International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 30, No. 5, May, 2025, pp. 1201-1207
Journal homepage: http://iieta.org/journals/isi

Hyperparameter Optimization for CNN, K-NN, and Decision Tree in Handwritten Digit Classification

Yaqeen Saad Ali[1*], Sura Mahroos Searan[1], Rihab Hazim Qasim[1], Salah Awad Aliesawi[2]

[1] Department of Computer Science, College of Computer Science and Information Technology, University of Anbar, Anbar 31001, Iraq
[2] Department of Computer Networks Systems, University of Anbar, Anbar 31001, Iraq

Corresponding Author Email: yaqeen.cs91@uoanbar.edu.iq

https://doi.org/10.18280/isi.3005208

Received: 7 March 2025
Revised: 20 May 2025
Accepted: 27 May 2025
Available online: 31 May 2025

Keywords:
handwritten recognition, MNIST data set, machine learning algorithms, hyper-parameters tuning

ABSTRACT

Handwritten recognition of characters appears to be the most fascinating field of image processing research among the many studies that have been completed. Handwritten character recognition methods use scanned photos, documents, and real-time devices such as tablets, tabloids etc. as input, which is converted into digital text. For machine learning algorithms, hyper-parameters are important since they guide the training process and have a significant impact on model performance. This study thoroughly examines and emphasizes how careful parameter adjustment is necessary to optimize model performance and generalization. Our findings provide valuable insights to obtain high classification accuracy for handwritten digits using machine-learning techniques which are decision tree (DT), convolution neural network (CNN), and k-nearest neighbors (KNN) that are optimized through hyper-parameter tuning techniques: grid search and random search to modifying a machine learning model's to determine the best values of the parameters to enhance the model's. The optimization machine learning models were applied and compared on the MNIST digit database. The suggested techniques were able to identify optimal hyper-parameters for a variety of ML models. Our major goal is to match the accuracy of the classifier models along with their implementation time to obtain the best possible model for digit recognition. the outcome of our work indicate an accuracy rate of 97.3% for k-nearest neighbors tuning by grid search and 97.03% for k-nearest neighbors tuning by random search while the test accuracy of CNN based on grid search is 99.18% and for random search test Accuracy is 99.08 %. Finally, the test accuracy for decision trees based on grid search is 87.94% and for random search is 88.26%.

1. INTRODUCTION

Handwritten-digit classification is a major subject in computer-vision. Alot of studies excuted in this domain to beat some challenges for the reason that handwritten digits are different in size, thickness and orientation. Typically, a machine learning algorithm transforms a problem into an optimization issue and employs various optimization mechanism to resolve it and sort them into 10 predefined categories from (zero to nine). currently, a large number of person employ divers forms of media-daily, not just for socializing, but also for expressing their thoughts, providing comments, and sharing their experiences [1]. The issue exists even when the language variable remains constant and handwriting is classified according to similar languages. Every written composition is distinct; however, specific similarities can be categorized. In this case, the data scientist's objective was to categorize the image utilized in writing a certain language into groups that reflect similar numbers or characters [2]. Handwriting digit recognition turns numbers that are written by hand into computer data. This method can be used for many things, like entering data, optical character recognition, and machine learning. There are a lot of different ways that people write numbers, which makes HWR hard. This difference could be because of:

- The writer's individual handwriting style
- The type of writing instrument used
- The quality of the paper
- The lighting conditions

