A PROJECT REPORT

on

Development and implementation of artificial intelligence (AI) models to accurately identify and classify handwritten digits

SUBMITTED TO

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

In partial fulfilment of the award of the course of

CSA1790- ARTIFICIAL INTELLIGENCE FOR

By

N.V. Tharun kumar (192210423)

K. Sai Nithin (192210551)

SUPERVISOR

Dr. A. JAYA MABEL RANI

SAVEETHA SCHOOL OF ENGINEERING,
SIMATS, CHENNAI-602105
MARCH-2024

BONAFIDE CERTIFICATE

This is to certify that the project report Development and implementation of artificial intelligence (AI) models to accurately identify and classify handwritten digits entitled submitted by N.V. THARUN KUMAR (192210423) and K.SAI NITHIN (192210551) , to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of Bonafide work carried out by him/her under my guidance. The project fulfils the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

DR.A. JAYA MABELRANI
COURSE FACULTY
Institute of CSE,
Saveetha School of Engineering,
SIMATS, Chennai -602105.

TABLE OF CONTENTS

ABSTRACT:

Handwritten digit recognition remains a cornerstone task in the fields of computer vision and machine learning, pivotal for various applications such as automated postal sorting, bank check processing, and digital form entry. This study delves into the development and implementation of a robust Convolutional Neural Network (CNN) to accurately recognize handwritten digits using the widely acknowledged MNIST dataset. The dataset comprises 60,000 training images and 10,000 test images of digits ranging from 0 to 9, providing a comprehensive basis for model training and evaluation.

The preprocessing phase involves normalizing the pixel values to fall between 0 and 1 and reshaping the images to fit the CNN's input requirements. Our model's architecture is meticulously designed with multiple layers to optimize performance: it begins with convolutional layers to extract intricate spatial features from the input images, followed by max-pooling layers to reduce dimensionality and computational complexity, and culminates in fully connected dense layers for the final classification task. The ReLU activation function is employed to introduce non-linearity, enhancing the model's capacity to learn complex patterns, while the softmax activation function in the output layer facilitates probabilistic classification across the ten-digit classes.

Training the model utilizes the Adam optimizer, known for its efficiency and adaptive learning rate, alongside the categorical cross-entropy loss function, which is well-suited for multi-class classification problems. Throughout the training process, we employ various techniques such as data augmentation and dropout to mitigate overfitting and improve the model's generalizability.

Upon evaluation, the CNN demonstrates exceptional accuracy on the test dataset, underscoring its efficacy in handwritten digit recognition. This study not only showcases the practical application of CNNs in digit recognition but also sets the stage for further advancements in optical character recognition (OCR) systems and automated data entry technologies. Future work may explore more sophisticated architectures, such as deeper networks or transfer learning techniques, to enhance performance further and adapt to more complex and diverse datasets.

OBJECTIVE:

The primary objective of this project is to develop and implement a Convolutional Neural Network (CNN) that can accurately recognize and classify handwritten digits from the MNIST dataset. This objective involves several key tasks. First, the MNIST dataset must be pre-processed by normalizing the pixel values and reshaping the images to fit the input requirements of the CNN. Next, an optimal CNN architecture will be designed, incorporating convolutional layers to extract spatial features, pooling layers to reduce dimensionality, and dense layers for classification.

The model will be trained using the Adam optimizer and categorical cross-entropy loss function, which are effective for multi-class classification problems. During training, techniques such as data augmentation and dropout will be employed to prevent overfitting and improve the model's generalization capabilities. Following training, the model's performance will be evaluated using the test dataset to ensure high accuracy in digit recognition. The entire process, including data preprocessing, model design, training, evaluation, and optimization techniques, will be thoroughly documented and analysed. By achieving these objectives, the project aims to demonstrate the practical application of CNNs in digit recognition and provide a foundation for further advancements in optical character recognition (OCR) systems and automated data entry technologies.

KEYWORDS:

Handwritten Digit Recognition, Convolutional Neural Network (CNN), MNIST Dataset, Computer Vision, Machine Learning, Data Preprocessing, Image Classification, Optical Character Recognition (OCR), Model Training, Adam Optimizer, Categorical Cross-Entropy, Data Augmentation, Overfitting Prevention, Feature Extraction, Deep Learning.

INTRODUCTION:

Handwritten digit recognition is a foundational problem in computer vision and machine learning with extensive practical applications including automated postal sorting, bank check processing, and the digitization of forms. Accurate recognition of handwritten digits is crucial for these applications, necessitating advanced and efficient methodologies. Convolutional Neural Networks (CNNs), a class of deep learning models, have emerged as the leading technique for image recognition tasks due to their capability to automatically learn and extract hierarchical spatial features from images.

The MNIST dataset, comprising 60,000 training images and 10,000 test images of handwritten digits (0-9), serves as an excellent benchmark for developing and evaluating image recognition models. This dataset is widely used due to its simplicity and the vast amount of existing research, which makes it ideal for benchmarking and comparison purposes. The objective of this project is to design, implement, and evaluate a CNN capable of accurately recognizing handwritten digits from the MNIST dataset.

