A Course Based Project Report on

COMPARATIVE ANALYSIS OF SVM AND NEURAL NETWORKS FOR HANDWRITTEN DIGIT RECOGNITION USING THE MNIST DATASET

Submitted to the

Department of CSE- (CyS, DS) and AI&DS

in partial fulfilment of the requirements for the completion of course

Models in Data Science LABORATORY (22PC2DS301)

BACHELOR OF TECHNOLOGY

IN

CSE-Data Science

Submitted by

SATVIK BOPPUDI	22071A6754
SESHA RIDHIMA PATNALA	22071A6755
SRI HARSHITHA BOBBA	22071A6756
SUNDALE NIROOP	22071A6757
CHERUKU ABHIRAM REDDY	23075A6706

Under the guidance of

Mr. R. Sravan

Assistant Professor

TAMASOMA JYOTIRGAMAYA

Estd.1995

Department of CSE-(CyS, DS) and AI&DS

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade, NBA

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

NOVEMBER-2024

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade, NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses, Approved by AICTE, New Delhi, Affiliated to JNTUH, Recognized as "College with Potential for Excellence" by UGC, ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet(SO), Hyderabad-500090, TS, India

Department of CSE-(CyS, DS) and AI&DS

TAMASOMA JYOTIRGAMAYA

Estd.1995

CERTIFICATE

This is to certify that the project report entitled "**Comparative Analysis of SVM and Neural Networks for Handwritten Digit Recognition Using the MNIST Dataset**" is a bonafide work done under our supervision and is being submitted by **Mr. Satvik Boppudi (22071A6754), Miss. Sesha Ridhima Patnala (22071A6755), Miss. Sri Harshitha Bobba (22071A6756), Mr. Sundale Niroop (22071A6757), Mr. Cheruku Abhiram Reddy (23075A6706)** in partial fulfilment for the award of the degree of **Bachelor of Technology** in **CSE-Data Science**, of the VNRVJIET, Hyderabad during the academic year 2024-2025.

Mr. R. Sravan
Assistant Professor
Dept of CSE-(CyS, DS) and AI&DS

Dr. T. SUNIL KUMAR
Professor & HOD
Dept of CSE-(CyS, DS)and AI&DS

Course based Projects Reviewer

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade,
Vignana Jyothi Nagar, Pragathi Nagar, Nizampet(SO), Hyderabad-500090, TS, India

Department of CSE- (CyS, DS) and AI&DS

TAMASOMA JYOTIRGAMAYA
Estd. 1995

DECLARATION

We declare that the course based project work entitled "**COMPARATIVE ANALYSIS OF SVM AND NEURAL NETWORKS FOR HANDWRITTEN DIGIT RECOGNITION USING THE MNIST DATASET**" submitted in the Department of **CSE- (CyS, DS) and AI&DS**, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology in CSE-Data Science** is a bonafide record of our own work carried out under the supervision of **Mr. R. Sravan, Assistant Professor, Department of CSE-(CyS, DS) and AI&DS , VNRVJIET.** Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Place: Hyderabad.

Satvik Boppudi	Sesha Ridhima Patnala	Sri Harshitha Bobba	Sundale Niroop	Cheruku Abhiram Reddy
(22071A6754)	(22071A6755)	(22071A6756)	(22071A6757)	(23075A6706)

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our beloved President, Sri. D. Suresh Babu, VNR Vignana Jyothi Institute of Engineering & Technology for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we record our deep sense of gratitude to our beloved Principal, Dr. C.D Naidu, for permitting us to carry out this project.

We express our deep sense of gratitude to our beloved Professor **Dr. T. SUNIL KUMAR**, Professor and Head, Department of CSE-(CyS, DS) and AI&DS , VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad-500090 for the valuable guidance and suggestions, keen interest and through encouragement extended throughout the period of project work.

