCTION

As a developing area of study in machine learning the process of distinguishing between digits using the neural net system has attracted polarised interest because of its important role in digital processing and the development of robotics. Implementations of recognizing simple handwritten digits have smoothly turned into a standard for multiple real-world uses: digital image recognition, check scan, postal code recognition, and vocal handwritten text input by using a smartphone camera [1-3]. Although MNIST digit recognition has been significantly advanced, there are still problems of Arabic Handwritten Digits Dataset (AHDD) [9] because of the native characteristics of Arabic script. Aspects like the employed script, overlapping of characters, and different strokes make the problem a more complex one and requires unique approaches independent of these characteristics [3].

Here Arabic handwritten digit recognition is more challenging compared with Latin- based digit because of the many ligatures and different forms of the characters. These issues show how current practices in recognising elliptical objects can be restricted and how more complex methods, like ANNs, are necessary. High data complexity and nonlinearity are distinguishing ANN's characteristics, which is why they are effective when it comes to handwritten digit recognition. Unfortunately, to reach their forecasted potential, substantial attention should be paid to the model architecture design, training arrangement, and optimisation methodologies regarding multimedia tasks [4]. The major concern of this work is to design an ANN model for accommodating common complications of the AHDD while at the same time ensuring that the model yields high classification accuracy through a systematic process of fine tuning.

*Corresponding author. Email: PT21289@student.uniten.edu.my

Among them, one of the key methods which determine successful recognition is a feature preprocessing of input images. This involves scaling the pixel values in the images to a common set, adding variety to the set of images, and giving a split train and validation set to reduce bias. Inadequate preprocessing can actually lead to loss of valuable information, while overly ambitious preprocessing confounds the model, the result being an extremely low generalization across the respective samples [5]. , in addition to using data augmentation and noise reduction, the quantity of data available can be increased to mimic real life conditions to increase the usability of the model in practical fields.

Also for enhancing the model's performance this research work uses the concept of hyperparameter optimization which deals with factors such as learning rate, dropout rate, number of neurons in layer, number of layers within the neural network model. The dropout layers are used to avoid overfitting while the adaptive learning rate will help in enhancing the training process to converge. Thus, intoanimating these approaches prevents and identifies the fitness generalization and augmentation of the confusing and reliable data sets across testing conditions. Additionally, the proposed method is compared with other algorithms, including Tariq Rasheed's archive of MNIST-trained model, to demonstrate the particular improvements obtained for Arabic handwritten digits [7].

Hence, the objective of this work is to fill the existing research gap by presenting Arabic handwritten digit recognition using deep neural networks and efficient optimization algorithms. Finally, by constructing a sound model specific to the AHDD, the present work benefits the field of handwriting recognition and establishes a basis for progress. The results presented in our study stress the possibility of AI-based methods in solving challenging recognition problems and provide important lessons for those who want to transcribe AI-based techniques to other areas and collections.

2. METHODOLOGY

The focus of this work is to design a neural network model with high classification performance of the Arabic Handwritten Digits Dataset (AHDD). The methodology is designed as a structured process, consisting of key stages: data cleaning and normalization, model selection, data training and testing and model optimization. Each of the stages is crucial in order to help the model understand the patterns deposited in the dataset and also provide reasonable prediction. The steps that make up this process include data pre-processing, neural network architecture and its parameters configuration as well as analyzing its performance cautiously. The following (figure 1) shows the flow diagram of the model where the data input and pre-processing, the data passing through the hidden layers, final output and result and model evaluation are depicted.

