HANDWRITTEN DIGITS

PROJECT TEAM-ID - PTID-CDS-DEC-24-2145

PROJECT-ID - PRCP-1002

INTRODUCTION

The ability to recognize handwritten digits is a fundamental task in computer vision, with practical applications in postal code recognition, bank check processing, and form digitization. This project utilized the MNIST dataset, a widely used benchmark comprising 70,000 grayscale images of handwritten digits, each of size 28x28 pixels. The primary objective was to apply fundamental computer vision techniques and various classification methods to achieve high accuracy in digit recognition.

DATASET DESCRIPTION

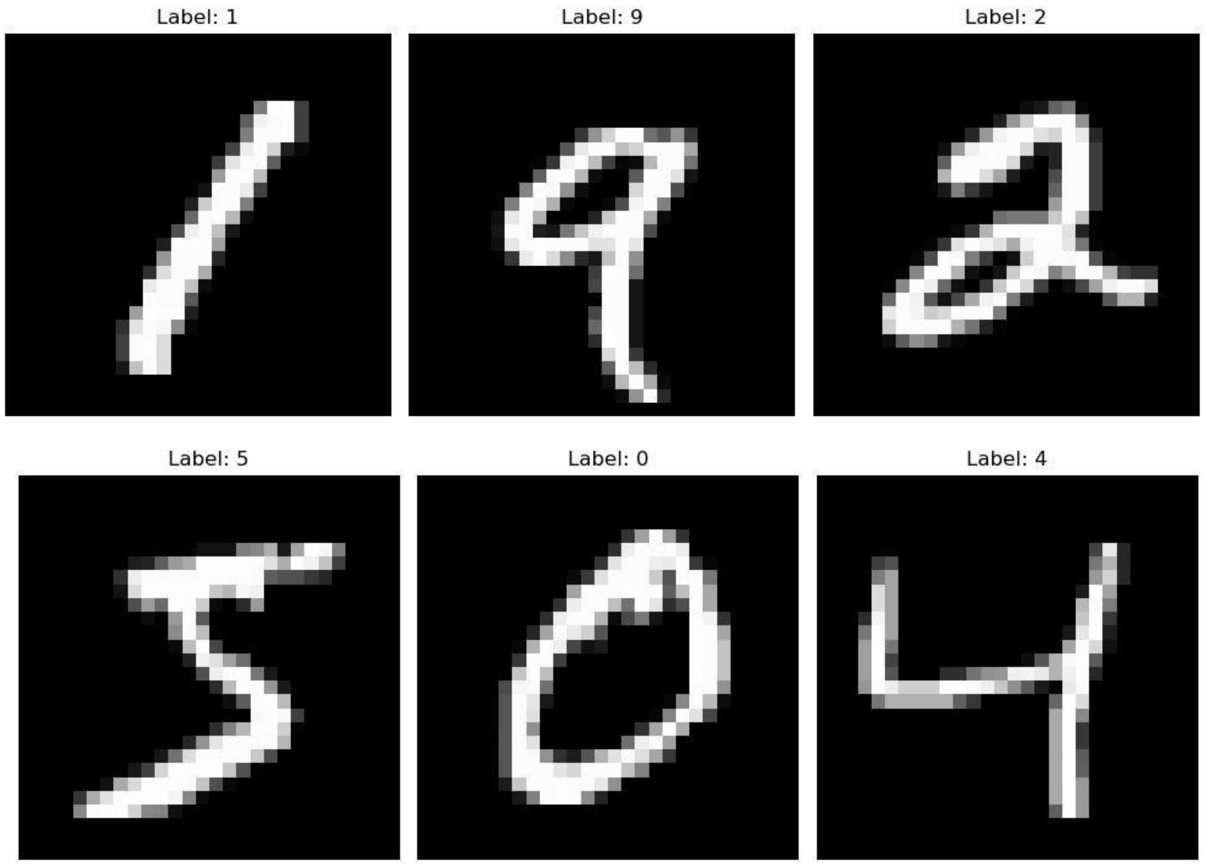
The MNIST dataset, containing 70,000 grayscale images of handwritten digits, was used for this project. Each image is:

- Size: 28x28 pixels
- Color: Grayscale (pixel values range from 0 to

255)

- Data Split: 60,000 training images and 10,000 testing images

The images are accompanied by corresponding labels, representing the digit depicted.