Despite these challenges, HWR is a valuable technique that can be used to automate a variety of tasks. One of the key areas of optical character recognition is handwritten digit recognition, which is a sub-problem of OCR [3, 4]. The Modified National Institute of Standards and Technology (MNIST) handwritten digit database is one of the most important areas of research in pattern recognition, has excellent research and practical value. Generally speaking, handwriting classification techniques can be divided into either statistical feature-related methods or structural feature-related approaches [5]. The difficulty originates from the machine's ability to efficiently process written inputs. Hence, it is imperative to harness the potential of machine learning through the creation of intelligent algorithms that can effectively analyze data and make well-informed choices. Image classification algorithms are utilized to examine an input image and forecast its content. Machine learning algorithms have shown remarkable efficacy in accurately predicting handwritten digits and producing suitable results [6]. A machine learning algorithm converts an issue that requires a solution into an optimized problem and employs various optimization techniques to resolve it. The optimization function affects how the ML algorithm adapts the model to the data and is made up of a number of hyper-parameters that are preset prior to the learning process. Internal model parameters, like a neural network's weights, which can be learning from data during the model training phase, are different from hyperparameters [7]. Handwritten character recognition is a wide-ranging area of study that includes different clearly defined methods of application. These approaches include utilizing prominent learning datasets, popular algorithms, and methods for feature scaling and feature extraction. One such widely used dataset is the MNIST dataset, which is a subset of the NIST (National Institute of Standards and Technology) database. The MNIST dataset is created by combining two specific databases from NIST [8, 9]. This paper was arranged as follows: Section 2 involves some previous work. Section 3 involves a view for techniques of hyper-parameter tuning that was applied in order to tune the algorithm parameter. Section 4 provides an overview of the algorithm survey, which are includes the K-Nearest Neighbor Algorithm (KNN), Convolution Neural Network (CNN), and decision tree. Section 5 involves the implementation in detail. Section 6 involves the conclusions.

2. RELATED WORK

Neural Network (NN) techniques are the main topic of this research [10]. The three most often used neural network (NN) techniques are deep neural networks (DNN), convolutional neural networks (CNN), and deep belief networks (DBN). The three NN way are compared and evaluated in terms of many factors near as treat and performance. Random and standard dataset of handwritten number have been used for run the test. The findings indicate that DNN is the most accurate algorithm among the three NN techniques, with a delicacy rate of 98.08. Nevertheless, DNN's prosecution time is comparable to that of the other two algorithms.

The connection between hyperparameters of ML models and their performance was determined using Gaussian processes [7]. Bayesian optimization is used for solving the hyper-parameter tuning problem, which may be represented as an optimization problem. the Bayesian optimization algorithm based on Gaussian process can get great accuracy in a few samples.

A novel multi-objective optimization framework was proposed to identify the most informative local regions from character images [11]. The framework was evaluated on isolated handwritten English numerals (MNIST images). The attributes take off from a convolutional neural network in their model and achieved maximum of 98.92% accuracy, 21.99% attribute lowering and 47.838% reduction/character recognition time are observed. this task performed on four datasets: isolated handwritten Bangla Basic characters, isolated handwritten Bangla numerals, English numerals, and isolated handwritten Devanagari characters

A neural network-based architecture based on adjusting hyperparameter values for handwritten digit recognition was investigated [2]. Numerous models based on neural networks are used to examine various facets of the same, mainly accuracy determined by hyperparameter values. The most precise and effective solution models are offered. The experimental results of the proposed methodexhibit perfect classification on three datasets namely 2-classleukemia, Ovarian, and SRBCT by scoring 100% and 0.97 test accuracy is achieved on two datasets namely 3-class Leukemia and MLL.

Offline (HDR) is a famous issue that remains at best partly resolved. The HDR procedure was implemented using diverse algorithms to address these issues [12]. These algorithms are: random forest (RF), multilayer perceptrons, decision tree. The working logic of the handwriting image classification process was tested, and the efficiency of different algorithms was measured on the same database. The top accuracy at 96% with sensible runtime execution in Random Forest algorithm.

In order to obtain improvement recognition rate of the MNIST dataset, an enhancement deep CNN model with a fast-converging rate in training was applied [5]. The designed model comes with a multi-layer deep arrange structure, including 3 convolution and activation layers for (feature extraction) and 2 fully connected layers for recognition. The hyper-parameters model's, such as the batch sizes, batch normalization, kernel sizes, activation function, and learning rate are optimized to improve the recognition performance. The average classification accuracy of the suggested methodology is found to reach 99.82% on the training dataset and 99.40% on the testing dataset.