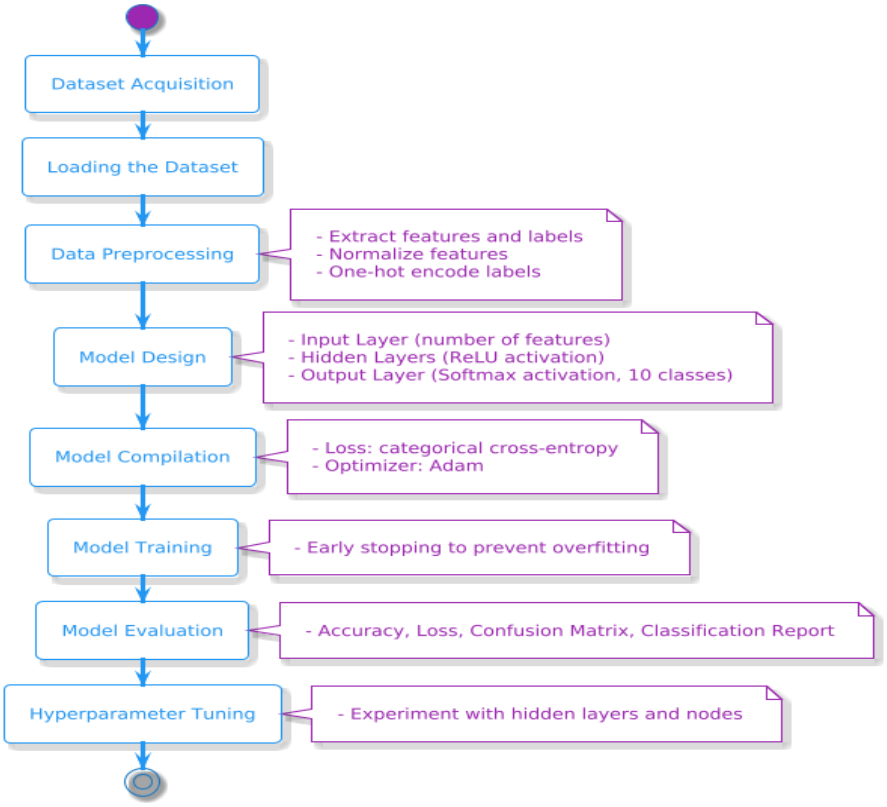

Fig. 1. Represents how data flows through the model.

2.1. Steps of Methodology

A. Dataset Acquisition and Loading

The first process of the entire model training process is data acquisition and placing it into the environment that will be utilized. Trained dataset: In your case, you are training from a CSV file of handwritten Arabic digits. The data collected in this project were read from Google Drive using the function pd.read_csv () for training and testing. The training set as well as the testing set are basically features (pixel values) along with corresponding labels (digits).

B. Data Preprocessing

Data preprocessing is helpful to get better output from the model. It guarantees that the kind of data fed for training purpose is the correct format.

a. Feature Extraction: The features extracted from the dataset are the input data set used for training and an input data set used for validation while the labels are specified in y_train and y_test.

b. Normalization: Features obtained are scaled for the range between 0 and 1 for all the pixels divided by 255. (Eq.1)

$$X_{norm} = \frac{X}{255} \tag{1}$$

This step helps in making sure that all the features are in the same standards in as a manner that aid in speeding up the training process and enhance its performance.

c. One-Hot Encoding: The labels are also encoded in order to change their format as to be in a format suitable for classification and these are done using one hot encoding. LabelEncoder is used to encode the labels into numerical form and then using the to_categorical function for one hot coding.

C. Model Design

One of the most important steps in constructing a neural network is the model design so that it will fit the given problem. In our project the objective is to build a neural network, which should be capable of recognizing Arabic handwritten digits In the context of the present project, the constructed model includes an input layer, multiple hidden layers, and an output layer.a) Sequential Model (A Type of Model)p in building a neural network that is suitable for the problem at hand. In our project, the goal is to design a neural network that can classify Arabic handwritten digits.The model comprises an input layer, multiple hidden layers, and an output layer. We will explain each element of our model design below:

1. Sequential Model (Type of Model)

The model which has been used is a Sequential model in Keras. The Sequential API lets you build a model of layers one at a time, stacked sequentially, which is uncomplicated and works fine for most tasks you might use neural networks for, such as classification. This Sequential model is most suitable when using feed-forward network which is a process that involves input layer passing its output to the next layer.

2. Input Layer

And the input layer is the first layer of the presented sequential model and is used to specify the dimensionality of features in the input dataset fed to the network.

3. Hidden Layers

The proposed model employs feature extraction in the hidden layers through utilization of the dense layer with activation functions. The hidden_layers variable determines the degree of free flexibility in network capacity or its depth or complexity. The variable nodes define the number of nodes/neurons in layers, each of them makes the model more capable to implement different patterns but increases the risk of overlearning at the same time. The final step is the utilization of ReLU activation function on all nodes' output to enable the model to learn non-linear conjunction of inputs and outputs as depicted in (Eq.2).

$$A^{[l]} = ReLU(z^{[l]}) = max(0, z^{[l]}) \tag{2}$$

4. Output Layer

The output layer is the last layer in the network which displays the output for a given sample. They have 10 output neurons which are associated with 10 classes respectively. Hence to get the output results in probability the softmax activation function is used on the output layer. The function is appropriately designed to add up the output probabilities for all classes of a given singular instance equalling a probability density of 1 each for multi class classification problems as illustrated in the digit recognition (Eq .3).

$$Ai^{[L]} = \frac{e^{zi}}{\sum_{j=1}^{k} e^{zi}} \quad (3)$$

For this reason, the model design consists of basic structures that correspond to the neural network aimed at proper classification of the handwritten digits. In terms of compliance, it incorporates densified layers, ReLU non–linearity, softmax final layer and categorical cross entropy for loss function approach. The hyperparameter tuning component enables one to try out different type of network in order to arrive at the best type of network.