CHALLENGES FACED ON DATA

- Data Imbalance: some digits had slightly fewer samples compared to others, which could have led to biased model performance.
- Certain handwritten digits were poorly written or ambiguous, making classification challenging.
- High Dimensionality: each image had 784 features (28x28 pixels), increasing computational complexity for models like KNN and SVM.
- Overfitting Risk: deep learning models like CNNs will be overfitting due to their complexity, especially with limited preprocessing.

TECHNIQUES USED

- Data Normalization
- Dropout Regularization: applied in the CNN architecture to prevent overfitting by randomly deactivating neurons during training. This encouraged the network to learn robust features rather than memorizing the training data.
- Use of RBF Kernel: the RBF kernel in SVM was chosen for its ability to handle non-linear data effectively. The transformed the feature space to make classes linearly separable, improving classification accuracy.

SYSTEMATIC METHODS USED

- Import libraries
- Load data
- Data preparation
- Normalization
- Re-shaping
- Model creation
- Preprocessing

LOADING AND PREPARING THE DATA

Divide the dataset into three subsets: train, test, and validation, using the split-folders library.

DATA PREPARATION

In this step, we will generate batches for both the training and validation datasets while ensuring the images are properly pre-processed. This involves preparing the data in a way that makes it suitable for feeding into the model. By batching the data, we can efficiently manage memory usage and optimize the training process.

NORMALIZATION

In this step the scale pixel values to a uniform range of 0 to 1 for better model convergence and stability and it is implemented by divided each pixel value (0– 255) by 255 to normalize the data.

RE-SHAPING

The neural network flattened each 28x28 pixel image into a 1D vector of size 784 for input into dense neural network layers. So that maintained the 1D vector format for compatibility with models like SVM and KNN.

