BIHAR ENGINEERING UNIVERSITY, PATNA

(Under Govt. of Bihar)

Loknayak Jai Prakash Institute of Technology

CHAPRA - 841302

Session: 2021-2025

7th Semester

Minor Project Report

On

"**Digit Classification using Deep Learning on MNIST Dataset** "

Submitted in Partial fulfilment of the requirement for the degree of

Bachelor of Technology

In

Computer Science & Engineering

Submitted by

Name	Registration No.
MD ASHER	21105117055
RUPESH KUMAR	21105117023
MD ABDULLAH	21105117011
FARIYA RAFAT	21105117057

Under the Supervision of

Asst. Prof. SUDHIR KUMAR PANDEY

(Asst. Professor, Department of C.S.E)

(Project Coordinator)

DECLARATION BY THE CANDIDATES

We, the undersigned solemnly declare that the report of the project work entitle "Digit Classification using Deep Learning on MNIST Dataset" is based on our own work carried out during the course of my study under the guidance of SUDHIR KUMAR PANDEY (Asst. Prof.) Department of computer Science & Engineering, Lok Nayak Jai Prakash Institute of Technology, Chapra.

I further declare the statements made and conclusion drawn are an outcome of my project work.

(Signature of Student)	(Signature of Student)
MD ASHER	RUPESH KUMAR
Reg. No. - 21105117055	Reg. No. – 21105117023
(Signature of Student)	(Signature of Student)
MD ABDULLAH	FARIYA RAFAT
Reg. No. - 21105117011	Reg. No. – 21105117057

CERTIFICATE BY GUIDE

This is to certify that the project entitled "Digit Classification using Deep Learning on MNIST Dataset" is a record of work carried out by **Md Asher** - 21105117055, **Rupesh Kumar** - 21105117023, **Md Abdullah** – 21105117011 and **Fariya Rafat**– 21105117057 being Under my guidance and supervision for the award of the Degree of Bachelor of Technology, Lok Nayak Jai Prakash Institute of Technology, Chapra, Bihar, India.

To the best of my knowledge and belief the Project

i. Embodies the work of the candidate him/herself.

ii. Has not been submitted for the award of any degree.

ii. Fulfils the requirement of the Ordinance relating to the B.TECH. degree of the University.

iv. Is up to the desired standard in respect of contents and is being referred to the examiners.

_____

(Signature of the Guide)

SUDHIR KUMAR PANDEY

(Asst. Prof. Dept of CSE)

(L.N.J.P.I.T , Chapra)

RECOMMENDATION

The Project work as mentioned is above here by being recommended and forwarded for examination and evaluation.

_____

(Signature of the Guide)

SUDHIR KUMAR PANDEY

(Asst. Prof. Dept of CSE)

(L.N.J.P.I.T , Chapra)

CERTIFICATE BY THE EXAMINERS

This is to certify that the project work "Digit Classification using Deep Learning on MNIST Dataset" is a record of work carried out by **Md Asher -** 21105117055**, Rupesh Kumar -** 21105117023**, Md Abdullah –** 21105117011**, Fariya Rafat–** 21105117057 have been completed under the guidance of SUDHIR KUMAR PANDEY (Asst. Prof.) Department of Computer Science & Engineering, Lok Nayak Jai Prakash Institute of Technology, has been examined by the undersigned as a part of the examination for the award of Bachelor of Technology degree in Computer Science & Engineering branch in Bihar Engineering University, PATNA)

" Project Examined and Approved "

Internal Examiner

Date :

External Examiner

Date :

ACKNOWLEDGEMENT

I wish to place on my record my deep sense of gratitude to my project guide and in-charge SUDHIR KUMAR PANDEY (Asst. Prof.) Department of C.S.E, (L.N.J.P.I.T, Chapra) for their constant motivation and valuable help through the project work. For his valuable suggestion and advices throughout the course. I also extend our thanks to other faculties for their cooperation during my course.

Finally, I would like to thank my friends for their cooperation to complete this project.