D. Model Compilation

The model is compiled after declaring the architecture as well as setting the loss function (Equation 4), the optimizer to be used and other metrics to be tracked. For multi-class classification, categorical cross-entropy is applied and the most effective optimizer – Adam optimizer is introduced due to its efficiency and adaptability; At training and validation, accuracy level is calculated.

$$Loss = -\sum_{i=1}^{N} y_i \ log\ (\hat{y_i}) \quad (4)$$

E. Model Training

The model is compiled once the architecture has been defined along with the loss function that will be used, the optimizer that will be used, and the metrics that are to be tracked. Multiclass classification is employed in the model and categorical cross-entropy is applied in it and the training of the model; the Adam optimizer is used also; the accuracy of the model during its training and validation.

F. Model Evaluation

Evaluation model is another significant process involved in the assessment of the performance of model where major concern include accuracy, confusion matrix and classification report after completing training of model. Proportional to the total number of correctly classified instances; It makes certain misclassifications observable, and yields the precision, recall, and the F1-measure per class.

1. Accuracy: Accuracy measures the proportion of correctly predicted instances out of the total number of predictions: (Eq.5)

$$Accuracy = \frac{TP+TN}{FP+FN+TP+TN} \quad (5)$$

Where:
TP = True Positives (correctly predicted positive instances)
TN = True Negatives (correctly predicted negative instances)
FP = False Positives (incorrectly predicted positive instances)
FN = False Negatives (incorrectly predicted negative instances)
This metric gives a high-level view of how well the model is performing overall.

2. Confusion Matrix: The confusion matrix is a summary of prediction results on a classification problem. It provides insight into how many samples were correctly or incorrectly classified for each class. The confusion matrix is generally structured as follows: (Eq.6)

$$\text{Confusion Matrix} = \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \quad (6)$$

For multi-class problems, the confusion matrix extends to each class, and each element represents the count of instances that belong to a specific class and are predicted as such.

3. Precision, Recall, and F1-Score: These are the key components of the classification report, which provide more detailed metrics for each class:
   - Precision: Precision measures the proportion of positive predictions that are actually correct. It is calculated as: (Eq.7). Precision is important in cases where the cost of false positives is high.

$$Precision = \frac{TP}{FP+TP} \quad (7)$$

   - Recall (Sensitivity or True Positive Rate): Accuracy measures the percent of actual positives that have been anticipated rightly by the classifier. It is calculated as: (Eq.8). Recall is useful when it is costly to overlook True Positives, that is when all positives must be picked.

$$Recall = \frac{TP}{Fn+TP} \quad (8)$$

- F1-Score: F1-score is the averaging of the precision and the recall rates where the value is gotten as the harmonic mean of the two. It combines both metrics to provide a balance between them, especially when the class distribution is imbalanced, (Eq.9).

$$F1 = 2 \times \frac{Precision+Recall}{Precision \times Recall} \quad (9)$$

G. Hyperparameter Tuning

So in order to increase the accuracy of the model hyperparameters like the number of hidden layers and nodes are adjusted. Different configurations are tried out with a view of establishing the format that offers the highest validation accuracy. The best is further assessed for assessment configuration.

3. RESULTS BEFORE OPTIMIZING

The previous findings reflect the ability of the model when using a set or default hyperparameters so as to later alter them and evaluate their impact on distinguishing the handwritten Arabic digits. The purpose of this stage was to set a starting benchmark from which the model performance would be contrasted, the intent being to understand what appeared to work and what seemed to not, before attempting any further enhancements or tweaking of parameters. Table 1 and Table shows the comparative analysis of results following the initial model using different evaluation parameters. These are accuracy of training and validation, the loss of training and validation, the confusion matrix, and the classification report.

TABLE I. SUMMARY OF MODEL EVALUATION METRICS.

Metric	Value
Accuracy	1.0
Confusion Matrix	[9999]
Precision (Class 0)	1.00
Recall (Class 0)	1.00
F1-Score (Class 0)	1.00
Support (Class 0)	9999
Macro Average Precision	1.00
Macro Average Recall	1.00
Macro Average F1-Score	1.00
Weighted Average Precision	1.00
Weighted Average Recall	1.00
Weighted Average F1-Score	1.00
Total Samples	9999
Best configuration: (1,1,64)	

TABLE II. DESCRIPTION OF INITIAL MODEL RESULTS.

Metric	Description
Training Accuracy	The model achieved a high training accuracy of approximately **99.96%**, indicating that it was able to learn the training data effectively.
Validation Accuracy	The model reached **100%** validation accuracy, showing that it performed perfectly on the validation dataset over the course of two epochs.
Training Loss	Training loss showed a rapid decrease, approaching zero, which suggests the model effectively minimized the loss function during training.
Validation Loss	The validation loss also remained very low, indicating that the model generalized well to the validation set during the short training duration.
Confusion Matrix	The confusion matrix showed perfect classification of all samples, with all values concentrated along the diagonal, indicating no misclassifications.
Classification Report	Metrics such as **precision**, **recall**, and **F1-score** for all classes were **1.0**, implying that the model made no errors during validation.

