DATA MODELING

DS3122

PROJECT TITLE:

MNIST Dataset Of Handwritten Digits

COURSE REPRESENTS :

DR. THANI ALQURASHI

SUBMITTED BY :

STUDENTS NAME	STUDENTS ID
RAKHAA MATUQ NOOH	444001287
MANAR ALI ALSUBHI	444003523
REHAM FAISAL ALSUBHI	444003014
JANA ABDULRAOUF ALLIHYANI	444001382
LAYAN ADEL BABKOUR	444002368

TABLE OF CONTENT

1-Abstract

The primary goal of this project was to classify handwritten digits from the MNIST dataset using four different machine learning algorithms: Convolutional Neural Networks (CNN), Support Vector Machines (SVM), Decision Trees, and Multilayer Perceptron (MLP). The MNIST dataset comprises 70,000 grayscale images of handwritten digits, labeled from 0 to 9, providing an excellent benchmark for testing classification models. Among the models tested, CNN proved to be the most effective for MNIST digit classification.

2-Introduction

Image classification, especially for handwritten digits, poses unique challenges due to the wide variability in individual handwriting styles. The MNIST (Modified National Institute of Standards and Technology) dataset is a well-known benchmark in image classification tasks, commonly used for evaluating algorithms in recognizing handwritten digits from 0 to 9. The MNIST dataset comprises 70,000 grayscale images, each sized 28x28 pixels, representing digits written by various individuals. Of these samples, 60,000 images are allocated for training, while the remaining 10,000 are used to test model performance.

Image classification, especially for handwritten digits, poses unique challenges due to the wide variability in individual handwriting styles. The MNIST (Modified National Institute of Standards and Technology) dataset is a well-known benchmark in image classification tasks, commonly used for evaluating algorithms in recognizing handwritten digits from 0 to 9. The MNIST dataset comprises 70,000 grayscale images, each sized 28x28 pixels, representing digits written by various individuals. Of these samples, 60,000 images are allocated for training, while the remaining 10,000 are used to test model performance. To address this problem, this project explores four different machine learning approaches: Convolutional Neural Networks (CNN), Support Vector Machines (SVM), Decision Trees, and Multilayer Perceptron (MLP). Each of these models employs distinct methodologies and strengths to manage the variations in MNIST data, making this comparison essential for evaluating the effectiveness of each model in handling such variability. For instance, CNNs have an inherent advantage due to their ability to capture spatial hierarchies in images and recognize complex patterns in visual data effectively. In contrast, MLP and SVM offer more traditional methods for handling structured data but require additional feature engineering to achieve comparable performance to CNNs. The methodology employed involves preprocessing the MNIST images, optimizing model parameters, and training each algorithm on the data to achieve the highest possible classification accuracy. After training, the models are evaluated based on test data accuracy, allowing for a comparison of their effectiveness and drawing conclusions about the most suitable methods for handwritten digit recognition. Convolutional Neural Networks (CNNs) have consistently demonstrated high effectiveness in image recognition tasks and serve as a benchmark for assessing the performance of other algorithms in this project.

3.Exploratory Dataset

3.1 Number of Classes:

Total Classes:10

Classes Represented:The digits from 0 to 9.

3.2 Types of Images

Image Format:Grayscale (1 channel)

Image Size:Each image is 28x28 pixels.

-Total Images: 70,000 images (60,000 training iThere are no missing images in the dataset images and 10,000 testing images).