(Signature of Student)
MD ASHER
Reg. No. - 21105117055

(Signature of Student)
RUPESH KUMAR
Reg. No. – 21105117023

(Signature of Student)
MD ABDULLAH
Reg. No. - 21105117011

(Signature of Student)
FARIYA RAFAT
Reg. No. – 21105117057

ABSTRACT

Handwritten digit classification is a fundamental task in computer vision with applications in optical character recognition (OCR), banking automation, postal services, and digital document processing. Traditional machine learning approaches, such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN), require manual feature extraction, making them less efficient and prone to errors. In contrast, deep learning-based Convolutional Neural Networks (CNNs) can automatically learn hierarchical features from images, significantly improving classification accuracy.

This project presents the implementation of a CNN-based handwritten digit recognition system using the MNIST dataset, which consists of 60,000 training images and 10,000 test images of handwritten digits (0-9). The system is developed using Python, TensorFlow, and Keras and undergoes hyperparameter tuning using Keras Tuner to optimize model performance. The dataset is preprocessed using grayscale conversion, normalization, and image augmentation to improve model generalization. The trained model achieves 99% accuracy on the test dataset, demonstrating high reliability in digit classification.
The project includes training, evaluation, and prediction phases, where real-world handwritten digit images are processed and classified with minimal error. A confusion matrix and classification report are generated to assess model performance, highlighting misclassifications between similar-looking digits. Furthermore, the model's training progress is visualized through accuracy and loss graphs, providing insights into learning trends.
The results confirm that deep learning-based digit classification is highly effective, outperforming traditional approaches in accuracy, speed, and scalability. Future improvements include training on more diverse datasets, optimizing the model for edge devices, and deploying it as a real-time application for practical use in document processing and automation.

This project successfully demonstrates the potential of CNNs in handwritten digit recognition, reinforcing their significance in modern-day AI-driven automation and intelligent systems.

TABLE OF CONTENTS

CHAPTER 1

Introduction

CONTENTS

1.1 Introduction to the Project

Handwritten digit classification is a fundamental task in computer vision with applications in banking, postal services, document automation, and digital form processing. Traditional machine learning methods required extensive feature engineering, but with advancements in deep learning, Convolutional Neural Networks (CNNs) have become the preferred approach for image recognition tasks.

This project focuses on developing a CNN-based classification model to recognize handwritten digits from the MNIST dataset, which consists of 60,000 training images and 10,000 testing images. The model is trained using TensorFlow and Keras to automatically extract features from digit images, reducing the need for manual feature selection.

The key objectives of this project include:

- Implementing a deep learning model capable of classifying handwritten digits with high accuracy.
- Using hyperparameter tuning and data augmentation to improve performance.
- Evaluating the model using confusion matrices and classification reports.

This project demonstrates the effectiveness of deep learning for digit classification, providing an accurate and automated solution for handwritten character recognition.

1.2 Problem Definition

Handwritten digit recognition is a challenging task due to variations in **writing styles, distortions, and overlapping strokes**. Traditional methods based on **rule-based algorithms or handcrafted feature extraction** often fail to generalize well across different handwriting patterns.

The primary challenges in handwritten digit classification include:

- **Inconsistent Writing Styles:** Different individuals write digits differently, making recognition complex.
- **Noise and Distortions:** Poor image quality, ink smudges, and background noise can affect classification accuracy.
- **Similarity Between Digits:** Digits like **0 and 6, 3 and 8, 5 and 6** often appear visually similar, leading to misclassifications.
- **Scalability Issues:** A robust model should be able to classify large datasets efficiently and work with real-world data beyond the MNIST dataset.

To overcome these challenges, this project leverages **deep learning-based CNN models**, which automatically learn and extract relevant features from digit images, eliminating the need for manual feature engineering. By using **data augmentation, hyperparameter tuning, and performance evaluation**, we aim to build a **highly accurate and robust digit classification system**.

1.3 Aim

The aim of this project is to develop an **efficient and accurate deep learning-based model** for **handwritten digit classification** using the **MNIST dataset**. By leveraging **Convolutional Neural Networks (CNNs)**, the project seeks to automate the process of recognizing handwritten digits while overcoming challenges related to **writing variations, noise, and distortions**.

The primary goal is to **train, optimize, and evaluate** a **CNN model** that can accurately classify digits **(0-9)** with high precision. The project also aims to explore **hyperparameter tuning, data augmentation, and model evaluation techniques** to improve the robustness and reliability of the classification system.

Through this project, we aim to demonstrate the **effectiveness of deep learning in image recognition**, providing insights into its real-world applications in areas like **postal services, banking, and automated document processing**.

