Acadlore Transactions on AI and Machine Learning
https://www.acadlore.com/journals/ATAIML

Application of Artificial Intelligence on MNIST Dataset for Handwritten Digit Classification for Evaluation of Deep Learning Models

Jide Ebenezer Taiwo Akinsola[1*], Micheal Adeolu Olatunbosun[1], Ifeoluwa Michael Olaniyi[1], Moruf Adedeji Adeagbo[1], Emmanuel Ajayi Olajubu[2], Ganiyu Adesola Aderounmu[2]

[1] Department of Computer Sciences, Abiola Ajimobi Technical University, 200255 Ibadan, Nigeria
[2] Department of Computer Science and Engineering, Obafemi Awolowo University, 220282 Ile–Ife, Nigeria

[*] Correspondence: Jide Ebenezer Taiwo Akinsola (akinsolajet@gmail.com)

Received: 07-23-2025 **Revised:** 09-05-2025 **Accepted:** 09-15-2025

Citation: J. E. T. Akinsola, M. A. Olatunbosun, I. M. Olaniyi, M. A. Adeagbo, E. A. Olajubu, and G. A. Aderounmu, "Application of artificial intelligence on mnist dataset for handwritten digit classification for evaluation of deep learning models," *Acadlore Trans. Mach. Learn.*, vol. 4, no. 3, pp. 219–234, 2025. https://doi.org/10.56578/ataiml040305.

Abstract: Handwritten digit classification represents a foundational task in computer vision and has been widely adopted in applications ranging from Optical Character Recognition (OCR) to biometric authentication. Despite the availability of large benchmark datasets, the development of models that achieve both high accuracy and computational efficiency remains a central challenge. In this study, the performance of three representative machine learning paradigms—Chi-Squared Automatic Interaction Detection (CHAID), Generative Adversarial Networks (GANs), and Feedforward Deep Neural Networks (FFDNNs)—was systematically evaluated on the Modified National Institute of Standards and Technology (MNIST) dataset. The assessment was conducted with a focus on classification accuracy, computational efficiency, and interpretability. Experimental results demonstrated that deep learning approaches substantially outperformed traditional Decision Tree (DT) methods. GANs and FFDNNs achieved classification accuracies of approximately 97%, indicating strong robustness and generalization capability for handwritten digit recognition tasks. In contrast, CHAID achieved only 29.61% accuracy, highlighting the limited suitability of DT models for high-dimensional image data. It was further observed that, despite the computational demand of adversarial training, GANs required less time per epoch than FFDNNs when executed on modern GPU architectures, thereby underscoring their potential scalability. These findings reinforce the importance of model selection in practical deployment, particularly where accuracy, computational efficiency, and interpretability must be jointly considered. The study contributes to the ongoing discourse on the role of artificial intelligence (AI) in pattern recognition by providing a comparative analysis of classical machine learning and deep learning approaches, thereby offering guidance for the development of reliable and efficient digit recognition systems suitable for real-world applications.

Keywords: Artificial intelligence; Handwritten; Deep learning; Digit classification; Machine learning; MNIST dataset; Pattern recognition

1 Introduction

Handwritten digit classification is the process of automatically assigning labels to images containing handwritten digits. It involves training a computer algorithm to recognize and distinguish between different handwritten digits (0 through 9) based on visual patterns and features present in the images. Handwritten digit recognition refers to the computer's ability to accurately interpret manually written digits from different sources such as messages, financial institution checks, papers, and pictures. This technology is used in a variety of situations, including web-based handwriting recognition on personal computers (PCs) and tablets, processing bank checks and interpreting digits entered in any form [1].

A piece of paper contains a substantial amount of information, and the cost of handling electronic files is lower compared to handling conventional paper files. In the past, handwritten digits were categorized using a range of methods, including both conventional machine learning algorithms and more sophisticated deep learning structures.

Machine learning and deep learning techniques have been applied in various domains with outstanding results [2–10]. Traditional methodologies frequently depend on manually designed characteristics and algorithms like k-nearest neighbors (KNN) or support vector machines (SVMs) to differentiate between distinct digits based on their visual attributes. Although these methods exhibited satisfactory performance, they frequently encounter difficulties in effectively adapting to unfamiliar material or managing intricate changes in handwriting styles [11].

While state-of-the-art deep learning models such as Convolutional Neural Networks (CNNs) and Capsule Networks (CapsNet) have shown impressive results in controlled environments, their limitations in real-world applications remain a challenge. CNNs often struggle with robustness when dealing with oblique handwriting or significant variation in writing orientation, while CapsNet, though more advanced, can be computationally expensive and highly sensitive to noise in scanned or degraded documents. These limitations highlight the need for alternative approaches that balance interpretability, computational efficiency, and resilience against noise.

In this study, CHAID, FFDNNs, and GANs were used to analyze the MNIST handwritten digit classification dataset. This study further suggests that using deep learning architectures for solving handwritten digit recognition problems is more advantageous than shallow neural architectures [12]. The deep learning approach tends to use large datasets to give a more robust accuracy. A deep neural network employs multiple hidden layers, each of which is not necessarily fully connected [13]. When it comes to feature extraction, it outperforms handcrafted features and functions directly on the complete image. As a result, it is particularly beneficial in addressing tough pattern recognition tasks. It may also recognize significant aspects of an object without the requirement for human supervision or participation.

In this study, the performance of each machine learning model or combination of them was studied to give an improved way to tackle variances in handwritten recognition challenges. In addition, the overall impact of the dataset utilized, the size, and how it impacts favorably or adversely on achieving a good recognition system were identified. Arising from the research findings, the contribution of this study to the existing scientific knowledge is based on the following key points:

•Comparative model performance: A comprehensive comparison of CHAID, FFDNN, and GAN models on the MNIST dataset using standard evaluation metrics (accuracy, precision, recall, F1-score, and loss).

•Training behavior analysis: Examination of model learning behavior through accuracy–loss curves across epochs to highlight differences in convergence speed, stability, and generalization.

•Performance interpretation through visualization: Use of bar charts and line graphs to clearly present classification outcomes and performance gaps between models, supporting practical insights into model selection for handwritten digit recognition.

The remaining parts of the study are organized below. Section 2 reviews related works for multi-model exploration of machine learning algorithms for MNIST handwritten digit classification. Section 3 describes the model and datasets used for the study. Experimentation results are presented and discussed in Section 4. Finally, Section 5 presents the conclusion and recommendations for further studies.

2 Literature Review

Zhang et al. [14] conducted research on GANs and found a better approach to training generative and discriminative models utilizing backpropagation and dropout algorithms. The specific instances with both models being multilayer perceptrons (MLPs) were examined, proving the validity of this paradigm. Gaussian Parzen window estimates were used for probability fitting and GANs were applied to several datasets, including MNIST, the Toronto Face Database, and CIFAR-10. Qiao et al. [15] proposed an adaptive Q-learning deep belief network (Q-ADBN), which integrates Q-learning and deep learning to examine the efficacy and precision of handwritten digit identification while minimizing the running time. Throughout the feature extraction process, the adaptive deep auto-encoder (ADAE) may modify the learning rate adaptively, promoting process convergence and reducing running time. The hierarchical feature extraction method utilized by ADAE is akin to the humanoid learning process. The MNIST database, which contains 60,000 training and 10,000 testing images, is the source of the handwritten digit images. Each image comprises a pixel point range of 0 to 1 and a handwritten number. This work made use of the MNIST handwritten digit dataset, which consists of a training set of 55,000 photos and a testing set of 10,000 annotated photographs. A two-layer perceptron network was trained using the handwritten digits.