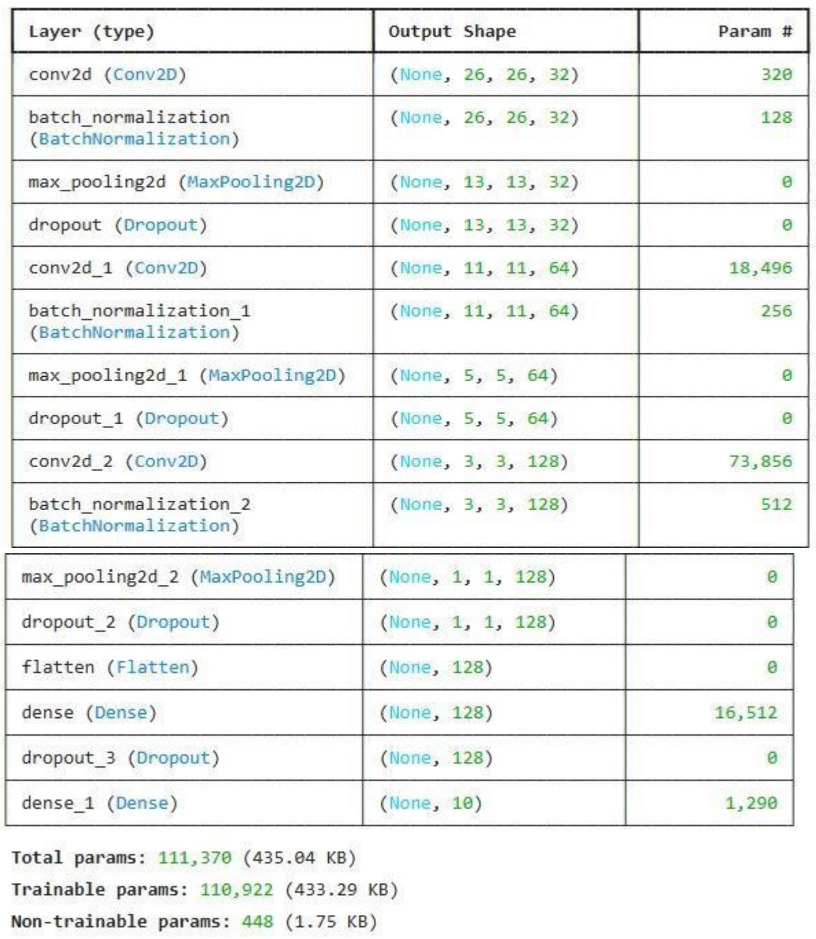
MODEL CREATION

- **Convolutional Neural Networks (simple neural network)**

It is a type of deep learning model specifically designed for processing grid-like data such as images. They use convolutional layers to automatically extract features like edges, textures, and shapes from raw input images. These features are then used to make predictions, making CNNs highly effective for image classification tasks.

In this model two convolutional layers with ReLU activation and max pooling and fully connected dense layer with with dropout for regularization also softmax activation is being used in the output layer.

MODEL SUMMARY



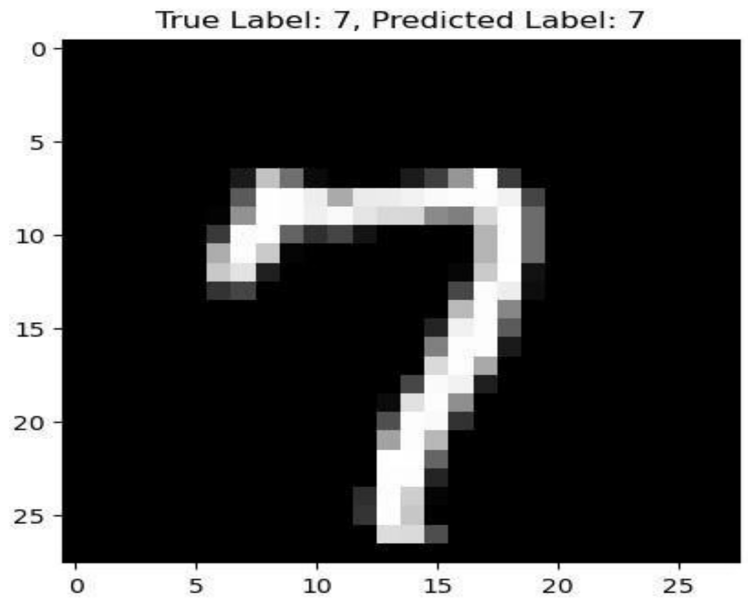
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
batch_normalization (BatchNormalization)	(None, 26, 26, 32)	128
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
dropout (Dropout)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 11, 11, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_1 (Dropout)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 3, 3, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 128)	0
dropout_2 (Dropout)	(None, 1, 1, 128)	0
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 128)	16,512
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

```
Total params: 111,370 (435.04 KB)
Trainable params: 110,922 (433.29 KB)
Non-trainable params: 448 (1.75 KB)
```

MODEL EVALUATION

Here the model accuracy is 99.43% and 1.67% loss also training time is about moderate with 10 epoches. The model test accuracy is 98.83% and 4.61% loss.

DISPLAYING IMAGE AND PREDICTION



CLASSIFICATION METHODS

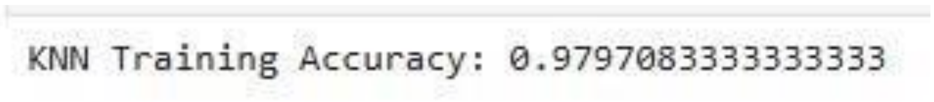
- **K-Nearest Neighbors (KNN)**

KNN is a simple, non-parametric algorithm that classifies data points based on the majority class of their nearest neighbors in the feature space. It is easy to implement and works well for smaller datasets. However, KNN can be computationally expensive for large datasets due to its reliance on distance calculations during inference.