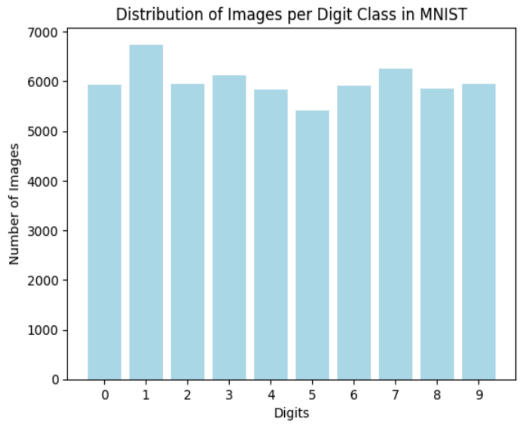
3.3 Variability

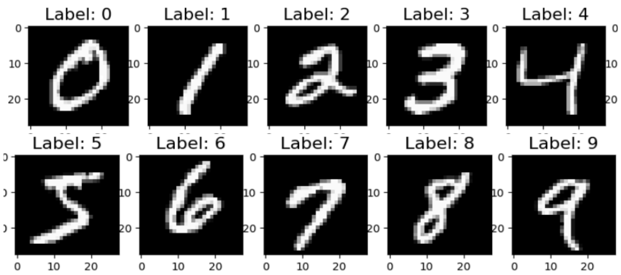
Diversity in Handwriting:The dataset contains handwritten digits from various individuals, leading to a significant variation in styles, sizes, and orientations of the digits.

Image Quality:The images are generally high quality, but variations in writing styles can affect the clarity and structure of the digits.

3.4 Preprocessing Challenges

There are no missing images in the dataset and the dataset is well-balanced so we didn't face any challenges.





4 Methodology:

4.1 Data Preprocessing Techniques

For the MNIST dataset, preprocessing involves a few essential steps:

- Normalization: Pixel values in the images, initially ranging from 0 to 255, are scaled to a range of 0 to 1 by dividing by 255. This scaling improves model convergence and stability.

- Reshaping: Each 28x28 image is converted into a single vector with 784 features, for CNN is reshaped to 28x28x1

- For MLP: Each 28x28 image is flattened into a single vector of 784 features to fit the fully connected structure of multilayer perceptrons (MLPs).

- For CNN: Each image is reshaped to 28x28x1 to match the expected input dimensions of convolutional layers.

- One-Hot Encoding: Labels are transformed into binary vectors, with each vector containing a "1" at the position corresponding to the class label, which is ideal for categorical classification.

No additional data cleaning is required, as MNIST is a curated, grayscale dataset of consistently formatted digit images.

4.2 Deep learning models

- Convolutional Neural Network (**CNN**) is a type of deep learning model designed for processing data like images. CNNs use convolutional layers to extract features and spatial patterns from the data by applying filters to sections of the image. They are effective for tasks such as image recognition and classification, often incorporating pooling layers to enhance performance and efficiency.

-A Multi-Layer Perceptron (**MLP**) is a type of deep learning model composed of multiple fully connected layers, where the first layer processes inputs and the final layer produces outputs. **MLPs** are used in supervised learning tasks, enabling pattern recognition and value inference from data using non-linear activation functions, making them effective for applications such as classification and prediction.

4.3 Evaluation metrics that is assessing deep learning models

Accuracy , Loss (Categorical Cross, Entropy Loss, Precision- Recall,F1 Score,Confusion Matrix

4.4 Experimental Setup

1. Hyperparameters

These are the hidden layers used in **CNN**

-Conv2D(32, (3, 3), activation='relu'): Applies 32 filters of size 3x3 to extract features.

-MaxPooling2D((2, 2)): Reduced dimensionality by taking the maximum value in each 2x2 block.

-Conv2D(64, (3, 3), activation='relu'): A second convolutional layer with 64 filters for deeper feature extraction.

-MaxPooling2D((2, 2)): Further down samples the feature maps.

-Flatten(): Converts 2D feature maps into a 1D vector.

-Dropout(0.5): Helps reduce overfitting by randomly stopping 50% of units during training.

-Dense(128, activation='relu'): A fully connected layer with 128 neurons for classification.

These are the hidden layers used in MLP

-Dense(128, activation='relu'): A layer with 128 neurons using the ReLU activation function for feature learning.

-Dense(64, activation='relu'): A layer with 64 neurons, also using ReLU, to refine the learned features.

-**(We used 2 in MLP)** Dropout(0.5): Helps reduce overfitting by randomly stopping 50% of units during training.

Batch size

We used a **batch size of 32** in both **CNN** and **MLP** architectures. This means that during training, the model processes 32 samples at a time before updating the weights, which can help balance training speed and memory usage

Learning rate

We used a **learning rate of 0.001** in **CNN** architecture. This choice often balances convergence speed and stability.