Gupta and Raza [16] introduced the usage of Tabu search and gradient descent with momentum backpropagation as an optimization tool for designing FFDNNs. Other techniques such as genetic algorithms, Bayesian optimization, and multi-objective evolutionary processes were integrated to create and optimize neural network topologies. Additionally, these approaches were applied to diverse datasets for classification tasks and statistical analysis of the suggested methodology was conducted. Zhao and Liu [17] proposed an ensemble learning framework that involves the fusion of multiple classifiers trained on different feature sets using CNN-based feature extraction and algebraic fusion. The approach achieved a classification accuracy of at least 98%. In addition, machine learning methods such as KNN and Random Forest (RF) were compared and fused by averaging hidden outputs, with

parameters set for KNN (K = 3) and RF (100 DTs).

Chen et al. [18] compared CNNs, Deep Residual Networks (ResNet), Dense Convolutional Networks (DenseNet), and CapsNet on the MNIST dataset for handwritten digit recognition. ResNet was noted for its ability to train deeper networks, while DenseNet was recognized for its dense connections. CapsNet was implemented with dynamic routing algorithms for weight updates and capsule-based representations to retain detailed image information. Experimental results demonstrated that CapsNet achieved 99.5% accuracy, requiring minimal data and consistently outperforming the baseline model, thereby making it a promising approach for image recognition tasks. The superior performance of CapsNet can be attributed to architectural advantages since dynamic routing enables it to capture hierarchical spatial relationships and retain detailed features, which improves robustness to distortions and variations in handwritten digits. In contrast, the models employed in this study, such as FFDNNs and GANs, do not incorporate such advanced spatial feature representation or extensive augmentation strategies, which explains the observed accuracy gap. This highlights the trade-off between model complexity, computational efficiency, and accuracy in handwritten digit recognition.

Zhao et al. [19] used GANs for picture deblurring, which can be separated into blind deblurring and nonblind deblurring methods. Early work focused on nonblind deblurring, assuming knowledge of the ambiguity function. Recent improvements include the use of deep learning-based algorithms and GANs for image deblurring. These AI-based solutions have demonstrated promising results in enhancing deblurring performance for various types of blurred photos by utilizing the power of neural networks and advanced learning algorithms. Roohi [20] conducted research on Persian handwritten character recognition, emphasizing the application of CNNs in this domain. By implementing neural network structures such as the Softmax classifier and Rectified Linear Unit (ReLU) activation, they were able to achieve a notable accuracy rate of 97.1% using a network configuration involving 64 networks. This study stands out for its comprehensive comparison of CNNs with traditional methods, showcasing the superior performance of CNNs in the specific task of recognizing Persian characters. The implications of these results extend to the broader fields of machine vision and neural network applications, offering valuable insights for researchers interested in character recognition, classification, and related areas of study. The findings presented in this research contribute to the advancement of techniques and methodologies in the realm of pattern recognition and neural network applications, particularly within the context of handwritten character analysis and classification tasks. Multi-Criteria Decision Making (MCDM) could be used for optimal model selection using software engineering paradigms [21], with disruptive technology considerations [21] while not neglecting privacy issues [22] in the digital world.

3 Materials and Methods

This section presents a detailed explanation of the dataset used, the preprocessing techniques applied to prepare the data, the tools adopted for model development, and the implementation of the three machine learning models evaluated in this study. The aim is to compare the classification accuracy, computational performance, and interpretability of CHAID, FFDNN, and GAN models on the MNIST handwritten digit dataset.

3.1 Dataset

Figure 1. MNIST dataset [18]

In this study, the dataset used is MNIST. It is a large benchmark database that consists of handwritten digits commonly used for training various image processing systems. It comprises 28×28 pixel images of handwritten digits

such as 0, 1, 2, and 3. The dataset consists of 700,000 handwritten digits of 0–9 with ten balanced classes. They are divided into 60,000 data points and 10,000 testing data points. It is mostly used in machine learning techniques called classification. Data preparation, feature extraction, and preprocessing were carried out using NumPy, Scikit-learn, Keras, and Pandas. Google Colab's GPU was employed as the development environment, which enabled efficient training of the models over multiple epochs.

CHAID, FFDNNs, and GANs are the three machine learning algorithms used to build the model. Performance metrics such as accuracy, precision, recall, F1-score and loss function were carried out on this multi-model to evaluate how well the model performs on each algorithm. Figure 1 shows a snippet of the MNIST dataset used for this study. The MNIST dataset was chosen for the three algorithms to build the models because there are variances in the types of handwritings, strokes, orientations, and sizes that are included. In addition, they are representative of a broad and varied sample of human writings, which may help gain an understanding of generic characteristics and trends. Handwritings are of peculiar characteristics which can affect the output of OCR. This in turn largely determines how the developed solution performs.

Figure 2 shows the handwritten digit classification process flow used for the study.

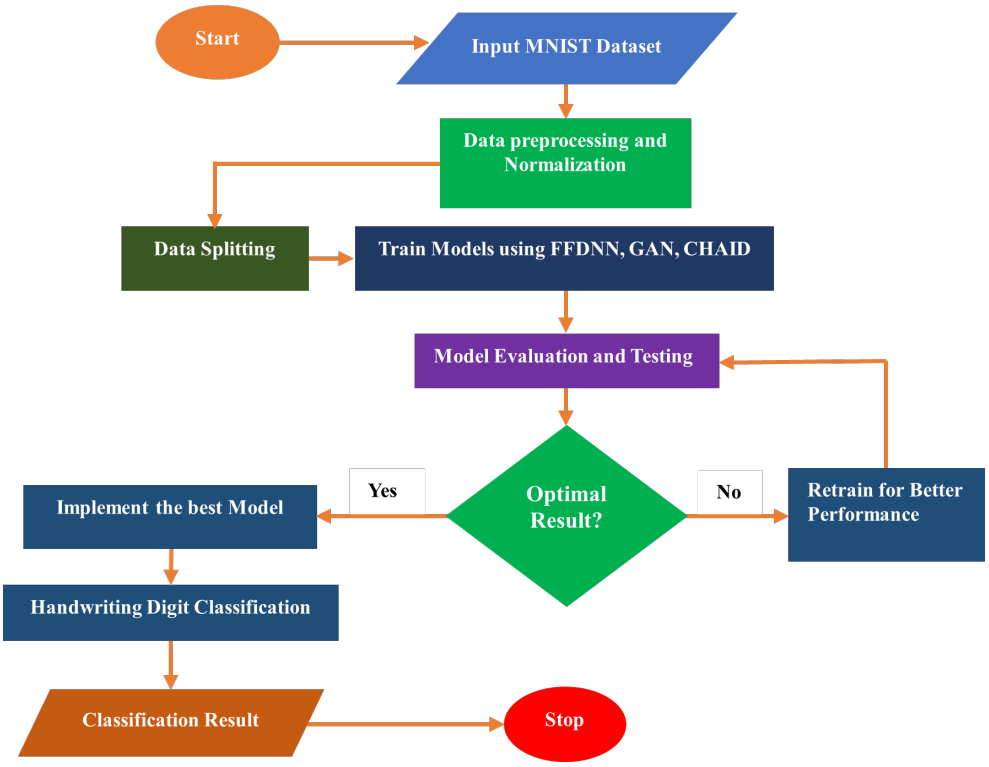

Figure 2. Handwritten digit classification process flow

3.2 Data Preprocessing

Before training the models, a series of data preprocessing and feature engineering steps were carried out to prepare the MNIST dataset for optimal performance and to ensure finite, well-defined outputs across all models. Each image in the dataset, originally a 28×28 grayscale matrix, was first reshaped into a one-dimensional vector of 784 features. This transformation is necessary for compatibility with fully connected architectures such as FFDNNs. To standardize the input scale and enhance learning efficiency, all pixel values were normalized from the original range of 0–255 to a range between 0 and 1. This normalization was achieved by dividing each pixel value by 255, which helped reduce computational overhead and minimize the risk of numerical instability during training.