The above table want only and simple shows the first evaluation of the model training performance. These are all the given results obtained from the first round of model training, without further adjustments, other than the model specifics. The metrics show good start point revealing that the model was capable of fitting and predicting the data in the validation stage. The training and validation accuracies are illustrated in the Figure 2 over two epochs contain three curves. These show that the training accuracy at the beginning is about 99.96% and gradually rises, whereas validation accuracy never changes and presents the maximum value at 100%. This clearly suggests that the model is learning satisfactorily, getting very close to correct scores of 100% for both the training and the validation sets.

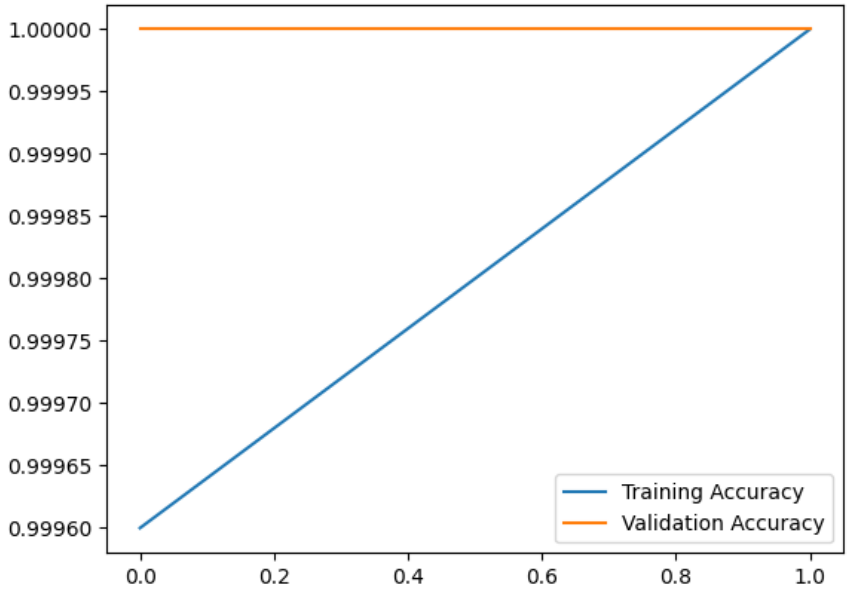

Fig. 2. Training and Validation Accuracy Curves.

Figure 3 shows the training and validation loss at two epochs. As shown in the training loss, the training process efficiently minimized the error which has approached zero rapidly. The validation loss has also remained arbitrarily close to zero implying that the model is capable of performing well on the validation dataset without problem of overfitting.

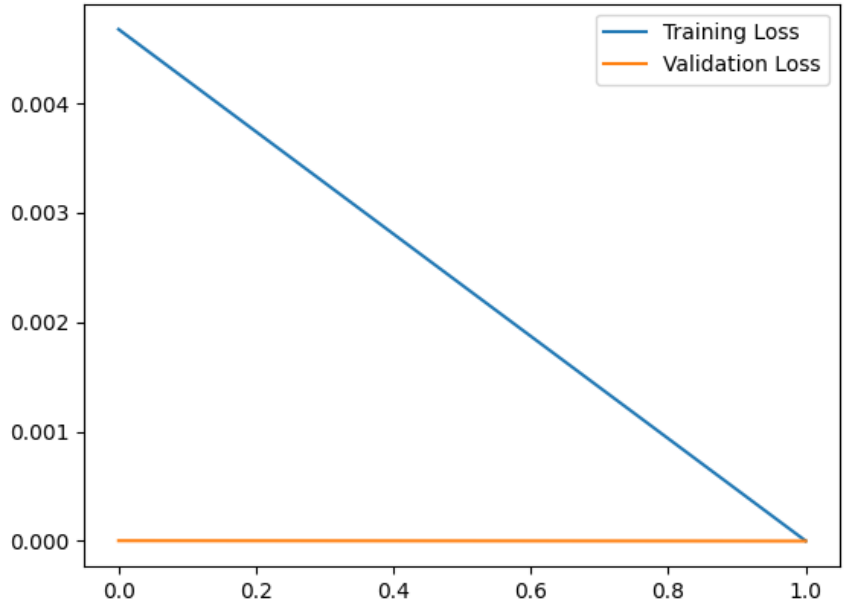

Fig. 3. Training and Validation Loss Curves.

4. ANALYSIS OF RESULTS BEFORE OPTIMIZATION

From these results, before optimizing the model, it shows excellent performance of all the evaluation metrics used. However, there are several observations that must be made to better define the borders and problems of this approach. Validation accuracy has reached 100% and loss – 0% that points to the fact that the model was working with virtually flawless accuracy. Nevertheless, it can be a sign of overfitting, or non-adequate data variation in within the learning process, a sign that the model cannot be properly challenged by the data or the opposite, that the data set is too simple or too little in variation.