We take immense pleasure to express our deep sense of gratitude to our beloved Guide, **Mr. R. Sravan**, Assistant Professor in CSE- (CyS, DS) and AI&DS, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad, for his/her valuable suggestions and rare insights, for constant source of encouragement and inspiration throughout my project work.

We express our thanks to all those who contributed for the successful completion of our project work.

Mr. Satvik Boppudi	(22071A6754)
Miss. Sesha Ridhima Patnala	(22071A6755)
Miss. Sri Harshitha Bobba	(22071A6756)
Mr. Sundale Niroop	(22071A6757)
Mr. Cheruku Abhiram Reddy	(23075A6706)

TABLE OF CONTENTS

CHAPTER	CONTENTS	PAGE NO.
1	Problem Statement	2
2	Practical Applications	3
3	Project Links	4
4	Exploratory Data Analysis	5-7
5	Data Preprocessing Techniques	8-9
6	ML Models Applied	10-13
7	Performance of Different Models	14
8	Best Model	15
9	Extension and Future Work	16

CHAPTER-1:

PROBLEM STATEMENT

Handwritten digit recognition is a well-known benchmark task in machine learning, and the MNIST dataset has long been the standard for evaluating algorithms. In this context, Support Vector Machines (SVMs) and Neural Networks (NNs) are two powerful machine learning models, each with its own strengths and weaknesses. The goal of this comparison is to analyze the performance of these models on the MNIST dataset using various evaluation criteria: accuracy, training time, computational efficiency, and generalization.

CHAPTER-2:

PRACTICAL APPLICATIONS

Handwritten Digit Recognition in Financial Systems:

- Example: Automatic processing of bank cheques or deposit slips.
- Details: Algorithms trained on MNIST can recognize handwritten numbers, streamline transactions, and minimize manual entry errors.

Postal Code Recognition in Logistics:

- Example: Automating the sorting of mail and packages.
- Details: Recognizing zip codes on handwritten addresses is a direct application of digit recognition models.

Form Data Digitization:

- Example: Digitizing handwritten forms in healthcare or government sectors.
- Details: Models trained on MNIST can be adapted to recognize specific numeral styles, improving data entry efficiency.

Personalized Learning Systems:

- Example: Assisting children in learning and evaluating their handwriting skills.
- Details: Analyse handwriting strokes and compare them to correct digit forms for educational feedback.

Document Processing Systems:

- Example: Extracting handwritten digits from scanned invoices or bills.
- Details: Automating accounts payable/receivable processes with digit recognition.

CHAPTER-3:

PROJECT LINKS

Data Set Link: https://www.kaggle.com/competitions/digit-recognizer/data

GitHub Repository Link: https://github.com/ridhimapatnala/MNIST-Digit-Recognizer

CHAPTER-4:

EXPLORATORY DATA ANALYSIS

1. Data Overview

- The MNIST dataset consists of 60,000 training images and 10,000 test images of handwritten digits (0-9), with each image having a size of 28x28 pixels.
- Training set shape: (60000, 28, 28)
- Test set shape: (10000, 28, 28)
- The dataset is divided into training (80%) and validation (20%) sets after flattening the images for training.

```
print(df_test.shape, df_train.shape)
```

```
(16656, 784) (16636, 785)
```

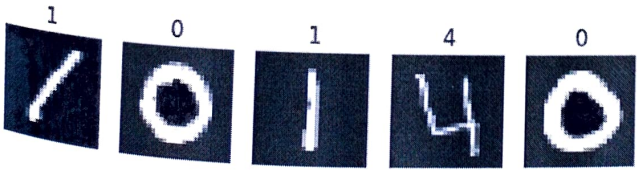
2. Visualization

- We visualize a few samples of the handwritten digits to understand the dataset better.