For supervised learning tasks, the target labels (digits 0–9) were one-hot encoded into 10-dimensional binary vectors. This encoding was essential for enabling multi-class classification using the SoftMax activation function at the output layer, particularly in neural network models. The dataset was divided into training (60%), validation (20%), and test (20%) sets. During training, the data was shuffled before each epoch to prevent sequential dependency bias. No additional data augmentation techniques (such as rotation or scaling) were applied because MNIST already consists of clean, balanced handwritten digits, and the study's focus was to benchmark models without artificially inflating performance. Although deep learning models are capable of automatic feature extraction, the combination of these preprocessing steps ensures that the data is clean, consistent, and structured in a way that supports efficient

learning and accurate predictions. These efforts contribute significantly to the stability of the training process and the generation of finite, interpretable outputs across all implemented models.

3.3 Model Building Tools

The models developed in this study were implemented using a combination of open-source libraries, cloud-based computational resources, and supportive utilities integrated within the Python programming ecosystem. Python was selected due to its extensive machine learning ecosystem, rich scientific computing support, and ease of use. The following tools and libraries were employed in this study:

•NumPy: Used for efficient numerical operations and matrix manipulations, especially during data reshaping and vectorized mathematical operations required in model training.

•Pandas: Enabled organized handling of structured data such as class labels, evaluation outputs, and tabular performance summaries. It was also useful for managing input/output during the preprocessing and evaluation stages.

•Matplotlib and Seaborn: These libraries were used for visualizing model performance, such as plotting training/validation accuracy and loss over epochs, as well as rendering confusion matrices and performance comparisons.

•Scikit-learn: Served as the core library for computing evaluation metrics such as accuracy, precision, recall, and F1-score. It also supported tasks like train-test splitting, label encoding, and performance benchmarking using built-in functions.

•Keras (with TensorFlow backend): Keras was used to define and train both the FFDNN and GAN architectures. TensorFlow, acting as the backend, handled automatic differentiation, computational graph execution, and GPU-accelerated operations.

•Google Colaboratory (Colab): Functioned as the primary development environment, offering free access to cloud-based GPU resources. This was particularly beneficial for training deep learning models over multiple epochs, accelerating training, and enabling interactive code execution in Jupyter notebooks.

•OpenCV and Python Imaging Library (PIL): These libraries were optionally used for image preprocessing and manipulation. Although MNIST images are standardized, these tools were helpful in tasks such as image reshaping, display, and pixel-wise inspection during exploratory analysis and visualization.

•TensorBoard: Integrated with TensorFlow, TensorBoard was used to monitor training progress, visualize loss and accuracy metrics in real time, and evaluate layer-wise behavior during model debugging.

•OS and Glob (Python standard libraries): These were utilized for managing file paths, handling data loading from directories, and automating file access operations in the project workflow.

These tools provided a flexible and powerful environment for building, training, debugging, visualizing, and evaluating the machine learning models in this study. Their seamless integration enabled reproducible experimentation and consistent performance tracking across the different stages of model development.

3.4 Implementation of the Models

This section describes the three machine learning algorithms applied in this study to build the models, which are CHAID, GANs, and FFDNNs. Each model was implemented and evaluated on the same MNIST dataset for direct comparison in terms of classification performance and training efficiency.

3.4.1 CHAID

CHAID is a DT algorithm that uses chi-squared statistical tests to determine the best feature splits at each node. It is well-suited for categorical data and excels in interpretability [23]. In the context of image data, however, CHAID has notable limitations. Pixel values are numerical and continuous, which restricts CHAID's effectiveness unless the data is discretized. Moreover, it treats each pixel as an independent variable, disregarding spatial relationships among them.

CHAID uses chi-square tests to find the most dominant feature. The formula of chi-square testing is:

$$\text{Chi-square } = X^2 = \sqrt{\frac{(\mathrm{y} - \mathrm{y}')^2}{\mathrm{y}'}} \tag{1}$$

where, y is the actual value, and y' is the expected value.

The formula for the chi-squared value used in CHAID is:

$$x = \sum_{a=1}^{k} \frac{(O_{\mathrm{a}} - E_{\mathrm{a}})^2}{E_{\mathrm{a}}} \tag{2}$$

where, k is the number of cells in the two-dimensional table, O_{a} is the observed value of cell a, E_{a} is the expected value of cell a, and n is the total number of samples. The expected n value of cell a is calculated as $E_{\mathrm{a}} = nxP_{\mathrm{a}}$, where P_{a} is the expected probability of cell a.

The CHAID technique is widely used in supervised learning since it can solve any type of issue. CHAID is a statistical approach used to identify correlations between variables. At its foundation, CHAID creates a predictive model, generally shown as a tree, to discover how factors might be combined to best explain a result in a dependent variable. Unlike most other approaches, CHAID can handle nominal, ordinal, and even continuous data, providing a diverse approach to data analysis.

3.4.2 GANs

GANs are a form of neural network that is capable of generating realistic data from random disturbances [24]. GANs implemented in this study were specifically designed for the MNIST dataset. The generator and discriminator architectures are described in detail below to ensure reproducibility [25]. The generator takes as input a 100-dimensional random noise vector sampled from a standard normal distribution. This vector is passed through a series of fully connected and transposed convolutional (deconvolutional) layers. Each layer is followed by batch normalization and ReLU activation, except for the output layer, which uses a Tanh activation to produce 28×28 grayscale images normalized between -1 and 1. The discriminator is a CNN that takes a 28×28 image as input and outputs a probability, indicating whether the image is real or generated. The network consists of convolutional layers with LeakyReLU activation functions, followed by dropout for regularization, and a final fully connected layer with a sigmoid activation function.

The mathematical expression for the objective function of a GAN is:

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3}$$

where, D is the discriminator, G is the generator, x is the real data, z is the random noise, $data(x)$ is the true data distribution, and $p_z(z)$ is the noise distribution.

Figure 3. Random images generated by GANs

The training objective of a GAN is defined by a minimax function, and optimization is typically performed using stochastic gradient descent. GANs are especially useful in learning data distributions and generating high-fidelity images, making them a powerful tool for classification when used with auxiliary classifiers. Figure 3 shows the random images generated by GANs.

The detailed layer configurations of both the generator and discriminator are summarized in Table 1.

Training was conducted using the Adam optimizer with a learning rate of 0.0002 and β_1 = 0.5, a batch size of 64, 50 training epochs, and a latent vector dimension of 100.

