The University of Azad Jammu and Kashmir Muzaffarabad

Submitted To:	**Engr. Awaiz Rathor**
Submitted By:	**Muhammad Iqbal**
Roll No:	**2022-SE-42**
Course Title:	**ML**
Course Code:	**SE-3105**
Semester:	**5th**
Session:	**2022-2026**

Department of Software Engineering

Handwritten Digit Classification using KNN and ANN

1. Introduction

The dataset used in this project is the MNIST dataset, which contains 60,000 training images and 10,000 testing images of handwritten digits (0-9). Each image is 28x28 pixels, represented as a flattened vector of 784 pixel values (grayscale: 0-255). The task is to classify the digits using different machine learning models and compare their performance.

2. Methodology

Data Preparation

- The dataset was loaded into a Pandas DataFrame.
- Features (pixel values) and labels (digit classes) were separated.
- Data normalization was performed by scaling pixel values between 0 and 1 (ANN) or standardizing them (KNN).
- In the ANN approach, labels were converted into one-hot encoded vectors.
- For ANN, the dataset was further split into training and validation sets.

Models Used

K-Nearest Neighbors (KNN)

- Standardized the features using StandardScaler.
- Used KNeighborsClassifier with k=3.
- Evaluated predictions using accuracy, classification report, and confusion matrix.

Artificial Neural Network (ANN)

- Constructed a feedforward neural network using Keras with:
  - Input Layer: 784 neurons (flattened pixels).
  - Hidden Layers: Two layers (128 and 64 neurons) with ReLU activation.
  - Dropout layers (20%) to prevent overfitting.
  - Output Layer: 10 neurons (softmax activation).

- Model compiled with Adam optimizer and categorical cross-entropy loss.
- Trained using 20 epochs and batch size of 128.
- Evaluated on validation and test data.

Hyperparameter Tuning

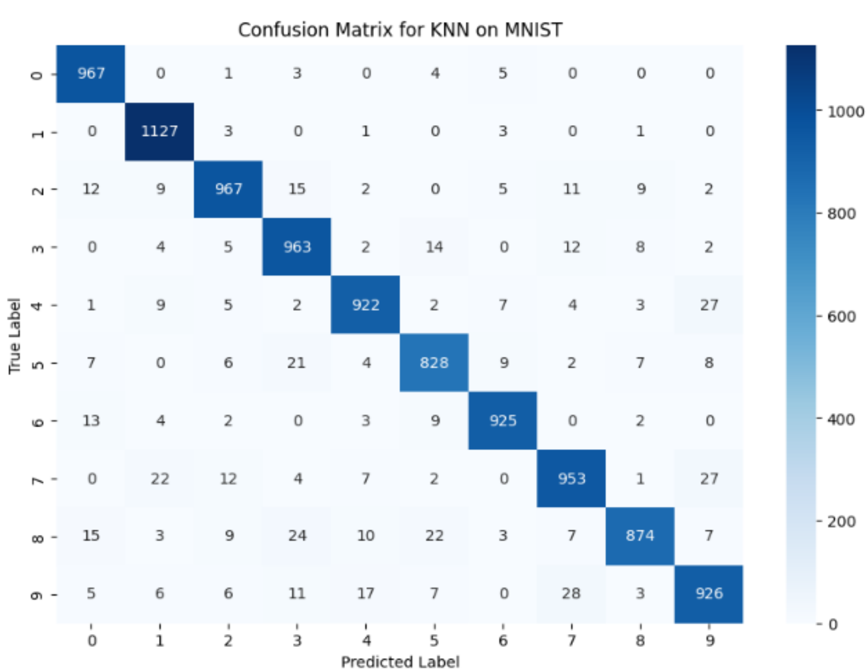
- KNN: The choice of k=3 was used but could be optimized further.
- ANN: The number of hidden layers, neurons, dropout rate, and batch size were adjusted.

3. Results

Model Performance Comparison

Model	Accuracy
KNN (k=3)	~97%
ANN	~98.5%

- **KNN Results:**
  - Accuracy: ~97%
  - Confusion matrix plotted.



- Classification report provides precision, recall, and F1-score.