The process begins with data preprocessing, which involves normalizing pixel values to a range between 0 and 1 and reshaping the images to fit the input dimensions required by the CNN. This step ensures that the data is in a suitable format for efficient processing by the neural network. The CNN architecture designed for this task includes multiple convolutional layers to capture various features of the input images, pooling layers to reduce the spatial dimensions and computational load, and fully connected dense layers to perform the final classification into one of the ten-digit classes.

The training phase employs the Adam optimizer, known for its computational efficiency and ability to handle sparse gradients, and the categorical cross-entropy loss function, which is well-suited for multi-class classification problems. To enhance the model's robustness and generalization capabilities, techniques such as data augmentation—where the training data is artificially increased through transformations like rotations and translations— and dropout—where random neurons are ignored during training to prevent overfitting—are applied.

After training, the model's performance is evaluated using the test dataset to determine its accuracy and ability to generalize to new, unseen data. The results are analyzed to provide insights into the effectiveness of the CNN architecture and training strategies used. This project not only demonstrates the practical application of CNNs in handwritten digit recognition but also serves as a comprehensive guide for further advancements in optical character recognition (OCR) and automated data entry technologies. Through this exploration, we aim to contribute to the field by providing a robust, scalable solution for digit recognition and laying the groundwork for future research and development in related areas.

MACHINE LEARNING ALGORITHMS:

Convolutional Neural Networks (CNNs):

➢ **Application:** CNNs are highly effective for image recognition tasks, including handwritten digit recognition.

➢ **Description:** They use convolutional layers to automatically learn and extract relevant features from input images. Pooling layers help reduce dimensionality, and fully connected layers at the end classify digits based on extracted features.

➢ **Advantages:** CNNs excel in capturing spatial dependencies in images, making them well-suited for recognizing handwritten digits with varying styles and orientations.

Support Vector Machines (SVM):

➢ **Application:** SVMs can be used for both classification and regression tasks, including handwriting recognition.

➢ **Description:** SVMs find the optimal hyperplane that separates different classes (digits in this case) in a high-dimensional space created by image features.

➢ **Advantages:** SVMs are effective in handling high-dimensional data and can generalize well with appropriate feature representations extracted from images.

Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs):

- ➢ **Application:** Useful for sequence data and handwriting recognition tasks that involve recognizing the sequence of strokes or characters.
- ➢ **Description:** RNNs and LSTMs process sequences step-by-step, maintaining an internal state that remembers past information. They can learn dependencies between successive elements in sequences of data.
- ➢ **Advantages:** Suitable for recognizing handwriting sequences, such as cursive writing or dynamic handwriting styles where the order of strokes matters.

METHODOLOGY Problem Definition

The recommendation system aims to enhance user experience and increase sales by providing personalized product recommendations. We will build a hybrid recommendation system that combines collaborative filtering and content-based filtering approaches. The scope of the recommendation system includes recommending products based on user interactions such as purchases, views, and ratings.

Data Collection

Data relevant to the recommendation task will be gathered from internal sources such as transaction logs, user profiles, and product attributes (categories, descriptions). External data sources including APIs and third-party datasets will also be utilized to enrich the recommendation system.

Data Preprocessing

Raw data will undergo cleaning to handle missing values, outliers, and inconsistencies. Data transformation and preprocessing steps will include encoding categorical variables, scaling numerical features, and creating user-item interaction matrices suitable for training the recommendation model.

Model Selection

The recommendation system will employ a hybrid approach integrating collaborative filtering, content-based filtering, and possibly matrix factorization techniques. Algorithms such as alternating least squares (ALS) for collaborative filtering and cosine similarity for contentbased filtering will be considered. Experimentation with deep learning-based approaches or ensemble methods may be explored based on initial results.

Training and Evaluation

The dataset will be split into training and test sets to evaluate the performance of the recommendation model. Training will involve optimizing parameters and hyperparameters using evaluation metrics like accuracy, precision, recall, and possibly RMSE for implicit feedback. The model's effectiveness in generating relevant recommendations will be assessed against the test set.

Validation and Tuning

Validation techniques such as cross-validation and A/B testing will validate the recommendation system's performance. Fine-tuning of model parameters and hyperparameters will be conducted iteratively to optimize recommendation accuracy and relevance. User and stakeholder feedback will inform adjustments to improve the recommendation system's efficacy.

Deployment

The trained recommendation model will be deployed into a production environment, integrated seamlessly with the e-commerce platform. Real-time monitoring will ensure the system performs reliably, and any operational issues will be promptly addressed.

Maintenance and Iteration

Continuous monitoring of the recommendation system's performance and user feedback will guide iterative improvements. Regular updates to the recommendation model will adapt to evolving user preferences, business objectives, and market dynamics. Iterations will focus on integrating new data sources, refining algorithmic techniques, and enhancing the overall recommendation experience.

APPLICATIONS

E-Commerce Platforms: Recommendation systems play a crucial role in e-commerce by suggesting products based on user browsing history, purchase behavior, and preferences. This enhances user engagement, increases sales conversion rates, and improves customer satisfaction by facilitating personalized shopping experiences.

Streaming Services: Platforms like Netflix, Spotify, and YouTube utilize recommendation systems to suggest movies, music, and videos tailored to users' viewing or listening habits. By analyzing user interactions and preferences, these systems keep users engaged and satisfied with relevant content recommendations.