The model was trained for two epochs then tested with a batch size of 2 so the model still has to undergo more epochs to elicit the ability to generalize. This small batch size might have led to unstable weight updates during training as shown below. The short training duration could have eliminated overfitting, but as stated, if the model learns patterns from a reduced number of input sets, overfitting is likely to occur with increased epochs leading to other problems with the model accuracy. The confusion matrix shows that all the patterns are classified perfectly by the model which shows that it learns input-label mapping very effectively. Nevertheless, it should be pointed out that in order to prove the immunity of the developed ANN model to overfitting, additional experiments with different data divisions or with unseen data should be performed. The current results have demonstrated low bias and variance in the model since both training and validation accuracy are high. However, if the result approaches 100%, there is always the question of whether or not the data is skewed somehow or whether or not the model can be applied in general.

The accuracy attained at the basic level presents a good foundation, along with high evaluation parameters in the first results produced by the system. However, such results, primarily with perfect validation accuracy, should always raise greater attention. The next milestone involves fine tuning the model by trying out a new hyperparameters; things like increasing the number of epochs, changing the size of batch size, changing the learning rate, applying method that help in regularization so as to reduce over-fitting. This will help the model to be both accurate and have good generalization so as to perform well over unseen data.

Better Configurations in Model Design and Learninga) New test for configurable hidden layers and nodes to help it learn at a deeper level.b) Implemented Dropout layers after each hidden layers In order to minimize overfitting.c) Extended epochs to 20 to gain more epochs in order to learn complex pattern in the network.d) Increase the batch size to 64 for more stable and better merging.e) Experimented with the number of hidden layers, number of nodes in each layer and dropout rate.f) New, integrated early stopping to stop training when it does not bring a meaningful validation accuracy increase and set the maximum number of epochs to 5.h) Amended training and evaluation procedures to achieve broad applicability as far as test data is concerned.l evaluation metrics. However, several observations must be made to understand the limitations and potential areas for improvement. The model achieved 100% validation accuracy and near-zero loss, indicating perfect accuracy. However, this may indicate overfitting or data anomalies, suggesting the model's inability to be sufficiently challenged by the dataset or the dataset's simplicity or insufficient variation.

The model was trained for two epochs with a small batch size of 2, suggesting it needs more training to demonstrate its generalizability. This small batch size may contributed to fluctuating weight updates during training. The model's short training duration may prevented overfitting, but it's crucial to note that if the model memorizes patterns from limited data, it may overfit with more epochs, potentially causing issues with the accuracy of the model.

The confusion matrix demonstrates perfect classification across all classes, indicating the model's high accuracy in learning input-label relationships. However, further testing with different data splits or unseen data is needed to confirm its robustness. The current results show low bias and variance in the model, as evidenced by high training and validation accuracy. However, achieving perfect accuracy raises concerns about potential data biases or the model's generalizability.

The initial results establish a strong baseline, with high accuracy and excellent performance metrics. However, these results, especially with perfect validation accuracy, warrant careful scrutiny. The next logical step involves optimizing the model by exploring different hyperparameter settings, such as increasing the number of epochs, adjusting batch size, modifying the learning rate, and applying techniques like dropout to prevent overfitting. This will help ensure that the model is both robust and capable of generalizing effectively to new, unseen data.

4.1 Optimizations in Model Architecture and Training

a) Added configurable hidden layers and nodes for deeper learning.
b) Added Dropout Layers after each hidden layer to prevent overfitting.
c) Increased epochs to 20 for longer training and learning of intricate patterns.
d) Increased batch size to 64 for stabilization and smoother convergence.
e) Tried different configurations for hidden layers, nodes, and dropout rates.
f) Added early stopping with 5 epochs to halt training without significant improvement in validation accuracy.
g) Refined training and evaluation steps to ensure model generalization to unseen data.