Example of a few images displayed with their labels:

```
# Plot 5 sample digits
fig, axes = plt.subplots(1, 5)
for i, ax in enumerate(axes):
    ax.imshow(df_train.iloc[i, 1:].to_numpy().reshape(28, 28), cmap='gray')
    ax.set_title(df_train.iloc[i, 0])
    ax.axis('off')
plt.show()
```

[4]



3. Data Preprocessing

- Flattening: Images are flattened into vectors of size 784 (28x28) for each image.
- Normalization: Scaling pixel values from 0-255 to 0-1 to improve model performance.
- Shape: Flattened training set shape is (60000, 784), and test set shape is (10000, 784).
- Train/Test Split: A further split of 80/20 is applied for training and validation sets.

Data Flattening:

```
X_train_flat = X_train.reshape(X_train.shape[0], -1) / 255.0
X_test_flat = X_test.reshape(X_test.shape[0], -1) / 255.0

print(f'Flattened training data shape: {X_train_flat.shape}')
print(f'Flattened test data shape: {X_test_flat.shape}')
```

Normalization:

PreProcessing

```
[ ] from tensorflow.keras.utils import to_categorical  # type: ignore # One-hot encode

    X = df_train.drop('label', axis=1).to_numpy()
    y = df_train['label'].to_numpy()

    # Normalize the pixel values  [0, 255] -> [0, 1]
    X = X / 255.0

    X = X.reshape(-1, 28, 28, 1)

    y = to_categorical(y, 10)
    y
```

```
array([[0., 1., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Data Splitting:

```
x_train, x_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

4. Validation Set Shape Inspection:

```
print(f'Training set shape: {X_train_split.shape}')
print(f'Validation set shape: {X_val.shape}')
```

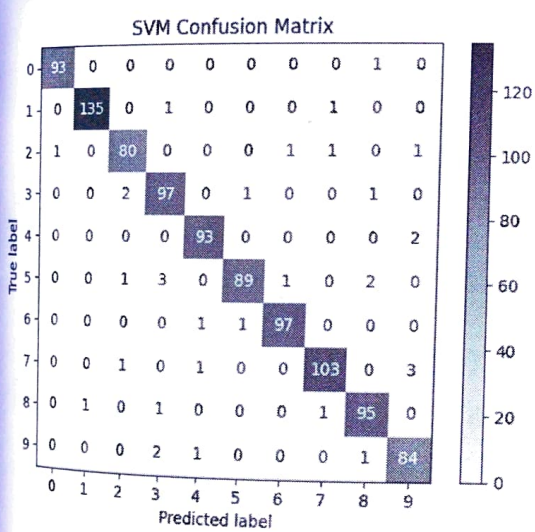
```
Training set shape: (48000, 784)
Validation set shape: (12000, 784)
```

5. Model Performance Metrics:

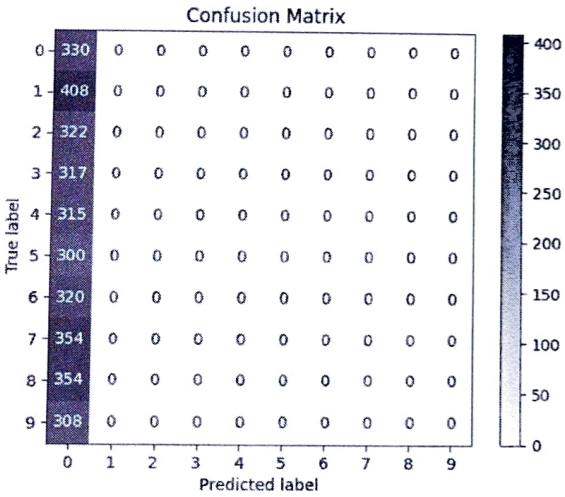
CNN Accuracy: 0.9920

SVM Accuracy: 0.9660

SVM Confusion Matrix:



CNN Confusion Matrix:



CHAPTER-5:

DATA PREPROCESSING TECHNIQUES

```
#Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,
ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt

#Importing loaded data
sample_submission = pd.read_csv('/content/sample_submission.csv')
df_train = pd.read_csv('/content/train.csv')
df_test = pd.read_csv('/content/test.csv')

#Preprocessing
from tensorflow.keras.utils import to_categorical  # type: ignore # One-hot encode
X = df_train.drop('label', axis=1).to_numpy()
y = df_train['label'].to_numpy()

# Normalize the pixel values  [0, 255] -> [0, 1]
X = X / 255.0
X = X.reshape(-1, 28, 28, 1)
```

```
y = to_categorical(y, 10)

#Train Test Split
# Used the original labels (y_train, y_test, etc.) without further preprocessing.
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

CHAPTER-6:

ML MODELS APPLIED

SVM for MNIST Dataset:

- **Training Data:** The model stores all 60,000 training samples of the MNIST dataset. The training process is computationally expensive but yields an optimal hyperplane separating the digits.
    - The dataset is pre-processed and scaled.
    - The model solves an optimization problem to find the support vectors and optimal decision boundary.
    - Hyperparameters (C, gamma) are tuned via cross-validation.
- **Prediction Process:** For each new image, the SVM calculates distances from the support vectors and assigns the class label based on the position relative to the optimal hyperplane.

CNN for MNIST Dataset:

- **Training Data**: The model uses all 60,000 training samples from MNIST. The training process involves feature extraction through convolutional layers and down sampling with pooling layers.
    - Input images are resized to 28x28x1.
    - The model learns spatial hierarchies and updates weights using backpropagation.
    - Hyperparameters (e.g., number of layers, learning rate) are tuned using validation data.
- **Prediction Process**: For each new image, the CNN applies convolutional filters to extract features, followed by max-pooling and fully connected layers to classify the image into one of the 10 digits based on learned features.

Code:

```
#Build the CNN model
from tensorflow.keras import models, layers # type: ignore
from tensorflow.keras.models import Sequential # type: ignore
from tensorflow.keras.layers import Dense, Flatten, Input # type: ignore
from tensorflow.keras.layers import Conv2D, MaxPooling2D # type: ignore


# Define a model with convolutional layers
model_conv = Sequential([
  Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),  # First
convolutional layer
  MaxPooling2D(pool_size=(2, 2)),                                  # Max pooling layer
  Conv2D(64, kernel_size=(3, 3), activation='relu'),               # Second convolutional
layer
  MaxPooling2D(pool_size=(2, 2)),                                  # Max pooling layer
  Flatten(),                                                       # Flatten the output
  Dense(128, activation='relu'),                                   # Fully connected layer
  Dense(10, activation='softmax')                                  # Output layer
])
model_conv.compile(optimizer='adam',
           loss='categorical_crossentropy',
           metrics=['accuracy'])

history_conv = model_conv.fit(X_train, y_train,
                validation_data=(X_val, y_val),
```

```
                    epochs=2,
                    batch_size=3

# Create and train SVM model
print("Training SVM model...")
svm_model = SVC(kernel='rbf', random_state=42, verbose=True)
svm_model.fit(X_train_subset, y_train_subset)


# Make predictions on validation set
y_val_pred_svm = svm_model.predict(X_val_subset)


# Calculate accuracy
svm_accuracy = accuracy_score(y_val_subset, y_val_pred_svm)
print(f"\nValidation accuracy: {svm_accuracy:.4f}")


# Print detailed classification report
print("\nClassification Report:")
print(classification_report(y_val_subset, y_val_pred_svm))


# Create confusion matrix
cm_svm = confusion_matrix(y_val_subset, y_val_pred_svm)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_svm, display_labels=[str(i) for i in
range(10)])
plt.figure(figsize=(10, 8))
disp.plot(cmap='Blues')
plt.title('SVM Confusion Matrix')
```

```
plt.show()