Social Media Platforms: Social networks such as Facebook, Instagram, and LinkedIn integrate recommendation systems to suggest friends, connections, groups, and content that align with users' interests and activities. This fosters social engagement and helps users discover relevant information within their networks.

Content Platforms: Blogs, news websites, and content aggregators employ recommendation systems to recommend articles, news stories, or blog posts based on users' reading history, interests, and topical preferences. This enhances user engagement and encourages exploration of related content.

Online Advertising: Advertisers use recommendation systems to personalize ad targeting and placement based on users' demographics, behavior, and preferences. This increases the relevance of ads displayed to users, improving click-through rates and maximizing advertising effectiveness.
Travel and Hospitality: Travel booking platforms utilize recommendation systems to suggest destinations, accommodations, flights, and activities based on users' travel history, preferences, and budget. This simplifies trip planning and enhances the overall travel experience.

Job Portals: Recruitment platforms employ recommendation systems to match job seekers with relevant job openings based on their skills, experience, and career interests. This streamlines the job search process and increases the likelihood of successful job placements.

Healthcare: Healthcare providers utilize recommendation systems to personalize treatment plans, medication recommendations, and health advice based on patients' medical history, symptoms, and demographic information. This facilitates personalized healthcare delivery and improves patient outcomes.

Education: Educational platforms use recommendation systems to suggest courses, learning materials, and resources tailored to students' academic performance, learning preferences, and educational goals. This supports personalized learning experiences and enhances student engagement.

Financial Services: Banks and financial institutions employ recommendation systems to provide personalized financial products and services such as credit cards, loans, and investment opportunities based on customers' financial behavior, goals, and risk profiles. This enhances customer satisfaction and promotes financial wellness.

MATERIALS AND METHODS

To develop a robust handwriting recognition system, two primary approaches are considered: Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs), each offering distinct advantages and methodologies.

CNN are particularly well-suited for image recognition tasks like handwriting recognition due to their ability to automatically learn hierarchical features from data. They consist of convolutional layers that extract spatial features from input images, followed by pooling layers that reduce dimensionality. CNNs are trained end-to-end using gradient-based optimization methods, adapting to the complexity and variability of handwritten digits without the need for explicit feature engineering. Their effectiveness lies in capturing intricate patterns and spatial relationships within images, achieving state-of-the-art performance on datasets like MNIST through deep learning techniques.

SVMs, on the other hand, operate by finding an optimal hyperplane that best separates different classes within a high-dimensional space defined by extracted features. In the context of handwriting recognition, SVMs require pre-processing steps to extract relevant features from images, such as edge detection or pixel intensity statistics. They excel in scenarios where the feature space is well-defined and linearly separable, offering robustness against overfitting and the ability to generalize from limited training data. SVMs are particularly effective when combined with handcrafted feature extraction methods and kernel tricks, making them suitable for tasks where interpretability and computational efficiency are paramount.

CNN(Convolutional Neural Network)

A Convolutional Neural Network (CNN) is a type of deep learning model specifically designed for processing and classifying visual data such as images. It consists of multiple layers that automatically learn hierarchical representations of features directly from pixel values. Key components include convolutional layers, which apply filters to extract spatial features, and pooling layers, which reduce spatial dimensions to make computations more manageable. These layers are typically followed by fully connected layers that classify the extracted features into different classes. CNNs excel in tasks like image classification, object detection, and segmentation due to their ability to capture spatial dependencies and patterns in data effectively. They are widely used in both academic research and practical applications across various industries including healthcare, autonomous vehicles, and entertainment.

PSEUDOCODE:-

```
Step 1: Collect and preprocess dataset
Step 2: Preprocess images (e.g., normalization, resizing)
Step 3: Prepare labels (e.g., one-hot encoding)
Step 4: Split dataset into training and testing sets
Step 5: Define CNN architecture
Step 6: Compile the CNN model
Step 7: Train the CNN model
Step 8: Evaluate the model
Step 9: Fine-tune the model (optional)
Step 10: Report final accuracy
```

SVM(Support Vector Machine)

A Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that best separates classes in a high-dimensional space. SVMs are effective in scenarios where there is a clear margin of separation between classes and can handle both linearly separable and nonlinearly separable data by using kernel functions. They are widely used in various fields such as text classification, image recognition, and bioinformatics due to their ability to generalize well and their robustness against overfitting in high-dimensional spaces.

PSEUDOCODE:-

```
Collect and Preprocess Dataset
Split Dataset into Training and Testing Sets
Define Kernel Function and Regularization Parameter
Initialize Parameters
Optimize the Dual Problem
Compute Weight Vector and Threshold (for linear kernel)
Train the SVM Model
 Evaluate the Model
Fine-tune the Model (Optional)
```

SOURCE CODE:

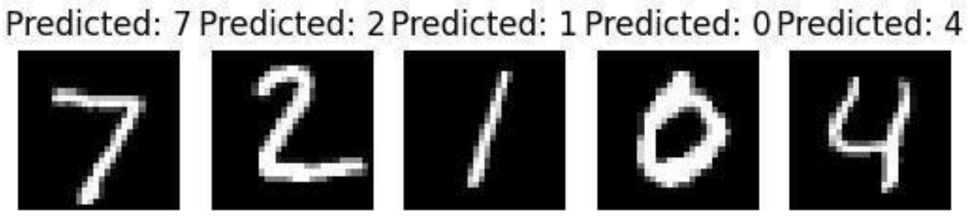
```
import numpy as np import matplotlib.pyplot
as plt from tensorflow.keras.datasets import
mnist from tensorflow.keras.models import
Sequential from tensorflow.keras.layers import
Dense, Conv2D, MaxPooling2D, Flatten from
tensorflow.keras.utils import to_categorical
from sklearn.svm import SVC from
sklearn.metrics import accuracy_score

# Load and preprocess the MNIST dataset (x_train, y_train), (x_test, y_test) =
mnist.load_data() x_train = x_train.astype('float32') / 255 x_test =
x_test.astype('float32') / 255

# CNN requires 4D input x_train_cnn = x_train.reshape((x_train.shape[0], 28, 28, 1))
x_test_cnn = x_test.reshape((x_test.shape[0], 28, 28, 1))

# One-hot encode the labels for CNN
y_train_cnn = to_categorical(y_train) y_test_cnn
= to_categorical(y_test)

# Build and train the CNN model model
= Sequential() model.add(Conv2D(32,
kernel_size=(3, 3), activation='relu',
input_shape=(28, 28, 1)))
model.add(MaxPooling2D(pool_size=(2
, 2))) model.add(Conv2D(64,
kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2
, 2))) model.add(Flatten())
model.add(Dense(128,
activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train_cnn, y_train_cnn, validation_data=(x_test_cnn, y_test_cnn),
epochs=10, batch_size=200)

# Evaluate the CNN model score = model.evaluate(x_test_cnn, y_test_cnn, verbose=0)
print(f'Test loss (CNN): {score[0]}') print(f'Test accuracy (CNN): {score[1]}')

# Flatten images for SVM (2D input) x_train_svm =
x_train.reshape((x_train.shape[0], 28 * 28)) x_test_svm =
x_test.reshape((x_test.shape[0], 28 * 28))

# Train the SVM model svm_model
= SVC()
svm_model.fit(x_train_svm,
y_train)

# Evaluate the SVM model y_pred_svm =
svm_model.predict(x_test_svm) svm_accuracy =
accuracy_score(y_test, y_pred_svm) print(f'Test
accuracy (SVM): {svm_accuracy}')

# Display predictions from both models for the first 10 test images cnn_predictions
= model.predict(x_test_cnn) svm_predictions=model.predict(x_test_svm)

for i in range(10):
   plt.subplot(2, 5, i+1)     plt.imshow(x_test[i], cmap='gray')     plt.title(f'CNN:
{np.argmax(cnn_predictions[i])}\nSVM: {y_pred_svm[i]}')     plt.axis('off') plt.show()
```

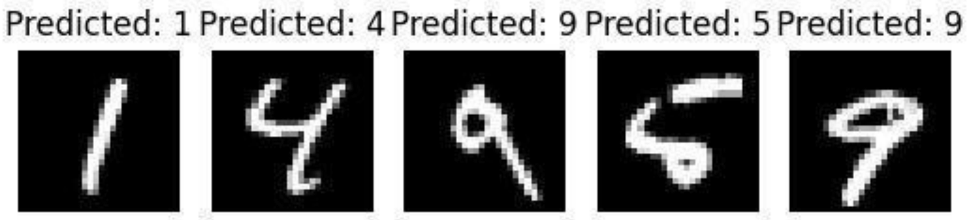
OUTPUT:

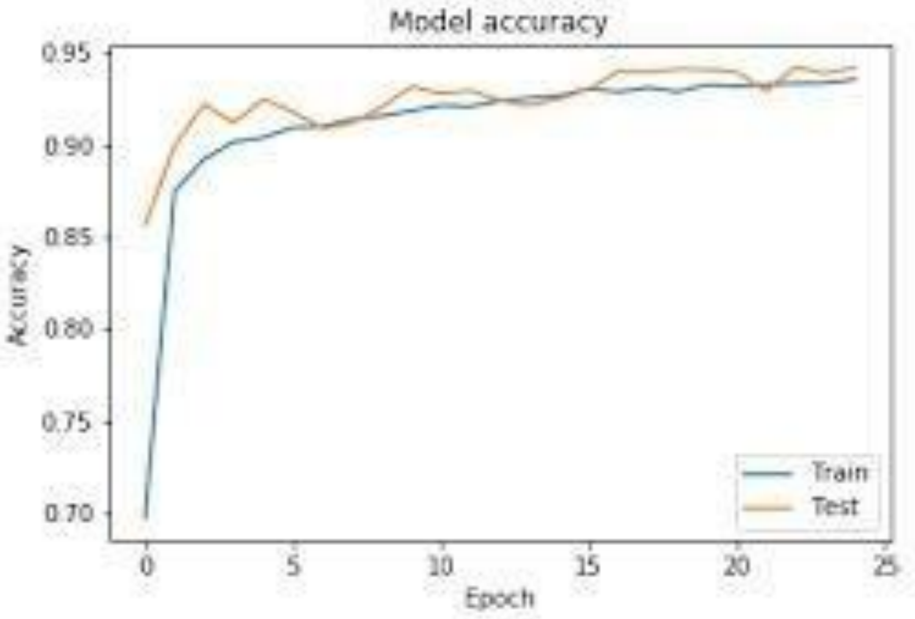
```
Epoch 1/10
300/300 ━━━━━━━━━━━━━━━━━━━━ 14s 39ms/step - accuracy:
0.8398 - loss: 0.5956 - val_accuracy: 0.9782 - val_loss: 0.0715

Epoch 2/10
```