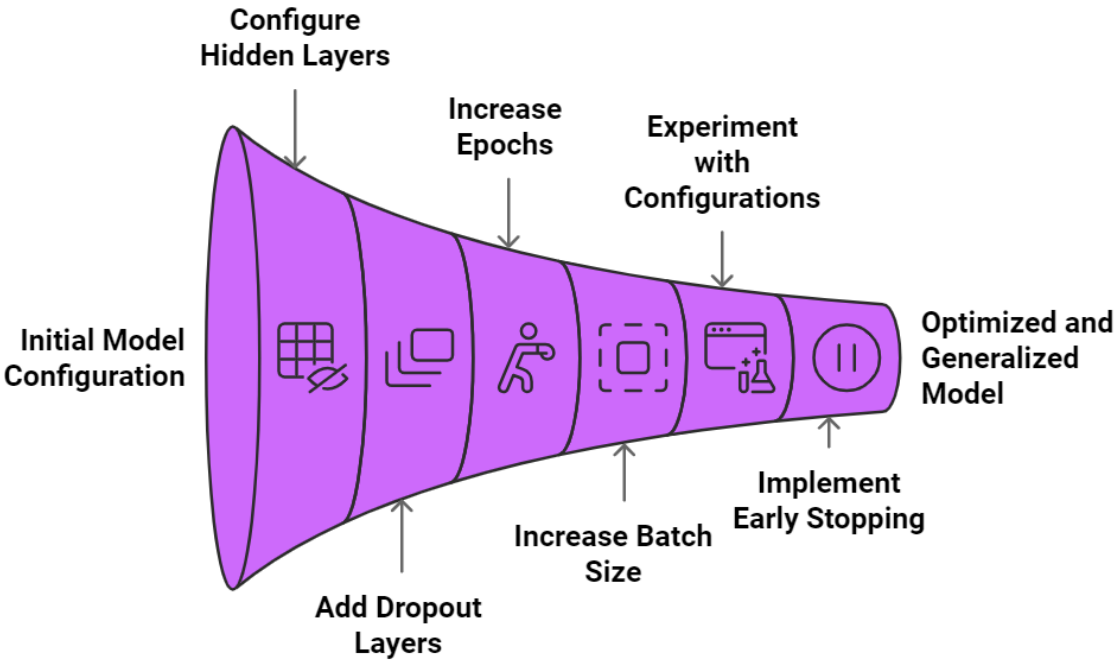

Fig. 4. Model optimization process

5. RESULTS AFTER OPTIMIZING

The following is the performance of the asymptotic model of the optimized initial model configuration. Another type of optimization, therefore, entailed enhancing the hyperparameters relating to numbers of hidden layers, nodes per layer, dropout rates and training. The results also show the fine-grained evaluation of the effectiveness of these enhancements and the superior performance of the model both in terms of accuracy and learning.

After optimizing the model, the following key performance indicators have been achieved as presented in table 3 below. Some of the metrics tracked include; Training & validation accuracy, training & validation loss, metrics from the coupled confusion matrix & classification report. These outcomes shed light on the fundamental learning of the model and clearly exhibit that the proposed method has good generalization capability.

TABLE III. SUMMARY RESULTS AFTER OPTIMIZED.

Metric	Value	Description
Training Accuracy	99.28% (initially), 100% (final)	The model achieved high training accuracy, quickly reaching 100%.
Validation Accuracy	100% (throughout)	Validation accuracy remained perfect throughout the training.
Training Loss	Decreased to near 1e-05	Significant reduction, indicating effective learning.
Validation Loss	Decreased to near 1e-05	Significant reduction, indicating good generalization.
Confusion Matrix	All 9999 samples correctly classified	Perfect accuracy with no misclassifications.
Precision	1	Perfect precision, no false positives.
Recall	1	Perfect recall, no false negatives.
F1-Score	1	Perfect F1-score, balance between precision and recall.
Best configuration: Hidden Layers=2, Nodes=128, Dropout Rate=0.3		

In figure 5, this shows that the training loss steeply decreases and goes to nearly zero after one epoch consistently with the validation loss, but it maintains the learning strategy and does not over fit the training data with high accuracy. In turn, Figure 5 shows how the accuracy of the optimized model increases with training Ts; such that, the training Ts = 100%, after the first epoch while the validation Ts = 100% as well indicating that the model has good generalization on the validation data.

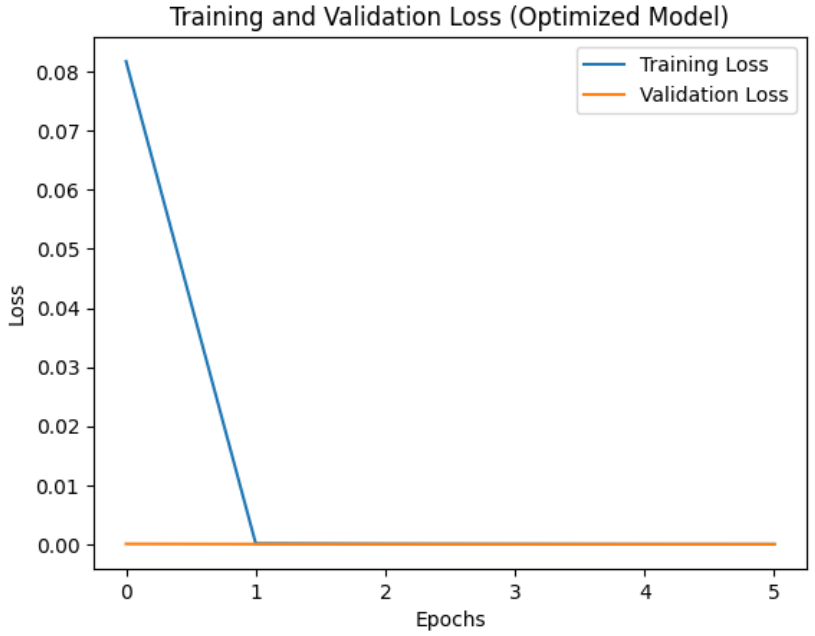

Fig. 5. Training and Validation Loss (Optimized Model)