Table 1. Comparison of previous strategies

Models	Applied Work	References
DNNs convolutional neural networks and deep belief networks	Neural Network (NN) techniques are the main topic of this work	[10]
Bayesian optimization	established the connection among the hyperparameters of the ML models and their performance using Gaussian processes.	[7]
Convolutional neural network	To determine which local areas of a character image are the most informative, a novel multi-objective optimization method was proposed	[11]
Neural network	In order to test neural network-based architectures for handwritten image classification, hyperparameter values are altered.	[13]
SVM, RF, DT, KNN, ANN, and K-Means	The effectiveness of several algorithms on the same database was evaluated and the working logic of the handwriting digit recognition process was investigated.	[12]
Deep CNN model	a deep CNN model is developed to enhance the recognition rate of MNIST handwritten digit dataset with a fast-converging rate in training	[5]

Table 1 provides a summary of the literature review on machine-learning algorithms for handwritten data classification and testing. To complement our work, this section provides a brief overview of the field's activities.

3. HYPER-PARAMETER OPTIMIZATION TECHNIQUES

Hyperparameter tuning can be conceptualized statistically as an investigation of the hyperparameter space to determine which values result in the best model performance. The set of all potential values for the hyperparameters is known as the hyperparameter space. Hyperparameter tuning aims to identify the values that optimize the model's performance on a validation set. The model's performance is assessed using a validation set of data, which is separate from the training set. The validation set is used to identify the ideal hyperparameter values but is not included in the model training procedure. Cross-validation is a technique used to test a model on a different validation set in order to determine its efficacy. The process of cross-validation involves dividing the data into several folds. The model is then evaluated on the remaining data segments after being trained on a subset of the data. There are a number of different hyperparameter tuning techniques that can be used, some of the most common techniques include [1, 2]:

Grid-search: This approach explores every conceivable hyper-parameter value combinations.

The cross validation technique are used on the training set to quantify performance, which should serve as a guide for training the machine-learning algorithm for every possible combination of hyper-parameters. This validation technique ensures that the trained model obtains most of the patterns from the dataset. The simplest way to adjust hyper-parameters is to use Grid Search

Random-search: This approach selects hyperparameter values from the hyper-parameter space at random.

Random search, further, examine sets from a given probability division and samples the search space. Briefly, it is a method that finds the optimal solution for the model in question by utilizing random combinations of hyper-parameters.

Bayesian optimization: This approach uses a Bayesian model to identify the ideal hyper-parameter values. The amount of hyper-parameters, the computational resources available, and the required level of accuracy all influence the choice of hyper-parameter tuning technique. Optimizing the hyper-parameters of a machine learning model is necessary to maximize its performance. It is crucial to modify the default values in order to attain the required degree of accuracy. Additionally, it is essential to break the data into three sets: the validation, testing, and training sets.

4. OVERVIEW OF ALGORITHMS SURVEY

4.1 KNN algorithm

KNN algorithm is an effective supervised learning method for classification and regression applications. It works by first determining which k data points are most similar to a new data point, and then using the labels of those data points, it predicts the new data point's label (Figure 1) [9, 14].

The *k* parameter in the KNN algorithm indicates how many neighbors to take into account. When forming a prediction, more data points will be taken into account the greater the value of *k*.

A number of distance metrics, including the Manhattan distance and the Euclidean distance, can be used to calculate how similar two data points are to one another.

The labels of the k most similar data points are utilized to predict the label of the new data point once they have been located.

KNN algorithm, a non-parametric technique for classification and regression in pattern recognition, is employed. The input in both situations is made up of the k training examples that are closest in the feature space.

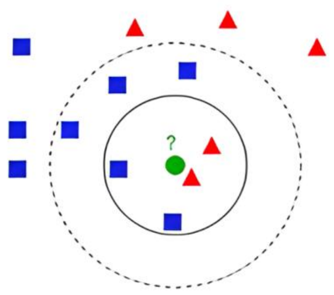

Figure 1. The illustration of KNN

4.2 CNNs

Convolutional Neural networks are popular Deep Learning technique for classifying and recognizing images. (CNNs) have evolved over time to achieve the level of performance they are known for today. The most models of significant deep learning which are: CNNs, which are widely used in face identification, object detection, and image recognition. CNNs work by performing multiple convolutions, which is where their architecture gets its name [15]. CNN extracts features from images by using a number of different layers which are: convolutional, pooling, flattening and fully connected layer (Figure 2).

However, 1 and 2 can be repeated multiple times to help increase learning from the abstract feature.

The first layer in a convolutional neural network is the convolutional layer. It receives the input image and extracts its features. The convolution process is a mathematical procedure that involves a filter sliding across the input image, calculating a dot product between the filter and the image at each. This procedure produces a feature map, which represents the input image by emphasizing specific features, like edges, lines, and forms.