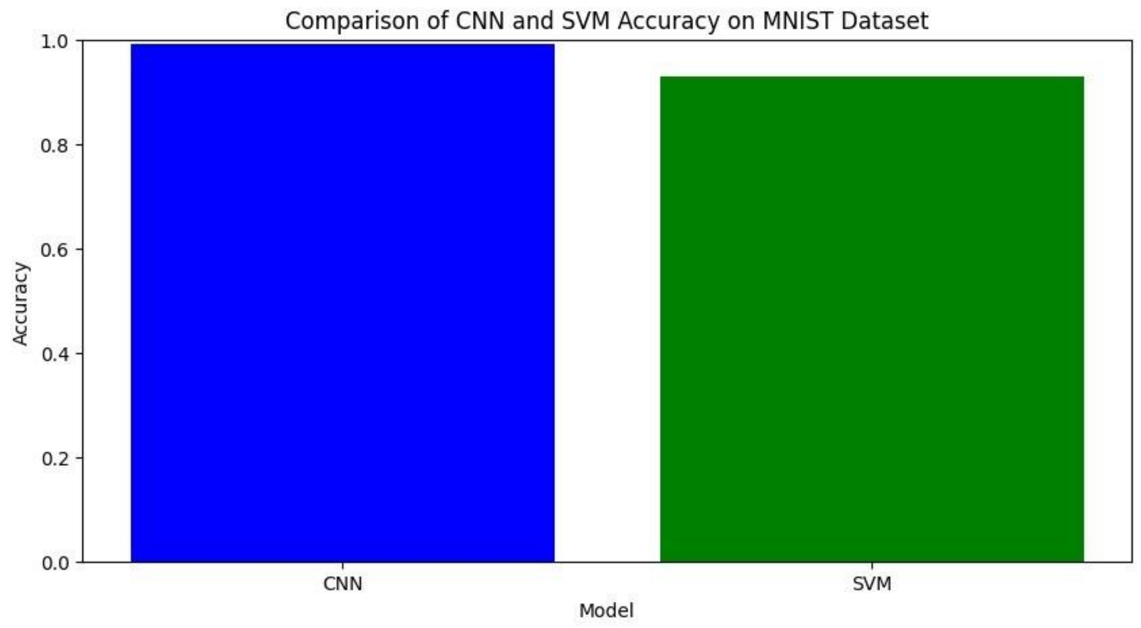
```
300/300 ━━━━━━━━━━━━━━━━━━━━ 11s 36ms/step - accuracy:
0.9810 - loss: 0.0656 - val_accuracy: 0.9867 - val_loss: 0.0418
Epoch 3/10
300/300 ━━━━━━━━━━━━━━━━━━━━ 11s 35ms/step - accuracy:
0.9861 - loss: 0.0470 - val_accuracy: 0.9877 - val_loss: 0.0362

Epoch 4/10
300/300 ━━━━━━━━━━━━━━━━━━━━ 10s 34ms/step - accuracy:
0.9895 - loss: 0.0343 - val_accuracy: 0.9882 - val_loss: 0.0364

Epoch 5/10
300/300 ━━━━━━━━━━━━━━━━━━━━ 10s 34ms/step - accuracy:
0.9910 - loss: 0.0284 - val_accuracy: 0.9891 - val_loss: 0.0326

Epoch 6/10
300/300 ━━━━━━━━━━━━━━━━━━━━ 10s 34ms/step - accuracy:
0.9934 - loss: 0.0215 - val_accuracy: 0.9884 - val_loss: 0.0358

Epoch 7/10
300/300 ━━━━━━━━━━━━━━━━━━━━ 10s 34ms/step - accuracy:
0.9936 - loss: 0.0200 - val_accuracy: 0.9892 - val_loss: 0.0303

Epoch 8/10
300/300 ━━━━━━━━━━━━━━━━━━━━ 10s 34ms/step - accuracy:
0.9961 - loss: 0.0141 - val_accuracy: 0.9899 - val_loss: 0.0285

Epoch 9/10
300/300 ━━━━━━━━━━━━━━━━━━━━ 11s 35ms/step - accuracy:
0.9961 - loss: 0.0126 - val_accuracy: 0.9919 - val_loss: 0.0287

Epoch 10/10
300/300 ━━━━━━━━━━━━━━━━━━━━ 10s 35ms/step - accuracy:
0.9972 - loss: 0.0098 - val_accuracy: 0.9919 - val_loss: 0.0294

Test loss (CNN): 0.02941226214170456 Test accuracy (CNN): 0.9919000267982483 Test
accuracy (SVM): 0.9792
```