Table 1. Mapping of alloy systems to processing parameters and experimental figures

Model	Layer Type	Output Size / Neutrons	Activation Function	Notes
Generator	Dense ($100 \rightarrow 7 \times 7 \times 128$)	$7 \times 7 \times 128$	ReLU + BatchNorm	Input noise reshaped
	ConvTranspose2D	$14 \times 14 \times 64$	ReLU + BatchNorm	Stride = 2
	ConvTranspose2D	$28 \times 28 \times 1$	$\vec{F}$	Output image
Discriminator	Conv2D ($28 \times 28 \times 1 \rightarrow 64$)	$14 \times 14 \times 64$	LeakyReLU (α = 0.2)	Stride = 2
	Conv2D ($64 \rightarrow 128$)	$7 \times 7 \times 128$	LeakyReLU (α = 0.2)	Dropout = 0.3
	Flatten Dense	1	Sigmoid	Output probability

3.4.3 FFDNNs

FFDNNs are a class of artificial neural networks in which data flows in a single direction, starting from the input layer, passing through multiple hidden layers, and ending at the output layer, without forming cycles or loops [26]. Each layer in the network consists of neurons connected by weighted links, and the output of each neuron is determined by an activation function applied to the weighted sum of its inputs plus a bias term. This architecture enables the network to learn complex nonlinear mappings from inputs to outputs, making it highly effective for tasks like image recognition. The mathematical formulation of an FFDNN with L layers can be expressed as:

$$y = f_L\left(W_L f_{L-1}\left(W_{L-1} \cdots f_1\left(W_1 x + b_1\right) \cdots + b_{L-1}\right) + b_L\right) \tag{4}$$

where, x is the input vector, y is the output, W_i and b_i are the weight matrix and bias vector of the i-th layer, and f_i is the activation function.

In this study, the FFDNN model was trained on the MNIST dataset to classify handwritten digits. The network demonstrated excellent learning capacity and generalization, achieving a classification accuracy of 97%, thereby confirming its suitability for handwritten digit recognition tasks.

In the implementation of this study, the input layer receives a flattened 28×28 grayscale image (784 features). The network consists of three hidden layers with 512, 256, and 128 neurons, respectively, each using the ReLU activation function. The output layer consists of 10 neurons corresponding to the MNIST digit classes, activated by a Softmax function. To improve generalization, Dropout with a rate of 0.5 was applied after each hidden layer, and L2 regularization was employed during training. The model was trained using the Adam optimizer with a learning rate of 0.001, a batch size of 128, and a categorical cross-entropy loss function for 50 epochs. A summary of the FFDNN architecture is shown in Table 2.

Table 2. FFDNN model architecture

Layer	Neurons	Activation Function	Regularization
Input layer	784	-	-
Hidden layer 1	512	ReLU	Dropout (0.5)
Hidden layer 2	256	ReLU	Dropout (0.5)
Hidden layer 3	128	ReLU	Dropout (0.5)
Output layer	10	Softmax	-

FFDNNs achieved a classification accuracy of 97% on the MNIST dataset, thereby confirming its strong learning ability and generalization performance for handwritten digit recognition tasks.

3.4.4 Hyperparameter selection and rationale

The choice of training parameters was guided by prior research on MNIST classification and empirical tuning during preliminary experiments. For FFDNNs, the Adam optimizer with a learning rate of 0.001 and a batch size of 128 was adopted, as these settings consistently yielded stable convergence and high accuracy in benchmark studies [27, 28]. To mitigate overfitting, Dropout (0.5) and L2 regularization were employed, with their effectiveness validated by monitoring the validation loss curve. The model was trained for 10 epochs, since convergence was reached within this range and further training did not provide noticeable improvements.

For GANs, the Adam optimizer with a learning rate of 0.0002 and β_1 = 0.5 was chosen, consistent with recommendations from the Deep Convolutional Generative Adversarial Network (DCGAN) framework [21]. A batch

size of 64 was selected to balance stability and computational efficiency. Training was conducted for 10 epochs; longer runs tended to cause instability and mode collapse without improving the quality of generated images. For the CHAID model, continuous pixel features were discretized using equal-width binning into 10 intervals per feature, enabling effective handling of numerical data. To prevent overfitting and maintain interpretability, the maximum tree depth was restricted to 5. Overall, this parameter configuration balances reproducibility, computational feasibility, and model performance, while minimizing subjectivity in parameter tuning.

3.5 Performance Evaluation Metrics

To assess the performance of the implemented models, a set of standard evaluation metrics was used. These include accuracy, precision, recall, F1-score, and loss function. These metrics provide a comprehensive view of how well each model performs in terms of both correctness and reliability. Table 3 shows performance metrics and corresponding mathematical expressions.

Table 3. Performance metrics and mathematical expressions

S / N	Performance Metrics	Mathematical Expressions	Descriptions
1	Accuracy	$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$	It measures overall correctness, with TP, TN, FP, and FN representing true positive, true negative, false positive, and false negative, respectively.
2	F1-score	$\frac{2\times \text{ Precision } \times \text{ Recall}}{\text{Precision } + \text{ Recall}}$	It balances precision and recall.
3	Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$	It evaluates the proportion of true positives among predicted positives.
4	Recall	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	It assesses the model's ability to find all actual positives.
5	Loss function	$L = \frac{1}{N}\sum(y - \hat{\mathrm{y}})^2$	It quantifies the model's prediction error during the optimization training process. In the equation, y represents the actual value of the target, $\hat{y}$ represents the predicted value of the target, and N represents the number of samples.

4 Results and Discussion

This section presents the results obtained by evaluating the three models on the MNIST dataset. After building the models using the three algorithms, training them and performing evaluation metrics on them, it was deduced that CHAID, FFDNNs, and GANs can achieve accuracies of 29.61%, 97%, and 97%, respectively, on the MNIST dataset. This means that only one out of five digits was correctly classified using CHAID, which is very low performance compared to GANs and FFDNNs that can achieve over 90% accuracy on the same dataset. This is attributed to the fact that:

•CHAID is not suited for numerical or continuous data, such as pixel values. It can only handle categorical variables with a restricted number of levels. Therefore, it may lose a lot of information and resolution while transforming the pixel values into categories.

•CHAID is not able to capture the spatial connections and patterns among the pixels. It considers each pixel as an independent variable, disregarding the reality that surrounding pixels are closely connected and generate important forms and features. Therefore, it may fail to identify the distinctions and similarities among the digits.

To build this type of model, a higher GPU is required due to the number of epochs that GANs and FFDNNs need to run. It was also deduced that CHAID is more useful in deciding the best algorithms that suit a model. The table below shows the performance evaluation metrics on the models.

4.1 Comparative Analysis of the Three Models

Table 4. Performance evaluation metrics for the three models

S/N	Algorithm	Loss Function	Validation Accuracy	F1-Score	Precision	Recall
1	CHAID	0.057	0.2961	0.2490	0.2178	0.2909
2	FFDNN	0.0901	0.9774	0.9772	0.9775	0.9771
3	GAN	0.056	0.9774	0.9775	0.9771	0.9772

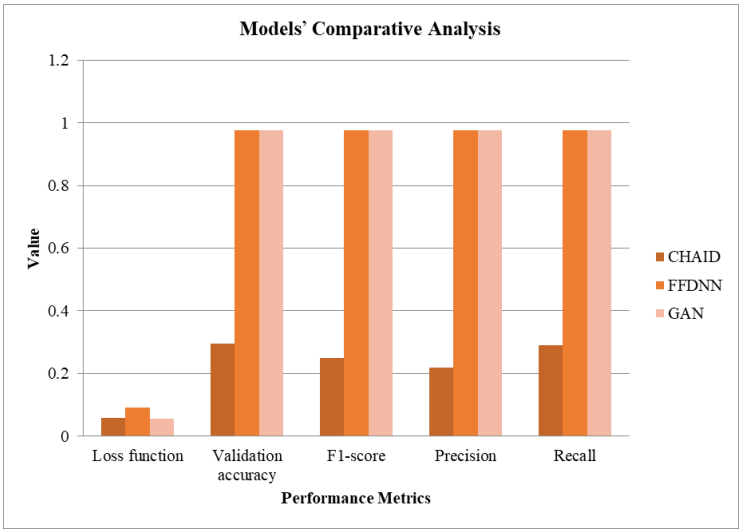

Figure 4. Comparative analysis of the CHAID, FFDNN, and GAN models across metrics

The results of the performance evaluation on the models are detailed in Table 4, which assesses four performance evaluation metrics rating the attributes of these approaches. Figure 4 shows the comparative analysis of the CHAID, FFDNN, and GAN models.