The second layer in a convolutional neural network is pooling layer. It reduces the size of the image while maintaining the most crucial elements by down sampling the convolutional layer's output. The three major kinds of pooling are sum, average, and maximum pooling. Sum pooling takes the sum of the data, average pooling takes the average value, and max pooling takes the maximum value from each region of the convolutional layer's output. The third layer in a convolutional neural network is the flattening layer. It flattens the pooling layer's output into a vector with one dimension. The fully linked layer then receives this vector. The final layer is the fully connected layer. The flattening layer's output is fully combined to this conventional neural network layer [13].

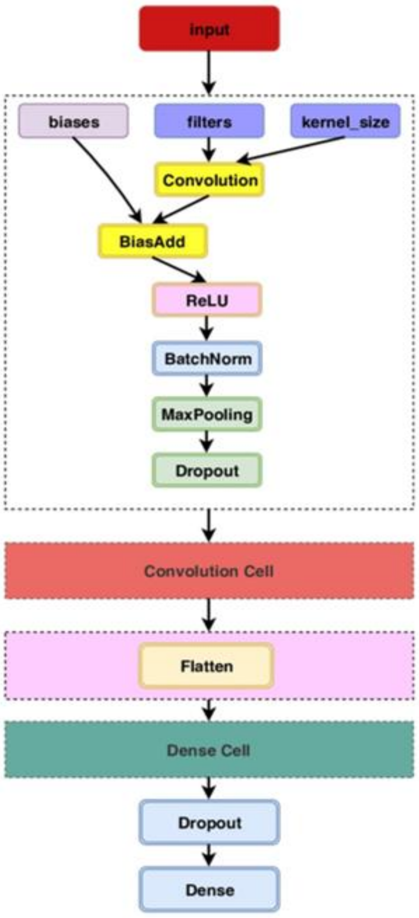

Figure 2. Basic CNN architectures

4.3 DT

The decision tree classifier is a common machine learning technique for addressing classification problems, like handwritten digit recognition (Figure 3). As the name implies, the decision tree model analyzes our facts by formulating conclusions through a sequence of inquiries.

A popular supervised learning technique for classification problems is the decision tree. Both continuous and categorical dependent variables can be used with it. This method separates the population into two or more homogeneous groups. This is done in order to create as many distinct groupings as possible [16, 17].

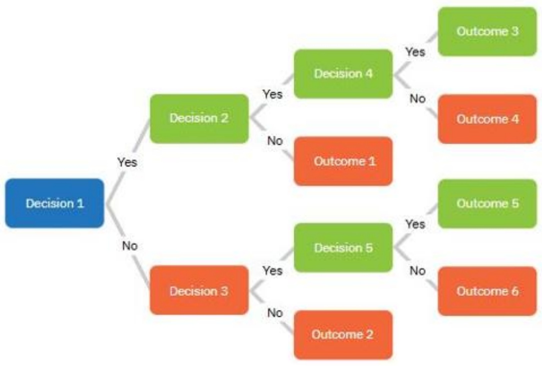

Figure 3. An illustration of DT

5. IMPLEMENTATION

The system was trained and tested using the MNIST dataset. An international dataset of handwritten numbers that is frequently utilized is the MNIST database. This work used the Python package, which provides a wide variety of machine learning techniques. Scikit-Learn's ease of use and quick training and testing of machine learning algorithms may be very helpful to researchers who want to evaluate and assess the efficacy of different machine learning algorithms. Figure 4 shows how our model is implemented. Loading the MNIST dataset is necessary. Both the input features (X) and the corresponding target labels (y) must be present in the dataset. Depending on the machine learning method used, the order in which the actions are performed will change. The MNIST dataset frequently requires pre-processing operations like:

Normalization: The input data (MNIST) contains original pixel-values ranging from (0 - 255). The process of normalizing these numbers involves rescaling them to a more condensed range, usually between 0 and 1. This is done to make machine learning algorithms more effective.

Image rescaling: The MNIST images have 28×28 pixel dimensions, however many machine learning methods require images to have a specific size. The performance of the algorithms may be improved by resizing the (Images) to guarantee that they all have consistent dimensions.