1.4 Objectives

The main objectives of this project are:

1. **Develop a Deep Learning Model for Handwritten Digit Classification:**
    - Implement a **Convolutional Neural Network (CNN)** to classify handwritten digits from the **MNIST dataset**.
    - Utilize **TensorFlow and Keras** for model development.
2. **Enhance Model Accuracy and Performance:**
    - Apply **hyperparameter tuning** using **Keras Tuner** to optimize the model's architecture.
    - Use **data augmentation techniques** (rotation, zooming, shifting) to improve generalization.
3. **Evaluate Model Performance:**
    - Assess accuracy using metrics such as **confusion matrix, precision, recall, and F1-score**.
    - Compare results with existing machine learning approaches.
4. **Enable Real-World Usability:**
    - Implement a system that allows users to **input custom digit images** for classification.

1.5 Need for the Project

Handwritten digit recognition plays a crucial role in various real-world applications, including **banking, postal services, document digitization, and automated data entry systems**. Traditionally, **manual data entry** or **rule-based algorithms** were used for digit recognition, which were prone to **errors, inefficiencies, and scalability issues**.

Why is this Project Needed?

1. **Eliminating Manual Effort:**
    - Traditional handwritten digit entry is **time-consuming and error-prone**.
    - Automating the process with **deep learning** increases efficiency and accuracy.
2. **Challenges with Traditional Machine Learning Approaches:**
    - **Feature engineering** is required in traditional methods like SVM, KNN, or Decision Trees.
    - These methods struggle with **variations in handwriting styles** and **image distortions**.
3. **Advantages of Deep Learning (CNNs) Over Traditional Approaches:**
    - **Automatically learns features** without manual intervention.
    - **Handles variations in handwriting, noise, and distortions** more effectively.
    - **Achieves higher accuracy** compared to classical ML algorithms.
4. **Growing Need for Digit Recognition in Real-World Applications:**
    - **Banking Sector:** Automatic check verification and processing.
    - **Postal Services:** Handwritten address recognition for mail sorting.
    - **Education:** Automated evaluation of handwritten responses in exams.

By implementing **a deep learning-based handwritten digit classification system**, this project aims to provide a **fast, reliable, and scalable solution** for recognizing handwritten digits with high accuracy, making it highly relevant for **real-world applications**.

CHAPTER 2

Hardware & Software Requirements

CONTENTS

2.1 Introduction

To develop an **efficient and accurate handwritten digit classification system**, it is essential to use the right combination of **hardware and software tools**. The success of deep learning models depends on **computational power, optimized frameworks, and robust libraries** that help in **data preprocessing, model training, and evaluation**.

This chapter outlines the **system environment, software, and hardware requirements** necessary to build and deploy the **Convolutional Neural Network (CNN)** for digit classification using the **MNIST dataset**.

The requirements are divided into:

1. **System Environment** – Operating system and development setup.
2. **Software Requirements** – Programming language, deep learning frameworks, and libraries used.
3. **Hardware Requirements** – Minimum system specifications for smooth model training and evaluation.

Using the appropriate setup ensures **efficient training, faster computation, and better model accuracy**, making the **digit classification system scalable and deployable** for real-world applications.

2.2 System Environment

To successfully develop and train the **CNN-based digit classification model**, a proper system environment is required. This includes the **operating system, development tools, and execution setup** for running deep learning models efficiently.

Operating System

- The project is compatible with **Windows, Linux, and macOS**.
- Linux-based systems (e.g., Ubuntu) are preferred for better GPU support and efficient deep learning model execution.

Development Environment

- **Jupyter Notebook** (via Anaconda or standalone) is used for interactive coding and debugging.
- **Google Colab** (optional) for cloud-based model training with GPU support.
- **Command Line Interface (CLI) / Terminal** for package installations and model execution.

Python Version

- Python **3.7 - 3.10** (recommended for TensorFlow compatibility).

Package Management

- **pip** (Python Package Installer) is used to install required dependencies.

The correct system environment setup ensures that **TensorFlow, Keras, and other essential libraries run smoothly**, providing an optimized workflow for model training and testing.