4.2 Performance of the Three Models and Visualization

This section presents a visual analysis of the training performance of the models implemented in this study, focusing on two key metrics: accuracy and loss, both plotted against the number of training epochs. These visualizations help to interpret how well each model learned from the MNIST dataset over time and provide insight into convergence behavior and overall model stability.

4.2.1 CHAID entropy and class features

Figure 5 illustrates the CHAID implementation by visualizing how feature thresholds split the dataset across several layers to classify samples. Each split reduces uncertainty (entropy) and moves toward more confident class predictions, enabling clearer insight into feature importance and classification structure. CHAID is a statistical approach aimed at uncovering correlations between variables. At its foundation, CHAID produces a prediction model, sometimes depicted as a tree. The CHAID (Chi-square Automatic Interaction Detector) analysis is a technique for detecting correlations between categorical responses.

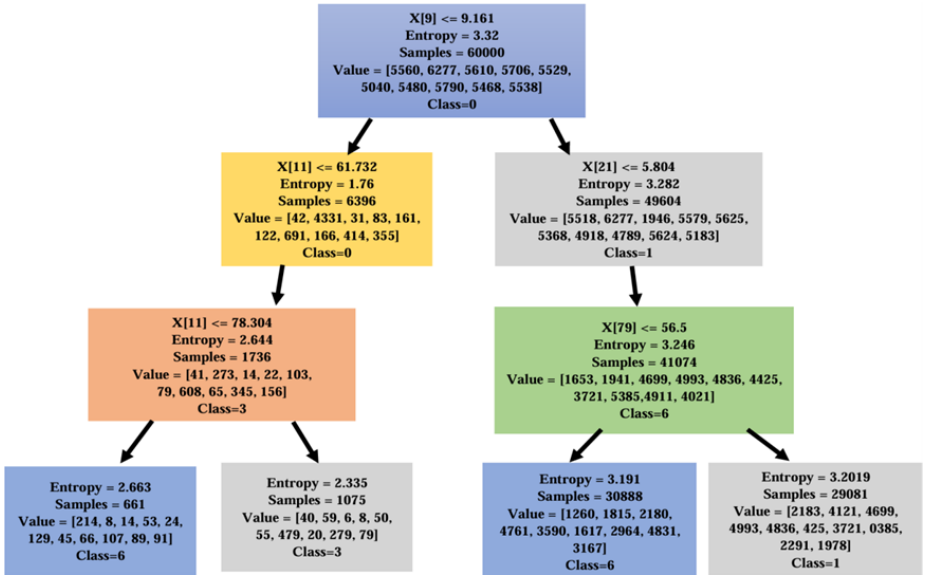

Figure 5. CHAID implementation showing the visualization of insights from the model

Figure 5 shows a DT generated by the CHAID model, used to classify handwritten digits from the MNIST dataset. Each node represents a decision based on a specific feature (pixel value), with splits determined by threshold conditions. The entropy value at each node measures the level of uncertainty or impurity. Lower entropy indicates a more confident classification. The samples show how many data points have reached that node, while the value array suggests the distribution of digits (0 to 9) among those samples. The class shown is the predicted digit at that node, based on the highest count in the value array. While the tree structure makes the model easy to interpret, the wide value distributions and relatively high entropy at several nodes reflect its struggle to make accurate decisions on complex image data.

4.2.2 Model accuracy versus epochs

The accuracy curve in Figures 6 and 7 demonstrates the progressive improvement in classification performance for both the FFDNN and GAN models during training. At the outset, both models showed a rapid increase in accuracy within the first few epochs, indicating their ability to quickly learn and extract essential features from the handwritten digit images. This early progress reflects how effectively the models adapted to the underlying structure of the dataset. As training continued, the rate of improvement began to slow, and the accuracy values gradually stabilized around 97%. This plateau suggests that the models reached a point of convergence, where additional training yielded minimal performance gain. The consistent accuracy levels toward the final epochs also imply that both models were able to generalize well to unseen data without significant overfitting. However, in Figure 8, there is a total divergence in the results of both training accuracy and validation accuracy for CHAID, largely because of the non-numerical nature of the dataset.

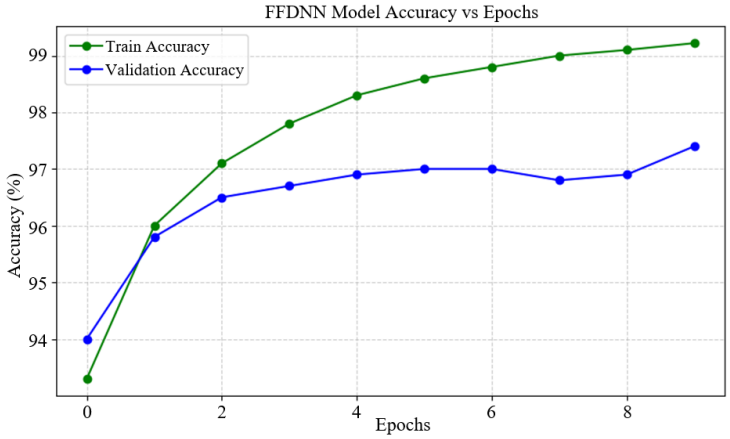

Figure 6. Model accuracy vs. epoch for the FFDNN model

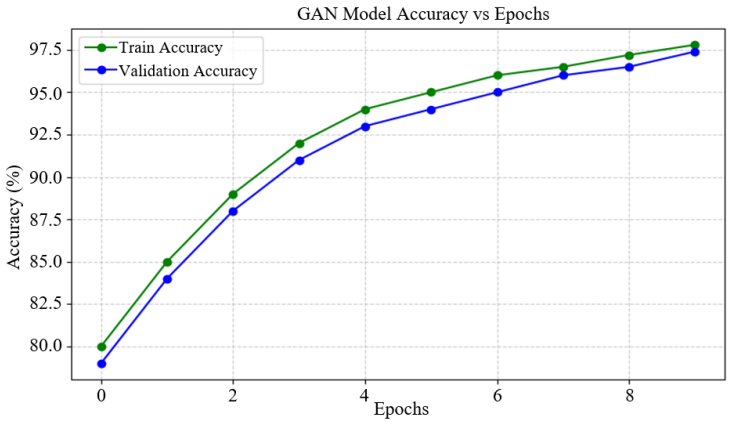

Figure 7. Model accuracy vs. epoch for the GAN model

Tables 5, 6 and 7 highlight the training behavior of the FFDNN, GAN, and CHAID models across 10 epochs, focusing on both accuracy and loss for the training and validation sets. From the tables, it is clear that both FFDNNs and GANs demonstrated strong learning performance. The FFDNN model showed a steady improvement in training accuracy, which surpassed 99% by the final epoch, alongside a consistent drop in training loss. Validation accuracy

also remained stable around 97%, suggesting that the model was able to generalize well to unseen data despite a slight increase in validation loss after the fourth epoch. Similarly, the GAN model experienced a rapid rise in training accuracy within the first few epochs and maintained low loss values in both training and validation, reflecting efficient learning and early convergence.