The data is grouped into two sets: train and test set. This is commonly done with the MNIST dataset. The training set is utilized to instruct the machine learning algorithm, whereas the test set is employed to assess the system's performance. The training set should ideally consist of approximately 80% of the data, whereas the test set should ideally consist of approximately 20% of the data

Centering and scaling the images: involves subtracting the mean value from each pixel value in order to achieve proper alignment. By centering the images around 0, the performance of machine learning algorithms can be enhanced.

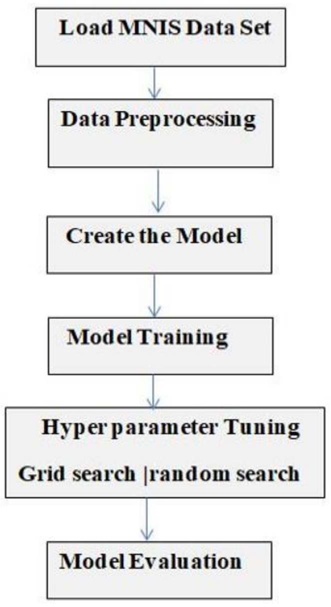

Figure 4. Model implementation overview

In this paper three machine learning algorithm applied to train (MNIST dataset) which are (CNN, DS and K-NN), and perform hyperparameter tuning to optimize the model's performance. This involves searching through different hyperparameter values to find the best combination. Techniques like Grid Search and Random Search are commonly used for this purpose. The Scikit-Learn's Metrics module provides a report that includes the following metrics [18-20]:

1. Precision is the rate at which positive estimates are accurate. It is computed by dividing the total number of false positives (TP + FP) by the number of true positives (TP).
2. Recall: This is the sensitivity or true positive rate (TPR). It is calculated as the number of true positives (TP) divided by the sum of the true positives and false negatives (TP + FN)
3. F1 Score: This is a measure of the harmonic mean of (precision and recall). It is calculated as 2 * (precision * recall) / (precision + recall).

5.1 Input dataset (MNIST)

The input data (MNIST dataset) is a standardized data that is widely used in the pattern recognition community. The MNIST dataset, which consists of 70,000 handwritten digit images, is 28 by 28 (784 pixels) in size. There are 60,000 training images and 10,000 test images. Numerous machine learning models have been trained and tested on this fiercely competitive dataset. Each image is divided by 255 steps to normalize it. Eight bits make up each channel, and when we split them, the result falls between 0 and 1 [21, 22].

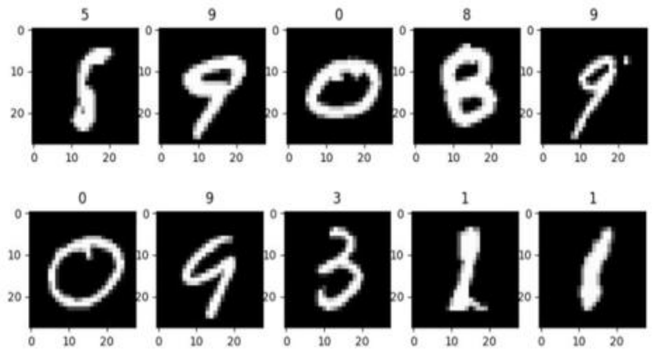

Figure 5. Sample random MNIST handwritten digits

Figure 5 shows an example of how to plot some random MNIST handwritten digits using the Python programming language.

5.2 Results

This experiment aimed to compare classifier methods, namely decision tree (DS), k-nearest neighbors (KNN), and convolutional neural networks (CNN). The experiment was conducted in two stages:

Stage 1: The three classifier techniques were trained and evaluated using grid search. Grid search is a method for finding the best hyperparameter values for a machine learning model by searching a grid of possible values.

Stage 2: The three classifier techniques were trained and evaluated using random search. Random search is a method for finding the best hyperparameter values for a machine learning model by randomly sampling hyperparameter values from a range of possible values.

5.2.1 Hyper parameter tuning for K-nearest neighbors

For grid search best parameters: {gride search: 'n_neighbors': 7, 'p': 2)}, Test Accuracy is: 97.343%, Execution Time: 170 s. Random search best parameters: {Random search: 'n_neighbors': 5, 'p': 2}, Test Accuracy is: 97.031%, Execution Time: 115 s (Table 2).