We used a **learning rate of 0.0005** in **MLP** architecture. This lower learning rate can enhance training stability and allow the model to converge more precisely, especially in cases where the dataset exhibits high variability

2. Training Epochs Both models were trained for a total of **five epochs**. Each epoch involved a complete pass through the training dataset, enabling the models to learn and adjust their parameters iteratively.

To further ensure model performance, we used 7 epochs to train the CNN. It appears that beyond this, additional epochs did not yield significant differences in results!

Data augmentation techniques

We did not employ data augmentation techniques in our work with the MNIST dataset. This choice was made to focus on evaluating the model's baseline performance using only the original dataset of 60,000 training images and 10,000 test images of handwritten digits.

4.5 Justification of Model Choice

(Multilayer Perceptron) MLP :

- Suitability for MNIST: MLPs can recognize patterns in the 28x28 pixel grayscale images, which have low spatial variability. By flattening the images into 784 features, MLPs can capture simple relationships without leveraging spatial hierarchies.
- Strengths in Variability: Effective for datasets with low spatial variability, such as MNIST, where digit shapes and positions are fairly consistent. However, MLPs are less effective on datasets that rely on spatial information.

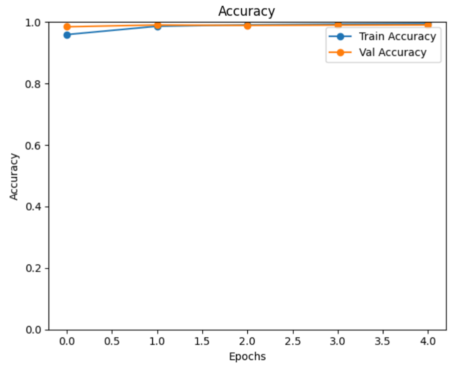
(Convolutional Neural Network) CNN:

- Suitability for MNIST: CNNs excel at identifying spatial patterns, such as edges and textures, which help in distinguishing handwritten digits. The architecture enables CNNs to efficiently capture local features in MNIST, improving accuracy.
- Strengths in Variability: CNNs are resilient to variations in images. This robustness makes CNNs preferable for datasets with moderate to high variability in local patterns.

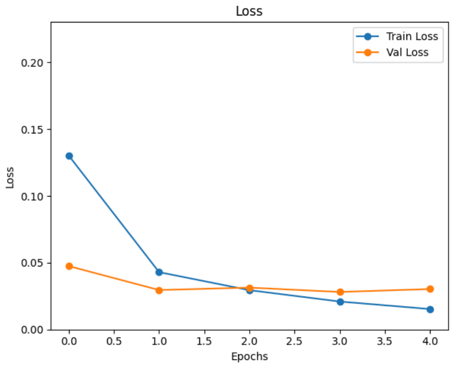
6. Patterns Discovery and Results

6.1 Deep learning model

CNN Model:



1. Figure:: Accuracy



2. Figure: Loss.

Figure (1) : **Accuracy**

Indicating that the model is generalising well without significant overfitting.

Figure (2) : **Loss**

Both train and validation loss decrease and stabilise, suggesting the model is learning effectively.

Overall, The plots suggest that the model is performing well, with high accuracy and low loss

Confusion Matrix

```
Confusion Matrix:
[[ 978    0    0    0    0    0    1    1    0    0]
 [    0 1131    2    1    0    0    1    0    0    0]
 [    1    0 1024    3    1    0    0    3    0    0]
 [    0    0    0 1008    0    1    0    0    1    0]
 [    0    0    0    0  976    0    2    0    2    2]
 [    0    0    0    3    0  887    1    0    0    1]
 [    4    2    0    0    1    4  947    0    0    0]
 [    0    3    5    2    0    0    0 1016    1    1]
 [    2    0    3    1    0    1    0    2  964    1]
 [    2    3    0    1    6    4    0    6    3  984]]
```

Key Findings

High Accuracy: The model shows very high accuracy (99.08%), correctly classifying the vast majority of digits.