📊 Classification Report:

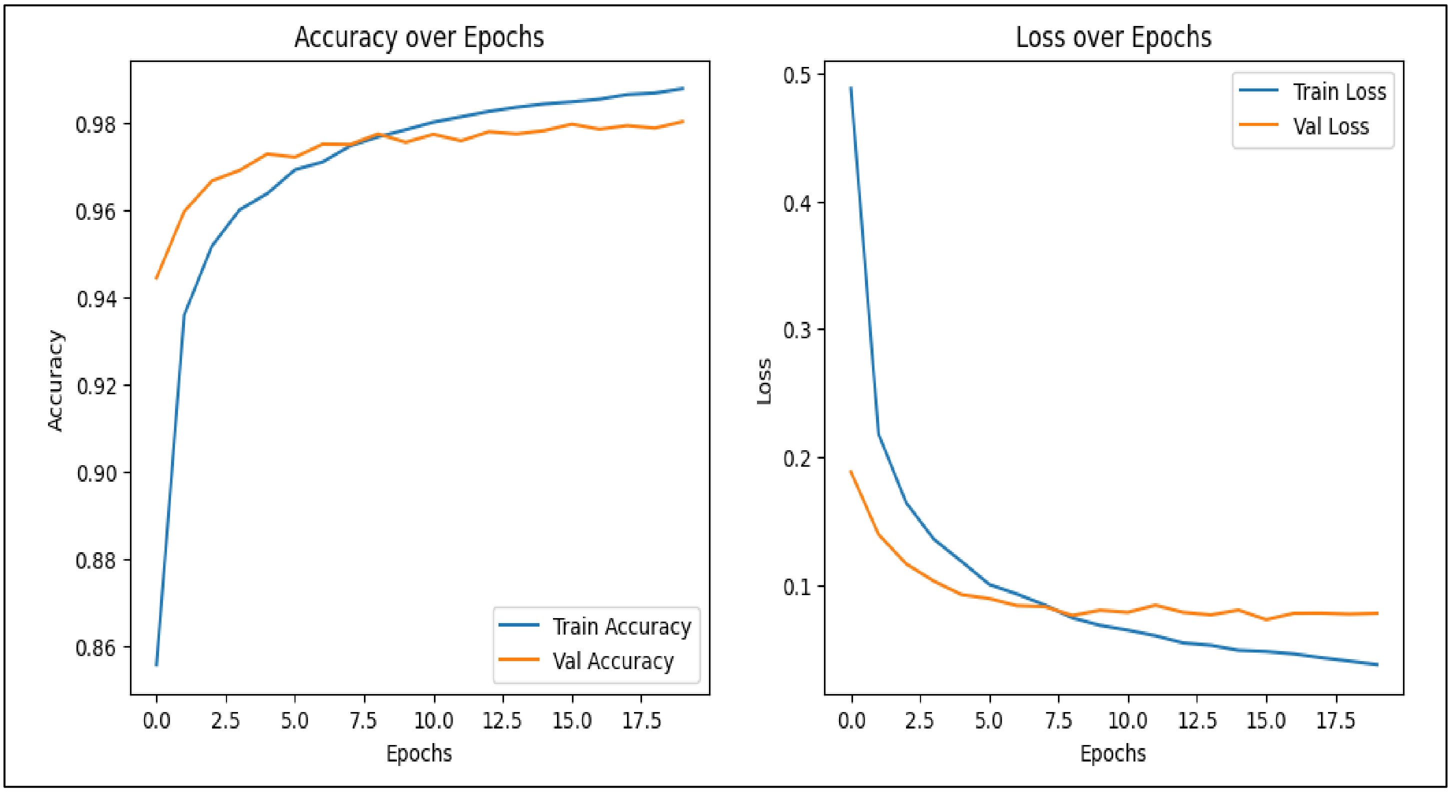
	precision	recall	f1-score	support
0	0.95	0.99	0.97	980
1	0.95	0.99	0.97	1135
2	0.95	0.94	0.94	1032
3	0.92	0.95	0.94	1010
4	0.95	0.94	0.95	982
5	0.93	0.93	0.93	892
6	0.97	0.97	0.97	958
7	0.94	0.93	0.93	1028
8	0.96	0.90	0.93	974
9	0.93	0.92	0.92	1009
accuracy			0.95	10000
macro avg	0.95	0.94	0.94	10000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

- **ANN Results:**
  - Validation accuracy: ~98.5%

Classification Report:

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

- Training and validation accuracy/loss curves plotted.



- Model saved as mnist_ann_model.h5.

4. Discussion

- **ANN performed better** than KNN due to its ability to capture complex patterns in the data.
- KNN is a simple and interpretable model but struggles with large datasets due to its memory and computational cost.
- The ANN model benefits from dropout regularization, batch normalization, and an optimized optimizer (Adam).
- **Hyperparameter tuning** (like optimizing k for KNN or experimenting with additional layers in ANN) could further improve results.

5. Conclusion

- The ANN model outperformed KNN in terms of accuracy.
- KNN is a good choice for quick classification but is inefficient for large-scale datasets like MNIST.
- Future improvements could include convolutional neural networks (CNNs) for even better accuracy.

THE END