5.2.2 Hyperparameter tuning for CNN

Based on grid search: Best Parameters: {'dropout_rate': 0.25, 'epochs': 10, 'filters': 64, 'kernel_size': (5, 5)}, Test Accuracy: 99.18%. The best parameters for the model were dropout_rate=0.25, epochs=10, filters=64, and kernel_size=(5, 5) .

Based on random search: Best Parameters:{ conv1_filters: 32, conv2_filters: 32, dense_units: 64, learning_rate: 0.001.

Trial 1 Complete [00h 03m]: validation accuracy: 99.19%, Test Accuracy: 99.08% (Table 3).

5.2.3 Hyperparameter tuning for DT

For grid search test: Accuracy is 87.94%, Execution Time: 280 seconds. Random search test: Accuracy is: 88.26%, Execution Time: 265 seconds (Table 4).

Table 2. Validation set's categorization report for K-nearest neighbors

	Gride Search			Random Search		
Class	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
0	98.68%	99.26%	98.97%	98.84%	99.41%	99.13%
1	98.43%	96.87%	97.65%	97.75%	96.35%	97.05%
2	98.14%	97.65%	97.90%	98.26%	98.33%	98.29%
3	97.91%	97.50%	97.70%	97.92%	97.81%	97.86%
4	99.21%	98.12%	98.66%	98.81%	97.92%	98.36%
5	98.43%	97.81%	98.12%	98.63%	97.50%	98.06%
6	98.95%	98.95%	98.95%	98.92%	98.75%	98.84%
7	98.82%	98.59%	98.71%	99.12%	98.33%	98.73%
8	98.75%	98.43%	98.59%	98.21%	98.44%	98.33%
9	98.68%	98.82%	98.755%	98.82%	98.75%	98.79%
Total	98.29	97.34	97.82	98.29%	97.03%	97.65%

Table 3. Model accuracy and loss evolution across epochs

Epoch	Loss	Accuracy	Val Accuracy
1/5	0.1740	0.9478	0.9835
2/5	0.0567	0.9831	0.9858
3/5	0.0401	0.9876	0.9898
4/5	0.0304	0.9904	0.9905
5/5	0.0227	0.9926	0.9920

Table 4. Validation set's categorization report for DT

	Gride Search			RANDOM SEARCH		
Class	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
0	93.0%	92.0%	93.0%	95.0%	93.0%	94.0%
1	95.0%	97.0%	96.0%	95.0%	97.0%	96.0%
2	86.0%	87.0%	86.0%	84.0%	87.0%	85.0%
3	84.0%	85.0%	84.0%	85.0%	84.0%	85.0%
4	85.0%	87.0%	86.0%	87.0%	88.0%	88.0%
5	84.0%	83.0%	83.0%	84.0%	83.0%	84.0%
6	91.0%	90.0%	91.0%	91.0%	91.0%	91.0%
7	91.0%	90.0%	91.0%	91.0%	91.0%	91.0%
8	84.0%	82.0%	83.0 %	84.0%	82.0%	83.0 %
9	85.0%	85.0%	85.0%	85.0%	84.0%	85.0%
Total	87.8%	87.8%	87.8%	88.1%	88.1%	88.2%

Table 5. Overall comparison between machine learning models

Model Classifier	Test Accuracy (Gride Search)	Test Accuracy (Random Search)	Excution Time for Grid Search	Excution Time for Random Search
CNN	99.18%	99.08%	300 s	180 s
K-NN	97.34%	97.03%	170 s	115 s
decision tree	87.94%	88.26%	280 s	265 s

5.3 Comparison of results and discussion

5.3.1 Accuracy

The comparison between machine learning models is as shown in Table 5, Figures 6 and 7.