Classification Report

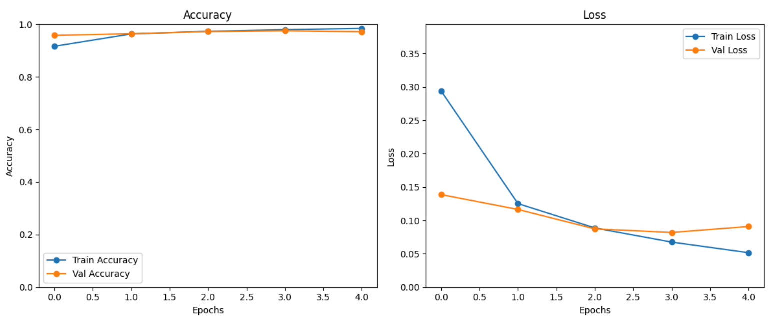
This indicates that the model performs excellently, with a few limited errors likely due to visual similarities between certain digits.

Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	980
1	0.99	1.00	0.99	1135
2	0.99	0.99	0.99	1032
3	0.99	1.00	0.99	1010
4	0.99	0.99	0.99	982
5	0.99	0.99	0.99	892
6	0.99	0.99	0.99	958
7	0.99	0.99	0.99	1028
8	0.99	0.99	0.99	974
9	0.99	0.98	0.98	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

The model performs exceptionally well across all metrics, with slight variations in precision and recall for a few classes (like 2 and 5) Overall, it achieves high accuracy, precision, recall, and F1-scores, indicating a robust classification performance across all digit classes.

MLP Model:



Accuracy Graph:
Both train and validation accuracy are high and converge, indicating good model performance without overfitting.
Loss Graph:
Both train and validation loss decrease and stabilise, suggesting the model is learning effectively.

The lines converging suggest that the model is not overfitting and is performing consistently across different data sets.

Confusion Matrix

```
Confusion Matrix:
[[ 968    0    0    2    0    3    4    1    2    0]
 [    0 1119    3    4    0    1    3    0    5    0]
 [    6    1  997    0    4    0    3    9   11    1]
 [    1    0   10  980    1    5    0    8    4    1]
 [    1    1    5    0  940    0    7    1    4   23]
 [    3    1    1   15    1  850    8    2   10    1]
 [    7    3    0    0    4    7  934    0    3    0]
 [    2   10   15    3    1    0    0  988    1    8]
 [    5    2    3    8    4    7    6    4  932    3]
 [    5    7    1   14   12    4    1    9    2  954]]
Classification Report:
```