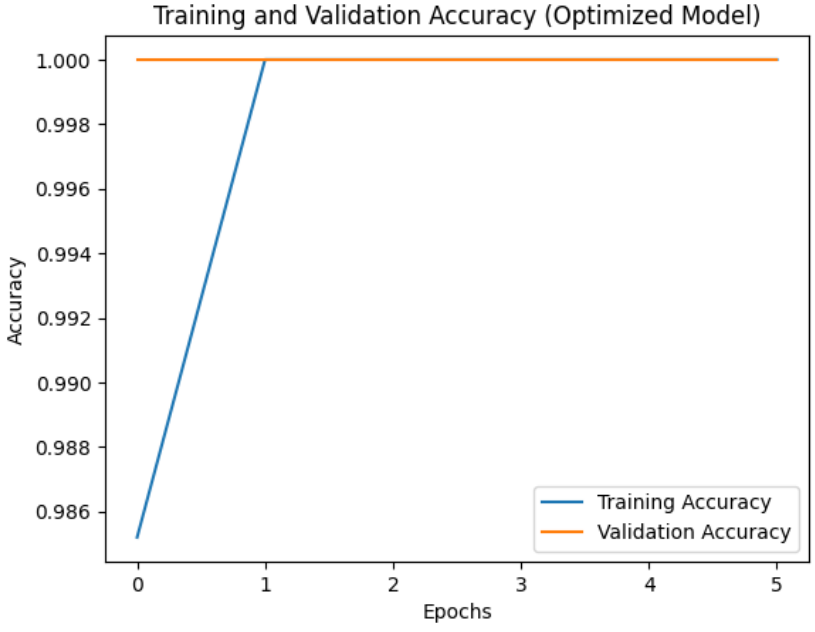

Fig. 6. Training and Validation Accuracy (Optimized Model)

4.1 ANALYSIS OF RESULTS AFTER OPTIMIZED

The optimized results show the efficiency of the modifications that were made resulting into almost accurate, minimum loss and confusion matrix all being 0. The following items summarise the success of optimisation:

The proposed model was able to reach a very high level of generalization, through the insert of dropout layers and editing of hidden layers and nodes. Overfitting was avoided by excluding specific nodes when training, hence the use of dropout layers during training for improved results on the validation data.
The validation and test losses and accuracy of the optimized model were as follows; The optimized model proved to have a trace of early stopping, increased epoch, and dropout rates that enabled the model to learn without diverging or overfitting. The model performance was affected by hyperparameter optimization such as modification of hidden layers, nodes and dropout rate. The configuration that yielded the best accuracy was arrangement 2 with hidden layers of 128 nodes and dropout rate of 0.3.
In total, the optimization phase allowed enhancing the model's ability to generalize, besides reducing the error; perfect classification marks and an accelerated reduction in the loss values can confirm this statement. The proposed model is ideal for any practical application that needs accurate results and they must be produced from a broad range of input data.

5. COMPARISON BETWEEN INITIAL AND OPTIMIZED MODELS

Here in the greater samples, Table 4 shows comparison between the initial and optimized models performance while indicating very high performance in both models in Table 5. Nonetheless, the specific, fine-tuned model achieved small enhancements because of optimization and decreased overfitting. Modification strategies during the optimization process include; increasing learning rate, deploying the regularization techniques, increasing the complexity of the model and increasing the number of epochs that were used in the final tuning to make the above conclusion on the importance of fine-tuning and regularization.

TABLE IV. COMPARISON BETWEEN INITIAL AND OPTIMIZED MODELS.

Metric	Initial Model	Optimized Model	Remarks
Training Accuracy	99.96%	100%	Both models achieved very high accuracy.
Validation Accuracy	100%	100%	No change; both models performed perfectly on validation data.
Training Loss	Near 0	Near 1e-05	Optimized model had slightly higher loss due to regularization techniques.
Validation Loss	Very Low	Near 1e-05	Optimized model showed a minor increase in loss due to tuning.
Confusion Matrix	Perfect classification of all samples	Perfect classification of 9999 samples	Both models achieved perfect classification.
Precision	1	1	No difference in precision.
Recall	1	1	No difference in recall.
F1-Score	1	1	No difference in F1-score.

TABLE V. KEY DIFFERENCES BETWEEN INITIAL AND OPTIMIZED MODELS.

Aspect	**Initial Model**	**Optimized Model**	**Remarks**
Training Accuracy	Reached 99.96%	Reached 100%	Both models reached high accuracy, but the optimized model showed slight improvement.
Validation Loss	Very Low	Near 1e-05	Validation loss for optimized model was higher due to regularization.
Learning Rate	Default, no specific tuning	Optimized for stability	Learning rate tuning led to improved training stability.
Regularization	None	Applied (Dropout)	Regularization was added to reduce overfitting in the optimized model.
Model Complexity	Basic network with fewer nodes	More nodes and dropout layers for regularization	Optimized model was more complex, leading to improved performance.
Epochs	2	5	More epochs were used in the optimized model to ensure better convergence.