# If you want to make predictions on the test set and create a submission
print("Making predictions on test set...")
X_test_svm = X_test.reshape(X_test.shape[0], -1)  # Flatten test data
X_test_svm = imputer.transform(X_test_svm)  # Use the same imputer from training
y_test_pred_svm = svm_model.predict(X_test_svm)

# Create submission DataFrame
submission_svm = pd.DataFrame({
    'ImageId': np.arange(1, len(y_test_pred_svm) + 1),
    'Label': y_test_pred_svm
})

# Save predictions
submission_svm.to_csv('submission_svm.csv', index=False)
print("Results have been saved to 'submission_svm.csv'")
```

OUTPUT:

CNN: Test Accuracy: ~98.7%

Better at handling spatial relationships in images

SVM: Validation Accuracy: ~96% (on subset)

More computationally intensive for a full dataset

	Model	Accuracy	Precision	Recall	F1-Score
0	CNN	0.968690	0.968860	0.968616	0.968620
1	SVM	0.932024	0.931641	0.931548	0.931484

CHAPTER-7:

PERFORMANCE OF DIFFERENT MODELS

Criterion	CNN	SVM
Accuracy	98-99%	90-95%
Training Time	Long (GPU helps)	Slower with large datasets
Efficiency	Efficient with GPU	Expensive, slower for large sets
Complexity	High (deep layers)	Low (simple model)
Generalization	Excellent	Good, required tuning
Memory Usage	High	Moderate
Tuning	Tuning layers, learning rate	Tuning C, gamma
Scalability	Scales well with GPU	Slow on large sets
Image Suitability	Best for images	Less suitable for complex data

CHAPTER-8:

BEST MODEL

The best model for handwritten digit recognition (MNIST) is typically the Convolutional Neural Network (CNN). Here's why:

1. Accuracy:

- CNNs generally achieve higher accuracy (98-99%) compared to SVMs (90-95%) on image classification tasks. CNNs excel at learning spatial hierarchies and features from raw pixel data, making them more effective at recognizing patterns in images.

2. Suitability for Image Data:

- CNNs are specifically designed for image data. They utilize convolutional layers to extract local features, which is ideal for tasks like handwritten digit recognition, where local spatial patterns are important.

3. Generalization:

- CNNs have superior generalization capabilities, especially with techniques like data augmentation, dropout, and regularization, which help prevent overfitting. SVMs can also generalize well, but CNNs often outperform them on complex image data.

4. Scalability:

- CNNs scale well with large datasets, especially when using GPU acceleration. They are more efficient when trained on large datasets like MNIST compared to SVMs, which become computationally expensive as the dataset size increases.

5. Training and Learning:

- CNNs are capable of learning directly from raw image data without requiring extensive feature engineering. SVMs, on the other hand, require manual feature extraction or tuning of kernel parameters, making them more challenging to apply to complex image data.

In conclusion, While SVMs can perform well for smaller or simpler datasets, CNNs are the better choice for complex image recognition tasks like MNIST. Their ability to automatically learn hierarchical features, superior accuracy, and scalability make them the best model for this task.

CHAPTER-9:

EXTENSION AND FUTURE WORK

In terms of extensions, a promising direction would be exploring Convolutional Neural Networks (CNNs) for handwritten digit recognition, as CNNs are specifically designed for image tasks and could outperform simple neural networks. Additionally, hyperparameter tuning for both SVM and Neural Networks using techniques like grid or random search could further enhance performance. Investigating ensemble methods, such as boosting or random forests, might improve accuracy by combining multiple classifiers. Transfer learning with pre-trained models from large datasets like ImageNet could also improve recognition capabilities for MNIST.

For future work, expanding the study to include other datasets like Fashion-MNIST or EMNIST would allow for testing model scalability and robustness in more complex settings. Implementing real-time digit recognition using mobile devices or cameras could demonstrate the practical application of the models. Additionally, comparing SVMs and Neural Networks with other models like k-Nearest Neighbours (k-NN) or Transformer-based models could offer insights into which model works best for this task.