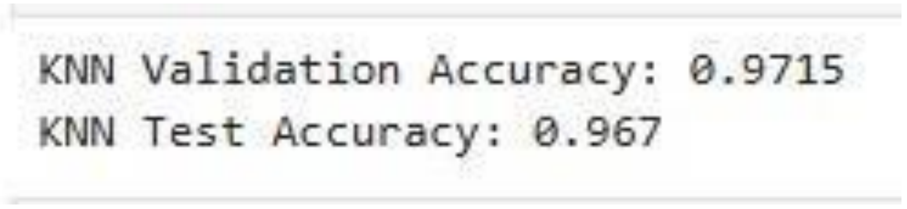
In this model the KNN uses Euclidean distance as distance metric with an optimal number of neighbors of five.

MODEL EVALUATION

Here the model got an training accuracy 97.9%, validation accuracy of 97.1% and test accuracy 96.7% here training time is tend to slow on large datasets.



```
KNN Training Accuracy: 0.9797083333333333
```



```
KNN Validation Accuracy: 0.9715
KNN Test Accuracy: 0.967
```

- **Support Vector Machine(SVM)**

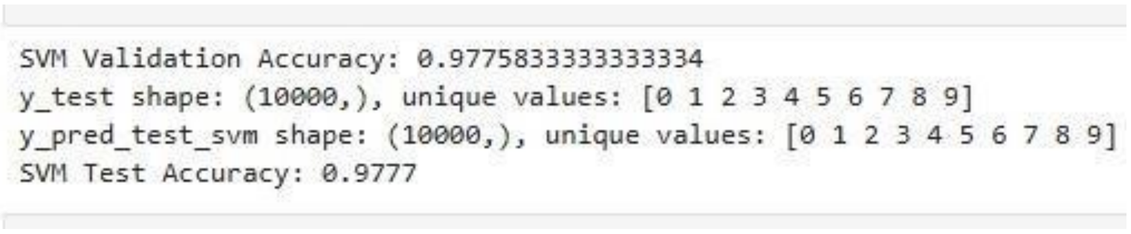
Support Vector Machines (SVM) are supervised learning models that classify data by finding the optimal hyperplane that separates different classes. SVMs are effective for small to medium-sized datasets and excel in handling non-linear data when used with kernel functions like the Radial Basis

Function (RBF). They are robust to overfitting and can achieve high accuracy, but their computational cost makes them less suitable for large datasets.

In this the model uses Radial Basis Function (RBF) as kernel as it is used because it effectively handles nonlinear data by transforming it into a higher dimensional space, enabling better separation of classes. It provides a good balance between flexibility and computational efficiency for datasets with complex patterns.

MODEL EVALUATION

Here the model attains a validation accuracy of 97.75% and test accuracy 97.7%. It also have a high training speed compared to others.


```
SVM Validation Accuracy: 0.9775833333333334
y_test shape: (10000,), unique values: [0 1 2 3 4 5 6 7 8 9]
y_pred_test_svm shape: (10000,), unique values: [0 1 2 3 4 5 6 7 8 9]
SVM Test Accuracy: 0.9777
```

- **Decision Tree**

Decision Tree Classifiers are tree-structured models that split data based on feature thresholds to maximize class separation. They are simple to interpret and can handle non-linear relationships effectively.

MODEL EVALUATION

In this model the decision tree attains a training accuracy 1.0, validation accuracy 86.7% and test accuracy 87.5%

COMPARISON

Model	Accuracy(%)	Training time	Inference speed	Scalability
CNN	99.4	Moderate	Fast	High
KNN	97.9	Fast	Slow	Low
SVM(rbf)	97.7	High	Moderate	Medium
Decision Tree	86.7	Fast	Fast	Medium

RESULT

Based on the evaluation, the Convolutional Neural Network (CNN) is the best model for production deployment due to its:

- Superior accuracy (99.4%)

- Scalability to handle larger datasets
- Fast inference

While SVM with the RBF kernel also performs well, its computational demands and scalability limitations make it less suitable for production. KNN, despite its simplicity, lacks the accuracy and efficiency needed for large-scale applications.

CONCLUSION

For deploying a handwritten digits recognition system in production, the CNN model is recommended. Its combination of high accuracy, scalability, and efficient inference ensures reliable performance in real-world scenarios. Further improvements, such as optimizing the CNN architecture or incorporating transfer learning, could enhance the model's performance even more.