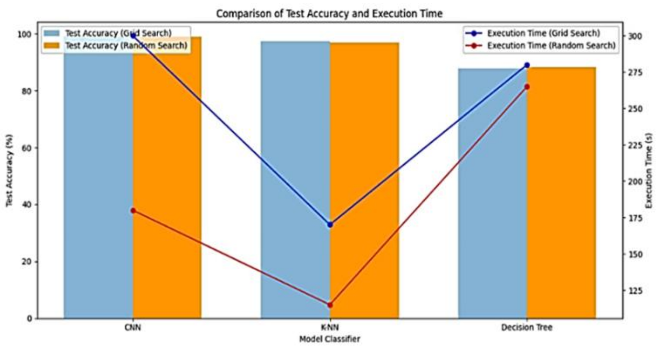

Figure 6. Bar graph depicting accuracy for ML model on both grid and random search

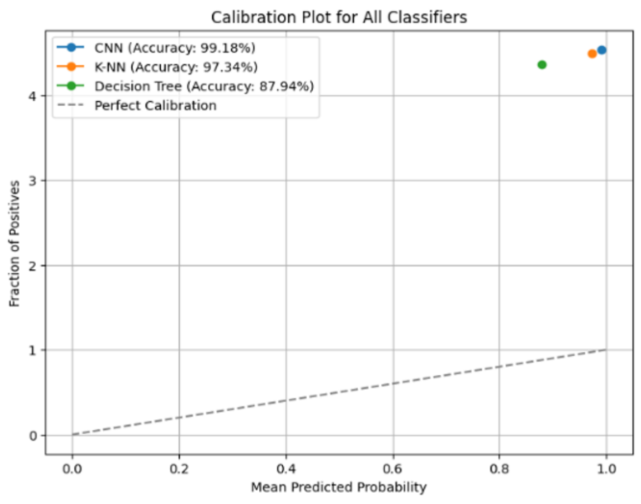

Figure 7. Calibration evaluation of machine learning classifiers

Highest: CNN models achieved the highest test accuracy, with Grid Search reaching 99.18% and Random Search reaching 99.08%.

Moderate: K-NN models performed well, with Grid Search scoring 97.34% and Random Search at 97.03%

Lowest: decision tree models had the lowest accuracy, with Grid Search at 87.94% and Random Search at 88.26%.

5.3.2 Execution time

Fastest: K-NN models were the fastest to train, with Random Search taking only 115 seconds and Grid Search requiring 170 seconds.

5.3.3 Discussion

CNNs may automatically extract low-level characteristics like edges and corners in initial levels, gradually integrating them into more abstract and high-level features in subsequent layers. CNNs are specifically made with numerous layers that allow them to learn complex information.

CNNs extract features automatically and locally. Because of this, manual feature engineering is no longer necessary.

Lastly, a crucial consideration is whether the CNN design is appropriate for picture datasets such as MNIST.

6. CONCLUSION

In this study, we evaluate and compare three optimizing classifiers (CNN, K-NN, and decision tree) that have been done for the classification of the MNIST handwritten digits. In order to improve accuracy and obtain highest recognition rate, two techniques in this work were employed for tuning the parameters of applied classifiers which are: grid and random search. This could result in better classification outcomes by improving the selection of hyperparameter values. The introduced results display that the best classifier to recognize handwritten-digits is CNN. The highest recognition rates (99.18%) are obtained for grid search while (99.08%) recognition rates are obtained for random search. Another mode like decision tree gives worse results, but their training is quicker. For this specific job, CNN models fared better in terms of accuracy than K-NN and decision tree models.

REFERENCE

[1] Elgeldawi, E., Sayed, A., Galal, A.R., Zaki, A.M. (2021). Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis. Informatics, 8(4): 79. https://doi.org/10.3390/informatics8040079

[2] Shekar, B.H., Dagnew, G. (2019). Grid search-based hyperparameter tuning and classification of microarray cancer data. In 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), Gangtok, India, pp. 1-8. https://doi.org/10.1109/ICACCP.2019.8882943

[3] Rahman, A.A., Hasan, M.B., Ahmed, S., Ahmed, T., Ashmafee, M.H., Kabir, M.R., Kabir, M.H. (2022). Two decades of bengali handwritten digit recognition: A survey. IEEE Access, 10: 92597-92632. https://doi.org/10.1109/ACCESS.2022.3202893

[4] Kaensar, C. (2013). A comparative study on handwriting digit recognition classifier using neural network, support vector machine and k-nearest neighbor. In the 9th International Conference on Computing and Information Technology (IC2IT2013) 9th-10th May 2013 King Mongkut's University of Technology North Bangkok, pp. 155-163. https://doi.org/10.1007/978-3-642-37371-8_19