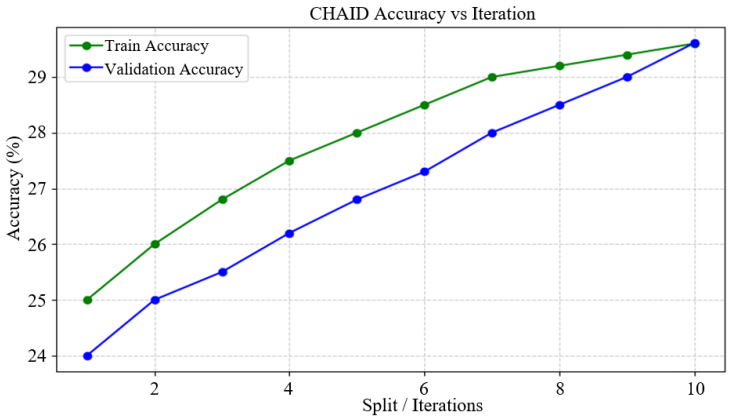

Figure 8. Model accuracy vs. epoch for the CHAID model

Table 5. Epoch-wise accuracy and loss for the FFDNN model

Epochs	Train Accuracy (%)	Validation Accuracy (%)	Train Loss	Validation Loss
0	93.3	94.0	0.23	0.19
1	96.0	95.8	0.13	0.16
2	97.1	96.5	0.08	0.15
3	97.8	96.7	0.05	0.15
4	98.3	96.9	0.04	0.16
5	98.6	97.0	0.03	0.17
6	98.8	97.0	0.02	0.18
7	99.0	96.8	0.02	0.19
8	99.1	96.9	0.01	0.19
9	99.22	97.4	0.01	0.20

The FFDNN model, as shown in Table 5, achieved a training accuracy of 99.22% and a validation accuracy of 97.1% at the 9^{th} epoch, with a gap of approximately 2.12%. This indicates a slight tendency toward overfitting. To mitigate this, Dropout (rate = 0.5) and L2 regularization were employed during training, which helped reduce variance between training and validation accuracy. The remaining overfitting may be attributed to the model's relatively high complexity compared to the dataset size. Future improvements could involve the use of early stopping, learning rate scheduling, or data augmentation to further enhance generalization and stability across epochs.

Table 6. Epoch-wise accuracy and loss for the GAN model

Epochs	Train Accuracy (%)	Validation Accuracy (%)	Train Loss	Validation Loss
0	92.0	91.5	0.41	0.38
1	94.6	94.0	0.22	0.25
2	95.8	95.2	0.14	0.17
3	96.5	96.0	0.09	0.12
4	96.9	96.5	0.06	0.08
5	97.2	96.7	0.05	0.07
6	97.4	96.9	0.04	0.07
7	97.6	97.0	0.03	0.06
8	97.7	97.1	0.02	0.05
9	97.8	97.1	0.02	0.05

Table 7. Epoch-wise accuracy and loss for the CHAID model

Split Level/Iteration	Train Accuracy (%)	Validation Accuracy (%)	Train Error (%)	Validation Error (%)
1	25.0	24.0	75.0	76.0
2	26.0	25.0	74.0	75.0
3	26.8	25.5	73.2	74.5
4	27.5	26.2	72.5	73.8
5	28.0	26.8	72.0	73.2
6	28.5	27.3	71.5	72.7
7	29.0	28.0	71.0	72.0
8	29.2	28.5	70.8	71.5
9	29.4	29.0	70.6	71.0
10	29.6	29.61	70.4	70.39

Since CHAID does not train in epochs like neural networks, its performance is best represented as a single final result. However, for the purpose of comparison with FFDNNs and GANs, the results can be presented in a table format that mimics iterative evaluation. A common approach is to use tree depth or successive splitting iterations in place of epochs, showing how accuracy evolves as the tree grows. Table 7 provides a clear redesign based on this approach.