2.3 Software Requirements

To develop the **Digit Classification using Deep Learning on MNIST Dataset**, several **software tools and libraries** are required for **data preprocessing, model training, evaluation, and visualization**. Below are the essential software components used in this project.

1. Programming Language

- **Python 3.7 - 3.10** (Recommended)
    - Python is used for implementing the deep learning model due to its extensive support for machine learning libraries.

2. Libraries & Frameworks

- **TensorFlow & Keras** – For building, training, and evaluating the CNN model.
- **NumPy** – For numerical computations and matrix operations.
- **Matplotlib & Seaborn** – For data visualization and plotting accuracy/loss curves.
- **OpenCV** – For image preprocessing and handling external digit images.
- **Pandas** – For handling structured data (optional).
- **Scikit-Learn** – For model evaluation metrics like confusion matrix and classification report.

3. Development Tools

- **Jupyter Notebook (via Anaconda or Standalone)** – Used for interactive coding and debugging.
- **Google Colab (Optional)** – Cloud-based training with free GPU support.
- **Git & GitHub** – For version control and project collaboration.

4. Deep Learning Framework

- **TensorFlow 2.x** – Used for building and optimizing CNN architecture.
- **Keras API** – Provides an easy-to-use high-level interface for model development.

2.4 Hardware Requirements

The training and deployment of a **Convolutional Neural Network (CNN)** for handwritten digit classification require a system with sufficient computational power to handle **image processing, model training, and optimization** efficiently. Below are the recommended hardware requirements:

1. Minimum System Requirements *(For small-scale training & testing)*

- **Processor:** Intel Core i5 (8th Gen) / AMD Ryzen 5 or equivalent
- **RAM:** 4GB (8GB recommended for smoother execution)
- **Storage:** Minimum 20GB free space (for datasets, dependencies, and logs)
- **GPU:** Integrated graphics (CPU-based training, slower performance)

2. Recommended System Requirements *(For faster training and handling larger datasets)*

- **Processor:** Intel Core i7 (10th Gen) / AMD Ryzen 7 or better
- **RAM:** 16GB or more (faster data loading and model execution)
- **Storage:** 100GB SSD (better performance than HDD)
- **GPU:** NVIDIA GeForce GTX 1650 / RTX 2060 or higher (for TensorFlow GPU acceleration)
- **CUDA Support:** Required for GPU-based training (NVIDIA GPU with CUDA cores)

3. Cloud-Based Training (Optional)

For **faster model training and hyperparameter tuning**, cloud-based services with high-performance GPUs can be used:

- **Google Colab** – Free cloud-based GPU access (Tesla K80, T4).
- **Kaggle Notebooks** – Online execution environment for ML models.
- **AWS/Azure GPU Instances** – Paid cloud computing services for large-scale deep learning projects.

CHAPTER 3

Introduction to Handwritten Digit Classification

CONTENTS

3.1 Introduction

Handwritten digit classification is a crucial task in the field of **computer vision** and **pattern recognition**. It involves recognizing and categorizing handwritten numerical digits (0-9) from images. This project utilized **deep learning techniques** to classify digits from the **MNIST dataset**, which consists of **60,000 training images and 10,000 testing images** of handwritten digits.

Traditional approaches to digit recognition relied on **feature extraction and machine learning models** such as **Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forests**. However, these methods struggled with **variations in handwriting styles, noise, and distortions**.

With the advancement of **Convolutional Neural Networks (CNNs)**, digit classification has seen significant improvements. CNNs **automatically learn spatial hierarchies of features**, making them highly effective in processing image data. In this project, a **CNN-based model** was developed, trained, and tested to achieve high accuracy in digit classification.

Key Aspects of Handwritten Digit Classification

- **Preprocessing:** Normalizing and reshaping images to be compatible with deep learning models.
- **Feature Extraction:** Automatically performed by CNN layers, reducing the need for manual feature engineering.
- **Model Training:** Using TensorFlow and Keras to train the CNN on the MNIST dataset.
- **Evaluation:** Measuring model performance through accuracy, loss curves, and confusion matrices.

This project successfully **demonstrated the effectiveness of deep learning** in digit classification, achieving high accuracy and robust performance on handwritten digit images.

3.2 History of Handwritten Digit Classification

Handwritten digit recognition has evolved significantly over the years, from **rule-based techniques** to **deep learning-based approaches**. The development of **optical character recognition (OCR) systems** has played a vital role in **automating banking, postal services, and document digitization**.