[5] Shao, H., Ma, E., Zhu, M., Deng, X., Zhai, S. (2023). MNIST handwritten digit classification based on convolutional neural network with hyperparameter optimization. Intelligent Automation & Soft Computing, 36(3): 3595-3606. https://doi.org/10.32604/iasc.2023.036323

[6] Hamida, S., Cherradi, B., Raihani, A., Ouajji, H. (2019). Performance evaluation of machine learning algorithms in handwritten digits recognition. In 2019 1st International Conference on Smart Systems and Data Science (ICSSD), Rabat, Morocco, pp. 1-6. https://doi.org/10.1109/ICSSD47982.2019.9003052

[7] Wu, J., Chen, X.Y., Zhang, H., Xiong, L.D., Lei, H., Deng, S.H. (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. Journal of Electronic Science and Technology, 17(1): 26-40. https://doi.org/10.11989/JEST.1674-862X.80904120

[8] Pashine, S., Dixit, R., Kushwah, R. (2021). Handwritten digit recognition using machine and deep learning algorithms. arXiv preprint arXiv:2106.12614. https://doi.org/10.48550/arXiv.2106.12614

[9] Shamim, S.M., Miah, M.B.A., Sarker, A., Rana, M., Al Jobair, A. (2018). Handwritten digit recognition using machine learning algorithms. Indonesian Journal of Science and Technology, 3(1): 29-39.

[10] Firdous, S. (2022). Handwritten character recognition. International Journal for Research in Applied Science & Engineering Technology, 10(5): 1409-1428.

[11] Gupta, A., Sarkhel, R., Das, N., Kundu, M. (2019). Multiobjective optimization for recognition of isolated handwritten Indic scripts. Pattern Recognition Letters, 128: 318-325. https://doi.org/10.1016/j.patrec.2019.09.019

[12] Karakaya, R., Kazan, S. (2021). Handwritten digit recognition using machine learning. Sakarya University Journal of Science, 25(1): 65-71. https://doi.org/10.16984/saufenbilder.801684

[13] Albahli, S., Alhassan, F., Albattah, W., Khan, R.U. (2020). Handwritten digit recognition: Hyperparameters-based analysis. Applied Sciences, 10(17): 5988. https://doi.org/10.3390/app10175988

[14] Géron, A. (2022). Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Inc.

[15] Bakhshi, A., Chalup, S., Noman, N. (2020). Fast evolution of CNN architecture for image classification. Deep Neural Evolution: Deep Learning with Evolutionary Computation, pp. 209-229. https://doi.org/10.1007/978-981-15-3685-4_8

[16] Assegie, T.A., Nair, P.S. (2019). Handwritten digits recognition with decision tree classification: A machine learning approach. International Journal of Electrical and Computer Engineering, 9(5): 4446-4451. https://doi.org/10.11591/ijece.v9i5.pp4446-4451

[17] Nasteski, V. (2017). An overview of the supervised machine learning methods. Horizons Series B, 4: 51. https://doi.org/10.20544/HORIZONS.B.04.1.17.P05

[18] Géron, A. (2017). Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Publishing.

[19] Saad, Y., Hazim, R., Mahroos, S., Mohammed, O., Sadoon, I. (2025). Optimizing random forest for handwritten digit recognition through hyper-parameter tuning. Fusion: Practice and Applications, 155-165. https://doi.org/10.54216/FPA.200112

[20] Saad, Y., Ali, N.A.M., Ali, F.A.M., Abdulbaqi, A.S. (2024). A deep learning-based cervical tumor classification system for telehealthcare monitoring. In International Conference on Innovative Computing and Communication, New Delhi, India, pp. 65-77. https://doi.org/10.1007/978-981-97-3591-4_6

[21] LeCun, Y., Cortes, C. and Burges, C.J.C. (1998). The MNIST database of handwritten digits. New York, USA. https://www.lri.fr/~marc/Master2/MNIST_doc.pdf.

[22] Nguyen, V. (2019). Bayesian optimization for accelerating hyper-parameter tuning. In 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Sardinia, Italy, pp. 302-305. https://doi.org/10.1109/AIKE.2019.00060