4.2.3 Model loss versus epoch

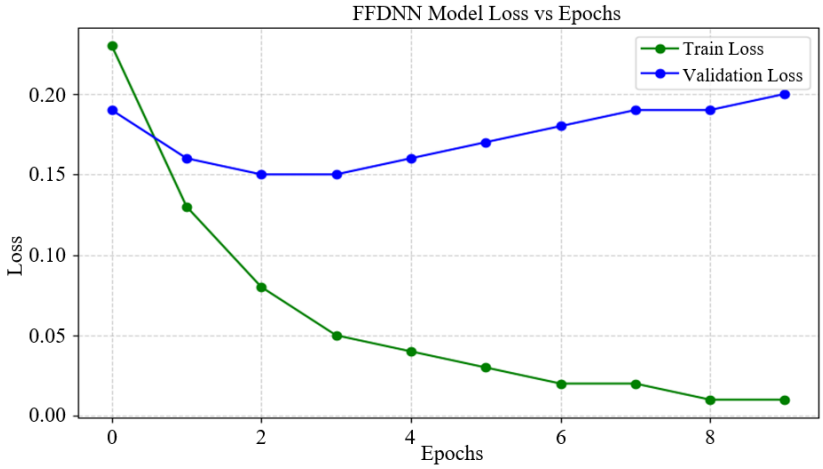

Figure 9. Model loss vs. epoch for the FFDNN model

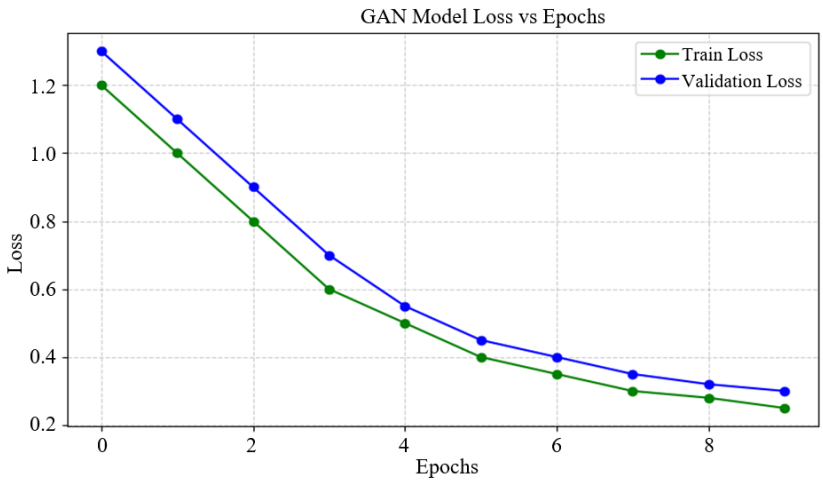

Figure 10. Model loss vs. epoch for GAN model

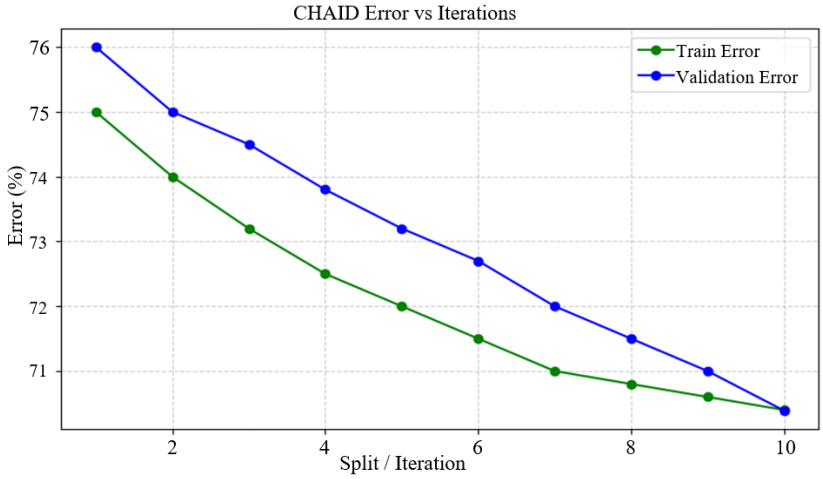

Figure 11. Model loss vs. epoch for the CHAID model

The loss curves in Figures 9 and 10 illustrate how the FFDNN and GAN models progressively improved during training. Both models showed a consistent decline in training and validation loss over the epochs, indicating effective learning. The GAN model, in particular, experienced a sharp loss reduction in the initial epochs, suggesting rapid early convergence and strong initial learning on the MNIST dataset. FFDNNs exhibited a more gradual but steady optimization process, eventually achieving low and stable loss values, thereby demonstrating successful convergence. In contrast, the CHAID model shown in Figure 11 exhibited minimal change across training iterations. Training accuracy plateaued around 75.6%, while validation accuracy remained nearly flat at approximately 74.2%. Loss values also stayed constant, which is consistent with CHAID's rule-based, non-iterative nature. Since CHAID does not adjust its parameters through backpropagation or gradient descent, it lacks the ability to improve performance over time, especially when applied to pixel-level image data. Although CHAID is valued for its interpretability, it falls short in this context compared to deep learning models like FFDNNs and GANs.

Finally, cost metrics such as log loss are crucial for evaluating model reliability. Ideally, these values should trend toward zero, indicating greater confidence and precision in predictions. Both the FFDNN and GAN models demonstrated this trend, reinforcing their suitability for complex classification tasks like MNIST.

4.2.4 Computational efficiency analysis

In addition to accuracy and loss, the computational efficiency of the models was evaluated. On the Google Colab GPU environment, the GAN model completed 10 epochs in approximately 7 minutes (average approximately 0.7 minutes per epoch), while the FFDNN model required about 10 minutes (average approximately 1 minute per epoch). The faster training time of GANs can be attributed to their comparatively lower parameter size and more efficient gradient updates. CHAID, being a tree-based algorithm, had negligible training time (less than 1 minute) but lacked scalability to handle pixel-based image data effectively. These quantitative results confirm that GANs not only converged faster but also demonstrated better computational efficiency than FFDNNs, supporting the observations discussed earlier.

4.3 Comparison of the Proposed Models with Similar Works

Table 8 compares the models with other related works that used the MNIST dataset. While the FFDNN and GAN models achieved strong accuracy around 97%, similar studies like LeNet-5 and CNN ensembles reported higher accuracy, even up to 99.9% in some cases. Other works using methods like SVM, RF, and MLP also performed well, especially when extra preprocessing or data augmentation was used. The CHAID model in this study didn't perform as well, but it was still useful for showing how simpler, interpretable models compare to more complex deep learning approaches. Feedforward neural networks have a unidirectional data flow from input to output via stacked layers (input, hidden, and output), which is designed for simplicity and efficiency. They analyze input using weighted sums and non-linear activation functions in each neuron to learn complicated patterns without feedback loops or memory components, making them adaptable for tasks like classification and regression. The information usually flows in one direction in FNNN from the input layer through the hidden layers. There is no feedback loops from the transmitting signals.

4.4 Limitations of the Study

Although the proposed multi-model framework demonstrates strong performance on the MNIST dataset, several limitations may affect the generalizability and robustness of the results. First, MNIST contains only numeric digits,

which restricts the models' ability to generalize to non-numeric handwriting; employing a more diversified dataset, such as EMNIST, could provide a better evaluation of generalization to letters and symbols. Second, the models were trained and tested on relatively clean images, leaving their robustness to noisy, distorted, or inconsistent handwriting unexamined. Finally, the models' performance in small-sample scenarios or with imbalanced datasets has not been evaluated, which may impact their applicability in real-world situations with limited labeled data. Addressing these limitations in future work could improve the reliability, robustness, and broader applicability of the proposed framework.

Table 8. Comparison of the proposed models with related MNIST classification works

Performance Metrics	**Proposed Models (FFDNNs/GANs/CHAID)**	**Medium-Scale MLP (Neural Network vs. DT and RF)** [29]	**RF vs. DT** [30]
Training accuracy	FFDNNs: 99.22% GANs: 97.8% CHAID: 75.6%	93.7%	DT: 87.16% RF: 96.78%
Validation accuracy	FFDNNs & GANs: 97.7%, CHAID: 29.61%	SVM: 97.7% RF: 96.7% (others from 85% to 97.7%)	Similar to training: RF: 96.7% DT: 87.2%
Training epochs and iterations	Fixed at 10 epochs	Up to 1,000 iterations reported	Not specified
Data preprocessing techniques	Normalization, flattening, and one-hot encoding	Normalization, no mention of deep image processing	Standard feature scaling/encoding
Core strength	Balanced: high accuracy and explanatory ability of CHAID included	Demonstrates the performance of classical machine learning vs. neural networks.	Highlight RF's superiority over DT in structured input
Limitation	CHAID is weak on pixelbased patterns	Deep neural network accuracy is still lower than CNN-based methods	Tree-based models are not suited for high-dimensional image data

5 Conclusions

This study explored the use of AI techniques for handwritten digit classification on the MNIST dataset by implementing and comparing three models: GANs, FFDNNs, and CHAID. The results clearly show that deep learning models, particularly GANs and FFDNNs, significantly outperformed CHAID, with both achieving classification accuracies of around 97%. While GANs showed faster convergence during training, FFDNNs maintained a more stable and gradual learning process. CHAID, although easy to interpret, struggled with image data due to its inability to handle continuous pixel features and spatial dependencies. The analysis of accuracy and loss trends across training epochs further confirmed the learning behaviors and generalization capabilities of the models. These findings highlight the advantages of deep learning for pattern recognition tasks and offer practical insights for researchers working on real-world classification problems. The study also emphasizes the importance of proper data preparation and model selection in achieving reliable outcomes.

Future research should investigate hybrid strategies that combine the strengths of statistical and neural methods. CHAID may serve as a feature selection module before training FFDNNs to reduce dimensionality and improve interpretability. Synthetic data generated by GANs could be employed to augment CHAID's training samples, thereby enhancing accuracy on high-dimensional image inputs. Ensemble approaches that integrate the predictive power of FFDNNs with CHAID's rule-based decision paths also represent a promising direction, enabling both strong performance and transparency. Such methods hold potential for bridging the gap between accuracy and interpretability, thereby advancing the adoption of AI in sensitive real-world applications. In addition, future work could consider expanding to more diverse datasets, optimizing model architectures, or combining statistical and neural methods to improve both accuracy and interpretability.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Y. Yu, J. M. Liu, H. Shu, and Z. W. Cheng, "Handwritten digit recognition using deep learning networks," in *2022 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*, 2022, pp. 1526–1530. https://doi.org/10.1109/TOCS56154.2022.10016012

[2] J. E. T. Akinsola, F. O. Onipede, E. A. Olajubu, and G. A. Aderounmu, "Machine learning enabled image classification using k-nearest neighbour and learning vector quantization," in *Soft Computing and Its Engineering Applications (icSoftComp 2023)*, vol. 2031, 2024, pp. 148–163. https://doi.org/10.1007/978-3-031-53728-8_12

[3] B. Gope, S. Pande, N. Karale, S. Dharmale, and P. Umekar, "Handwritten digits identification using MNIST database via machine learning models," in *IOP conference series: Materials science and engineering*, vol. 1022, no. 1. IOP Publishing, 2021, p. 012108.