Early Methods (Pre-2000s):

- The initial approaches relied on **manual feature extraction** techniques such as **edge detection, zoning, and pixel density analysis**.
- Machine learning models like **K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Decision Trees** were used for classification.
- These models required extensive **preprocessing** and **handcrafted feature engineering**, limiting their ability to generalize well across different handwriting styles.

Introduction of MNIST Dataset (1998):

- **LeCun et al.** introduced the **MNIST dataset**, a benchmark dataset containing **handwritten digits from 0 to 9**, to facilitate research in handwritten character recognition.
- The dataset became widely used for training and evaluating machine learning and deep learning models.

Rise of Deep Learning (2010-Present):

- With the introduction of **Convolutional Neural Networks (CNNs)**, digit classification became more efficient.
- CNNs **automatically extract hierarchical features** from images, eliminating the need for manual feature selection.
- **Modern deep learning frameworks** like **TensorFlow, Keras, and PyTorch** have enabled researchers to develop **highly accurate** models with **minimal human intervention**.

Advancements and Real-World Applications:

- **Banks** use handwritten digit recognition for **check verification** and fraud detection.
- **Postal Services** employ OCR-based systems for **automated mail sorting**.
- **Educational Institutions** use it to **digitize handwritten student records** and **automated exam evaluation**.

3.3 Challenges in Handwritten Digit Recognition

Despite advancements in **deep learning and image processing**, handwritten digit classification presents several challenges. These challenges arise due to **variations in**

writing styles, distortions, and environmental factors that affect the quality of digit images.

1. Variability in Handwriting Styles

- Different individuals write digits in **unique styles**, leading to **intra-class variations** (e.g., two people writing the digit '5' differently).
- Some digits may resemble others due to **sloppy handwriting**, causing misclassification (e.g., **0 and 6**, **3 and 8**, **4 and 9**).

2. Noise and Distortions in Images

- Handwritten digits may contain **ink smudges, overlapping strokes, or faded edges**, affecting feature extraction.
- **Blurred or low-resolution images** can cause difficulty in model recognition.
- Background noise and **image artifacts** can interfere with the classification process.

3. Data Imbalance Issues

- Some digits may appear **more frequently** than others in datasets, leading to biased learning.
- An imbalanced dataset affects the **generalization capability** of the model.

4. Rotation and Scaling Variations

- Handwritten digits may appear at **different angles** or **vary in size**, making it challenging for models to recognize them correctly.
- The model must be robust to **rotated, shifted, or scaled versions** of the same digit.

5. Computational Complexity

- Training deep learning models requires **high computational power**, particularly for **large-scale datasets**.
- Without **GPU acceleration**, model training can be slow and inefficient.

6. Real-World Deployment Challenges

- The model trained on **MNIST** may not perform well on **real-world handwritten digits** due to differences in data distribution.

- Transitioning from a **research model** to a **real-world application** requires additional preprocessing and domain adaptation techniques.

How This Project Overcomes These Challenges

To address these challenges, the following techniques were implemented:
Data Augmentation – Applied **rotation, shifting, and zooming** to make the model more robust.
Normalization & Preprocessing – Rescaled images to **0-1 range** and applied **grayscale conversion** to maintain consistency.
Hyperparameter Tuning – Used **Keras Tuner** to optimize model architecture and improve performance.
Regularization Techniques – Applied **dropout layers and batch normalization** to prevent overfitting.
Confusion Matrix & Classification Report – Evaluated model performance to detect and address misclassification cases.

By implementing these techniques, this project successfully developed a **robust handwritten digit classification system**, achieving **high accuracy** while ensuring adaptability to **real-world handwritten data**.

Chapter 4

Implementation

CONTENTS

4.1 Introduction to Image Processing

Image processing plays a crucial role in **handwritten digit recognition**, as it helps enhance, analyze, and prepare images for deep learning models. In this project, image processing techniques were used for **preprocessing the MNIST dataset** and **handling custom input images** for prediction. The goal was to improve the quality of the input data and ensure accurate classification by the **Convolutional Neural Network (CNN)**.