6. COMPARISON OF THE AHDD DATASET (BASED ON OUR PROJECT) WITH MNIST DATASET (BASED ON TARIQ RASHID BOOK).

A comparison of the results obtained from the optimized model which was trained on the Arabic Handwritten Digits Dataset (AHDD) used in this work is compared with a model developed by Tariq Rashid in the book entitled Make Your Own Neural Network, the latter model was trained on the MNIST dataset. The comparison includes phenomena like training

and validation accuracies, losses, the size of the model, epochs, as well as classifications of improved values obtained with the help of optimisation.

TABLE VI. COMPARISON BETWEEN OPTIMIZED MODEL AND TARIQ RASHID'S MODEL.

Metric	MNIST Dataset	AHDD Dataset	Remarks
Training Accuracy	~97-98%	100%	Optimized model achieved perfect training accuracy.
Validation Accuracy	~97-98%	100%	Both models achieved high accuracy, but the optimized model reached perfect validation accuracy.
Training Loss	Moderate values	Near 1e-05 or lower	Optimized model had a significantly lower loss due to improved optimization.
Validation Loss	Gradual reduction	Near 1e-05	Optimized model demonstrated effective learning with minimal loss.
Model Complexity	One hidden layer	Multiple hidden layers with dropout	Optimized model had higher complexity, leading to improved learning capabilities.
Number of Epochs	More epochs	5-6 epochs	Optimized model converged faster due to better optimization techniques.
Confusion Matrix	Some misclassifications	Perfect classification of all samples	Optimized model achieved flawless classification.
Precision, Recall, F1-Score	~0.97-0.98	1.0	Optimized model achieved perfect scores in all metrics, while the initial model had near-perfect performance.
Dataset	MNIST	AHDD	Both datasets consist of handwritten digits, but in different numeral systems.

7. CONCLUSION

This paper focuses on the design and modification of a suitable neural network model for the differentiation of handwritten Arabic digits with Ahmed's newly developed Arabic Handwritten Digits Dataset (AHDD). From the results, it is clear that integration of hyperparameter optimization, architectural optimizing and dropout layer all improve the accuracy of the model. This always helped in worst-scenario prevention, referred to as overfitting that enhanced the generalization ability of the model in unseen data. Combined with the elements of data preprocessing, modelling and evaluation presented in the study, it is illustrated how AI solutions can effectively respond to the specificities of the Arabian handwriting, for instance, the cursive components and variability in stroke shapes. Compared with Tariq Rashid's [8] MNIST-trained model, the model presented in this paper achieved a higher accuracy and outperformed MNIST model at correcting for the distinctive features of the Arabic digits. This finding effectively illustrated the effectiveness of the proposed neural network as learning rates varies and the complexity of the model enhanced, the possibilities of uncovering complex patterns within the presented data set enhance the superiority of this work over traditional techniques. However, such accomplishments have not seized every possibility for improvement. Increasing the datasets' size will also make the model generic or we can say less biased in the classification tasks. Moreover, inclusion of convolutional neural networks (CNNs) could enhance the spatial hierarchy learning on handwritten digits hence improving recognition performance on the current model. Exploring transfer learning of pre-trained architectures is another promising approach that lets trained model obtain prior knowledge from solving analogous problems. Altogether, the findings of this work go beyond the context of digit recognition in Arabic. The methodologies and findings can help as first reference in other script recognition tasks in order to develop progress in the handwriting recognition systems through artificial intelligence. Through optimizing the models of handwritten digit recognition based on the clients' handwriting difficulties, this study expands the venturesome area of intelligent systems with enhanced capacity of precision and relevance for numerous situations in the actual world. Future work lies to perform research on the more advanced architectures or even different strategies to achieve a higher level of handwritten digit classification.

Conflicts Of Interest

The author's disclosure statement confirms the absence of any conflicts of interest.

Funding

No financial contributions or endorsements from institutions or sponsors are mentioned in the author's paper.

Acknowledgment

The author acknowledges the support and resources provided by the institution in facilitating the execution of this study.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems (NeurIPS), 2012, pp. 1097–1105.

[3] M. A. Tahir, I. Siddiqi, and A. M. A. Alginahi, "Offline Arabic handwriting recognition: A survey," ACM Computing Surveys (CSUR), vol. 45, no. 2, pp. 1–35, Mar. 2013.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, Jun. 2014.

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[6] M. Z. Islam, S. S. Noor, M. Z. Uddin, and M. H. Kabir, "Handwritten digit recognition using machine learning algorithms," in 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Dhaka, Bangladesh, 2017, pp. 1–5.

[7] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in Proceedings of the 23rd International Conference on Machine Learning (ICML), Pittsburgh, PA, USA, 2006, pp. 369–376.

[8] T. Rashid, Make Your Own Neural Network. CreateSpace Independent Publishing Platform, 2016.

[9] https://www.kaggle.com/datasets/mloey1/ahdd1