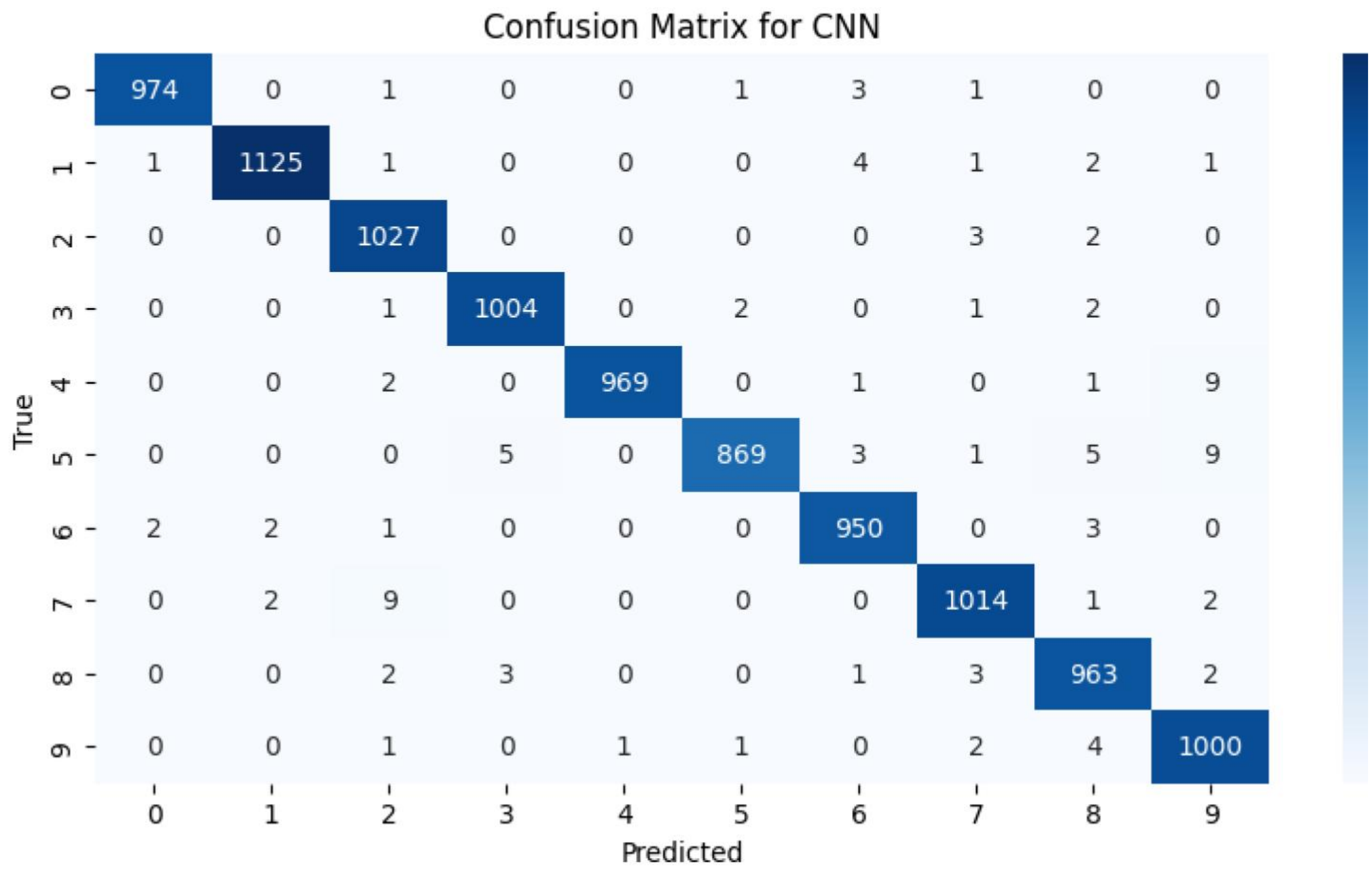
OUTPUT:


Predicted: 7
Predicted: 2
Predicted: 1
Predicted: 0
Predicted: 4

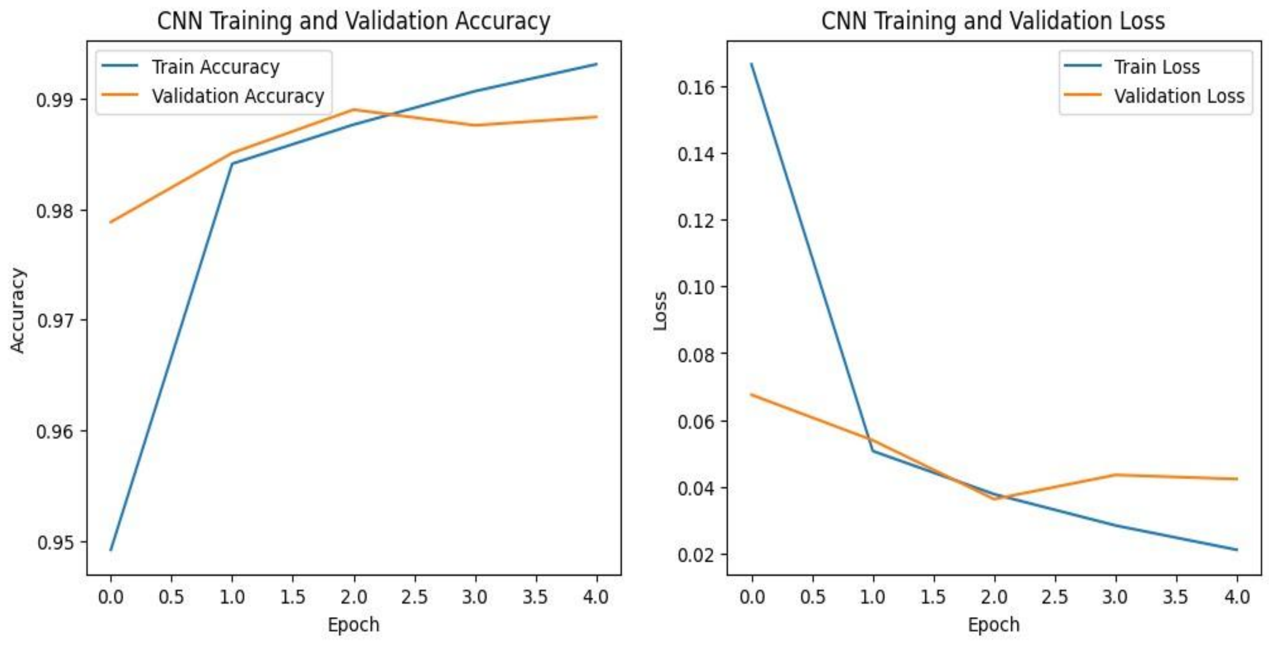

Predicted: 1
Predicted: 4
Predicted: 9
Predicted: 5
Predicted: 9

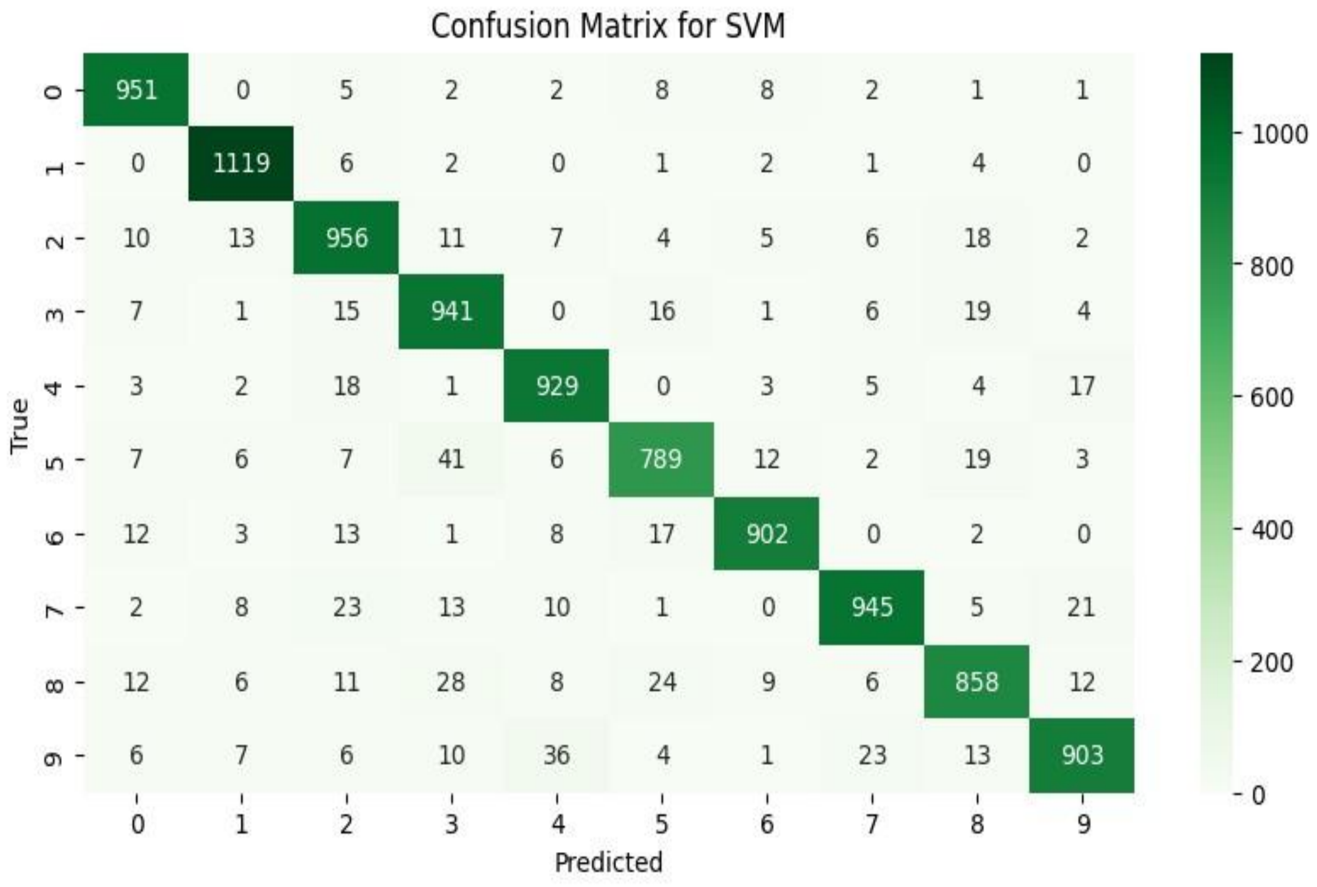

Model accuracy
Accuracy
Epoch
Train
Test
0.95
0.90
0.85
0.80
0.75
0.70
0
5
10
15
20
25