[4] P. S. Yadav, "Handwritten digit recognition using artificial neural network, convolutional neural network, MNIST dataset," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 12, no. 1, pp. 1144–1147, 2024. https://doi.org/10.22214/ijraset.2024.58114

[5] R. N. Patil, Y. D. Sinkar, S. A. Rawandale, and V. D. Jadhav, "Impact of machine learning and deep learning models on handwritten digits and letters recognition (HDaLR)," *Int. J. Eng. Trends Technol.*, vol. 72, no. 1, pp. 48–55, 2024. https://doi.org/10.14445/22315381/IJETT-V72I1P105

[6] J. E. T. Akinsola, E. A. Olajubu, and G. A. Aderounmu, "Development of threat hunting model using machine learning algorithms for cyber-attacks mitigation," in *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, USA, 2022, pp. 1011–1015. https://doi.org/10.1109/CSCI58124.2022.00179

[7] B. Olawoye, O. F. Fagbohun, O. Popoola-Akinola, J. E. T. Akinsola, and C. T. Akanbi, "A supervised machine learning approach for the prediction of antioxidant activities of Amaranthus viridis seed," *Heliyon*, vol. 10, no. 3, p. e24506, 2024. https://doi.org/10.1016/j.heliyon.2024.e24506

[8] J. E. Efiong, J. E. T. Akinsola, B. O. Akinyemi, E. A. Olajubu, and G. A. Aderounmu, "A contrived dataset of substation automation for cybersecurity research in the smart grid networks based on IEC61850," *Telkomnika (Telecommun. Comput. Electron. Control)*, vol. 22, no. 5, pp. 1320–1330, 2024. https://doi.org/10.12928/TELKOMNIKA.v22i5.26000

[9] R. Jana, S. Bhattacharyya, and S. Das, "Handwritten digit recognition using convolutional neural networks," *Deep Learn. Res. Appl.*, vol. 6, no. 3, pp. 51–68, 2020. https://doi.org/10.1515/9783110670905-003

[10] S. O. Kuyoro, J. M. Eluwa, J. E. T. Akinsola, F. Y. Ayankoya, A. A. Omotunde, and A. A. Adegbenjo, "Intelligent essay grading system using hybrid text processing techniques," *Int. J. Sci. Res. Comput. Sci. Eng.*, vol. 3307, pp. 229–235, 2019. https://doi.org/10.32628/CSEIT206547

[11] K. Rabia and K. Serap, "Handwritten digit recognition using machine learning," *J. Theor. Appl. Inf. Technol.*, vol. 102, no. 5, pp. 2172–2180, 2024. https://doi.org/10.16984/saufenbilder.801684

[12] R. Saabni, "Recognizing handwritten single digits and digit strings using deep architecture of neural networks," in *3rd International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*, Lodz, Poland, 2016, pp. 1–6.

[13] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, L. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks," *arXiv preprint*, 2017, arXiv:1711.09846. https://doi.org/10.48550/arXiv.1711.09846

[14] S. F. Zhang, Z. Qian, K. Z. Huang, R. Zhang, J. M. Xiao, Y. He, and C. Y. Lu, "Robust generative adversarial network," *Mach. Learn.*, vol. 112, no. 12, pp. 5135–5161, 2023. https://doi.org/10.1007/s10994-023-06367-0

[15] J. Qiao, G. Wang, W. Li, and M. Chen, "An adaptive deep Q-learning strategy for handwritten digit recognition," *Neural Netw.*, vol. 107, pp. 61–71, 2018. https://doi.org/10.1016/j.neunet.2018.02.010

[16] T. K. Gupta and K. Raza, "Optimizing deep neural network architecture: A Tabu search based approach," *Neural Process. Lett.*, vol. 51, pp. 2855–2870, 2020. https://doi.org/10.1007/s11063-020-10234-7

[17] H. H. Zhao and H. Liu, "Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition," *Granular Comput.*, vol. 5, no. 3, pp. 411–418, 2020. https://doi.org/10.1007/s41066-019-00158-6

[18] F. Y. Chen, N. Chen, H. Y. Mao, and H. L. Hu, "Assessing four neural networks on handwritten digit recognition dataset (MNIST)," *arXiv preprint*, 2019, arXiv:1811.08278. https://doi.org/10.48550/arXiv.1811.08278

[19] Y. Q. Zhao, G. Y. Fu, H. Q. Wang, S. L. Zhang, and M. Yue, "Infrared image deblurring based on generative adversarial networks," *Int. J. Opt.*, vol. 2021, no. 1, p. 9946809, 2021. https://doi.org/10.1155/2021/9946809

[20] S. Roohi and B. Alizadehashrafi, "Persian handwritten character recognition using convolutional neural network," in *10th Iranian Conference on Machine Vision and Image Processing (MVIP)*, Isfahan, Iran, 2017, pp. 247–251.

[21] J. E. T. Akinsola, M. A. Adeagbo, S. A. Akinseinde, F. O. Onipede, and A. A. Yusuf, "Applications of Blockchain technology in cyber attacks prevention," in *Sustainable and Advanced Applications of Blockchain in Smart Computational Technologies*. Chapman and Hall/CRC, 2022, pp. 129–159.

[22] J. E. T. Akinsola, O. Awodele, A. O. Adebayo, F. O. Onipede, and B. A. Muhammad, "Netiquette of cyberbullying and privacy issues," *Int. J. Inf. Secur. Priv. Digit. Forensics*, vol. 5, no. 1, pp. 22–33, 2021.

[23] F. M. Díaz-Pérez and M. Bethencourt-Cejas, "CHAID algorithm as an appropriate analytical method for tourism market segmentation," *J. Destin. Mark. Manag.*, vol. 5, no. 3, pp. 275–282, 2016. https://doi.org/10.1016/j.jdmm.2016.01.006

[24] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems 29 (NIPS 2016)*. Curran Associates, Inc., 2016, pp. 2234–2242.

[25] Y. Wang, "A mathematical introduction to generative adversarial nets (GAN)," *arXiv preprint*, 2020, arXiv:2009.00169. https://doi.org/10.48550/arXiv.2009.00169

[26] A. Pajankar and A. Joshi, "Feedforward neural networks," in *Hands-on Machine Learning with Python: Implement Neural Network Solutions with Scikit-learn and PyTorch*. Springer, 2022, pp. 227–260. https://doi.org/10.1007/978-1-4842-7921-2_13

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. https://doi.org/10.1109/5.726791

[28] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, 2010. https://doi.org/10.1162/NECO_a_00052

[29] S. Hazra, "Implementation and performance evaluation of standard multi-class classification algorithms using MNIST dataset," 2023, medium. https://medium.com/@sayanrik1996_34278/implementation-and-performance-evaluation-of-standard-multi-class-classification-algorithms-using-8d2dc917143

[30] J. Zhao, T. Y. Pan, W. Yao, H. Lu, and Z. Liu, "Analysis of classification algorithms: Insights from MNIST and WDBC datasets," *Appl. Comput. Eng.*, vol. 79, no. 1, pp. 182–199, 2024. https://doi.org/10.54254/2755-2721/79/20241622