The model performs well with an accuracy of 97.18%

Classification Report

```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       980
           1       0.98      0.99      0.98      1135
           2       0.96      0.97      0.96      1032
           3       0.96      0.97      0.96      1010
           4       0.97      0.96      0.96       982
           5       0.97      0.95      0.96       892
           6       0.97      0.97      0.97       958
           7       0.97      0.96      0.96      1028
           8       0.96      0.96      0.96       974
           9       0.96      0.95      0.95      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

The model performs consistently well across all classes, with slight variations.
High precision, recall, and F1-scores indicate robust performance.

6.2 Classical machine learning models

- A Decision Tree is a machine learning model used for classification and regression tasks. It organises data into a tree-like structure, where each node represents a decision based on a specific feature, and the leaf nodes represent the final outcomes. Decision Trees are known for their ease of understanding and interpretation.

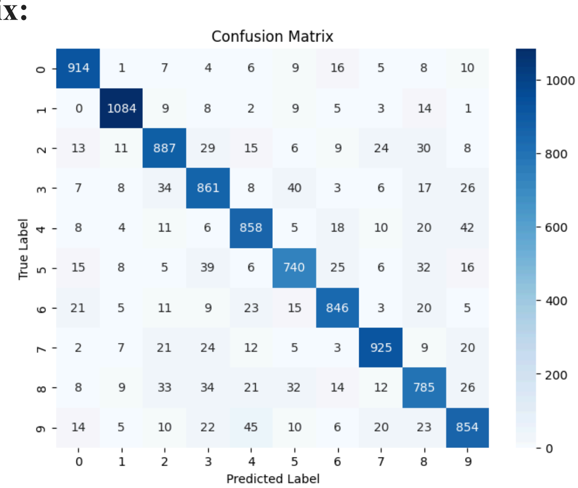
- A Support Vector Machine (SVM) is an algorithm used for classification by finding a hyperplane that separates different classes. SVM aims to maximise the margin between the closest data points from different classes, making it effective in high-dimensional spaces.

Decision Tree Classifier

	precision	recall	f1-score	support
0	0.91	0.93	0.92	980
1	0.95	0.96	0.95	1135
2	0.86	0.86	0.86	1032
3	0.83	0.85	0.84	1010
4	0.86	0.87	0.87	982
5	0.85	0.83	0.84	892
6	0.90	0.88	0.89	958
7	0.91	0.90	0.91	1028
8	0.82	0.81	0.81	974
9	0.85	0.85	0.85	1009
accuracy			0.88	10000
macro avg	0.87	0.87	0.87	10000
weighted avg	0.88	0.88	0.88	10000

Accuracy: 88% of predictions are correct.
Some classes, like 2 and 8, have room for improvement in precision and recall.
The model shows consistent but variable performance across classes.

Confusion Matrix:



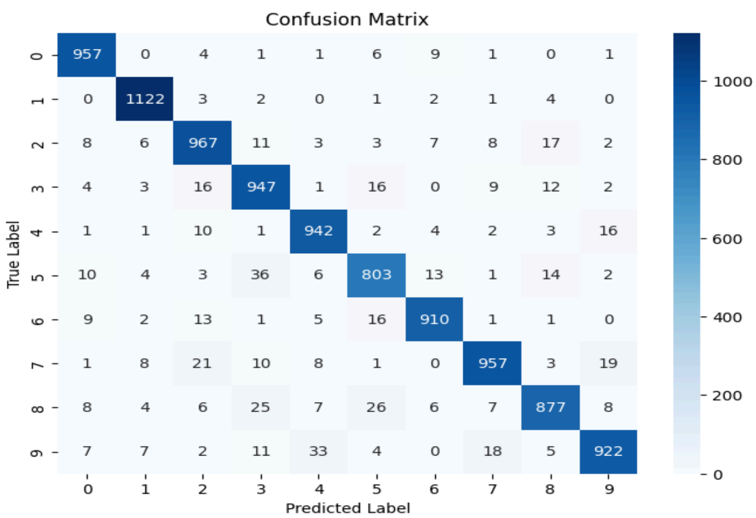
Performance Generally good with most classes having high correct predictions.

SVM model

Test Accuracy: 0.9404

	precision	recall	f1-score	support
0	0.95	0.98	0.96	980
1	0.97	0.99	0.98	1135
2	0.93	0.94	0.93	1032
3	0.91	0.94	0.92	1010
4	0.94	0.96	0.95	982
5	0.91	0.90	0.91	892
6	0.96	0.95	0.95	958
7	0.95	0.93	0.94	1028
8	0.94	0.90	0.92	974
9	0.95	0.91	0.93	1009
accuracy			0.94	10000
macro avg	0.94	0.94	0.94	10000
weighted avg	0.94	0.94	0.94	10000

Accuracy: 94.04% of predictions are correct.
Overall strong performance across all classes.



The model performs well with most predictions correct.

6.3 Deep Learning Models VS Classical Machine Learning:

Deep Learning	Accuracy	Machine Learning	Accuracy
CNN	99%	SVM	94%
MLP	97%	Decision Tree	88%

Based on the model's accuracy, it seems that the deep learning model, particularly the CNN (Convolutional Neural Network), has demonstrated superior performance.

7.Conclusion:

In this project, we explored the performance of two deep learning models on the MNIST dataset for handwritten digit classification. The results showed that the Convolutional Neural Network (CNN) was the most effective, achieving a high accuracy of 99%, making it the optimal choice for this task. Although the Multilayer Perceptron (MLP) model demonstrated good performance with notable accuracy, CNN outperformed it due to its superior ability to capture spatial patterns in images. This underscores the effectiveness of CNNs in complex visual data classification, highlighting their suitability for applications that require precise handling of image variability.