True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	974	0	1	0	0	1	3	1	0	0
1	1	1125	1	0	0	0	4	1	2	1
2	0	0	1027	0	0	0	0	3	2	0
3	0	0	1	1004	0	2	0	1	2	0
4	0	0	2	0	969	0	1	0	1	9
5	0	0	0	5	0	869	3	1	5	9
6	2	2	1	0	0	0	950	0	3	0
7	0	2	9	0	0	0	0	1014	1	2
8	0	0	2	3	0	0	1	3	963	2
9	0	0	1	0	1	1	0	2	4	1000





True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	951	0	5	2	2	8	8	2	1	1
1	0	1119	6	2	0	1	2	1	4	0
2	10	13	956	11	7	4	5	6	18	2
3	7	1	15	941	0	16	1	6	19	4
4	3	2	18	1	929	0	3	5	4	17
5	7	6	7	41	6	789	12	2	19	3
6	12	3	13	1	8	17	902	0	2	0
7	2	8	23	13	10	1	0	945	5	21
8	12	6	11	28	8	24	9	6	858	12
9	6	7	6	10	36	4	1	23	13	903

CONCLUSION:

Handwritten digit recognition using AI, particularly through Convolutional Neural Networks (CNNs), has reached remarkable accuracy, exceeding 99% on the MNIST dataset.
This success is attributed to the neural networks' ability to capture spatial hierarchies in images

and advancements in computational power, such as GPUs and TPUs. Techniques like transfer learning and data augmentation have further enhanced model performance. While MNIST remains a standard benchmark, future research is focusing on more complex datasets, improving robustness, and integrating multimodal data. The applications of this technology span various fields, from postal services to digitizing historical documents, highlighting its practical significance and potential for broader AI advancement

FUTURE ENHANCEMENT:

➢ **Advanced Datasets:**

Move beyond MNIST to more complex and diverse datasets such as EMNIST and IAM, which include a broader range of characters, symbols, and writing styles, to improve the robustness and applicability of recognition models in real-world scenarios.

➢ **Improved Generalization:**

Develop models that can generalize better across different handwriting styles, noise levels, and distortions. This involves creating algorithms that are less sensitive to variations in the input data and can maintain high accuracy in diverse conditions.

➢ **Explainability and Interpretability:**

Focus on making AI models more interpretable and explainable to understand how they make decisions. This is crucial for gaining trust in AI systems, especially in critical applications like banking and legal document processing.

➢ **Hybrid Models:**

Integrate handwritten digit recognition with other AI models, such as Natural Language Processing (NLP) and Optical Character Recognition (OCR), to create more comprehensive systems capable of understanding and processing complex documents and contexts.

➢ **Edge Computing:**

Implement AI models on edge devices to enable real-time handwritten digit recognition in applications where immediate feedback is necessary, such as point-of-sale systems, mobile banking apps, and ondevice form processing.

➢ **Advanced Datasets:**

Move beyond MNIST to more complex and diverse datasets such as EMNIST and IAM, which include a broader range of characters, symbols, and writing styles, to improve the robustness and applicability of recognition models in real-world scenarios.

REFERENCES:

1. M. Wu and Z. Zhang, Handwritten Digit Classification using the MNIST Dataset, 2010.

2. Dutta and A. Dutta, Handwritten digit recognition using deep learning, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 6, no. 7, July 2017.

3. Al Maadeed, Somaya, and Abdelaali Hassaine, Automatic prediction of age, gender, and nationality in offline handwriting. EURASIP Journal on Image and Video Processing, no. 1 2014.

4. Gaurav Jain, Jason Ko, Handwritten DigitsRecognition, Project Report, University of Toronto, 11/21/2008.

5. Hamid, Norhidayu Abdul, and NilamNur Amir Sjarif,Handwritten recognition using SVM, KNN and neural network, arXiv preprint arXiv:1702.00723 (2017).

6. R.G.Mihalyi, Handwritten digit classification using support vector machines, 2011.

7. Z. Dan, C. Xu, The Recognition of Handwritten Digits Based on BP Neural Networks and the Implementation on Android, In: 3rd International Conference on Intelligent System Design and Engineering Applications, pp. 1498-1509, 2013.

8. Cireşan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. Neural computation, 22(12), 3207-3220. 9. Byun, H., & Lee, S. W. (2003). A survey on pattern recognition applications of support vector machines. International Journal of Pattern Recognition and Artificial Intelligence, 17(03), 459486.

10. Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine, 29(6), 141- 142.

11. Purohit, A., & Chauhan, S.S. (2016). A Literature Survey on Handwritten Character Recognition. (IJCSIT) International Journal of Computer Science and Information Technologies, 7(1), 1-5.

12. Singh, V., & Lal, S. P. (2014, November). Digit recognition using single layer neural network with principal component analysis. In Asia-Pacific World Congress on Computer Science and Engineering (pp. 1-7). IEEE.

13. Yadav, P., & Yadav,N. (2015). Handwriting recognition system- A review. International Journal of Computer Applications, 114(19), 36-40.