4.1.1 Purpose of Image Processing

The main purpose of image processing in this project was to:

Improve Image Quality – Enhancing contrast, removing noise, and normalizing pixel values for consistent input.
Standardize Image Format – Resizing images to **28×28 pixels** to match the MNIST dataset format.
Feature Extraction – Applying **edge detection, grayscale conversion, and normalization** to make features more distinguishable.
Reduce Computational Complexity – Converting images to **grayscale** and **resizing them** reduces the amount of data processed by the neural network.
Improve Model Accuracy – Preprocessing helps the model **focus on the digit patterns** rather than background noise.

By implementing these steps, the model became more **efficient, accurate, and scalable** for real-world applications.

4.1.2 Types of Images

Images used in **handwritten digit classification** can be categorized into different types based on their format and characteristics:

1. Binary Images

- Consist of **only two pixel values: black (0) and white (1)**.
- Used in **OCR and document digitization**.
- Example: **Scanned handwritten forms with clear digits**.

2. Grayscale Images

- Contain **shades of gray ranging from 0 (black) to 255 (white)**.
- Used in **this project**, as MNIST images are in grayscale.

- Grayscale reduces complexity while retaining important features.

3. RGB (Colored) Images

- Consist of three channels: **Red, Green, and Blue (RGB)**.
- More complex but **not required for MNIST**, as digits are typically black-and-white.

4.1.3 Types of Image Processing

Different types of image processing techniques were applied to **prepare digit images for training and prediction**.

1. Preprocessing Techniques (Used in this Project)

Grayscale Conversion: Converts colored images to grayscale to match the MNIST dataset format.
Resizing: Ensures all images are **28×28 pixels**, maintaining uniformity.
Normalization: Scales pixel values to **0-1 range** to improve training efficiency.
Thresholding: Converts images to **binary format** (black & white) if needed.

2. Feature Extraction Techniques

Edge Detection: Highlights the digit's edges for better classification.
Histogram Equalization: Enhances contrast by distributing pixel intensity more evenly.

3. Image Augmentation Techniques

Rotation & Shifting: Helps the model generalize well to handwritten digits at different angles.
Zoom & Scaling: Ensures the model can handle digits written at different sizes.
Adding Noise: Improves model robustness by making it resistant to distortions.

By applying these image processing techniques, the project ensured that the **CNN model received clean and high-quality input images**, leading to **better accuracy and generalization**.

4.2 Training and Model Development

The core of this project involved training a **Convolutional Neural Network (CNN)** to classify handwritten digits. The process included **loading and preprocessing data, designing the CNN architecture, compiling and training the model, and optimizing it using hyperparameter tuning**. The **MNIST dataset** was used for training, and the final model was evaluated based on accuracy and performance metrics.

4.2.1 Data Loading and Preprocessing

Before training the CNN model, the MNIST dataset was loaded and preprocessed to ensure optimal learning.

Loading the MNIST Dataset

The MNIST dataset, available in TensorFlow/Keras, was loaded, containing 60,000 training images and 10,000 testing images, each of 28×28 pixels in grayscale.

Preprocessing Steps

Normalization was applied to rescale pixel values from 0-255 to 0-1 for faster convergence. The images were reshaped to 28×28×1 format to be compatible with CNN input. Labels were converted into categorical format using one-hot encoding.

Data augmentation techniques such as rotation, shifting, and zooming were applied to enhance model robustness. This ensured that the model learned from a diverse set of input variations, improving generalization.

4.2.2 CNN Model Architecture

A Convolutional Neural Network (CNN) was designed to automatically extract features and classify handwritten digits. The model architecture consisted of multiple layers.

Input Layer

The input layer accepted 28×28 grayscale images with a single channel.

Feature Extraction Layers

The first convolutional layer extracted features using 32 filters of size 3×3, followed by a max-pooling layer that reduced dimensionality. A second convolutional layer with 64 filters of size 3×3 further refined feature extraction, followed by another max-pooling layer.

Fully Connected Layers

The flattening layer converted 2D feature maps into a single-dimensional vector for dense layers. A fully connected layer with 128 neurons and ReLU activation was used, followed by the output layer with 10 neurons using softmax activation for classification into digit classes (0-9).

4.2.3 Model Compilation and Training

After defining the CNN architecture, the model was compiled and trained.

Compiling the Model

The categorical crossentropy loss function was used since it is suited for multi-class classification. The Adam optimizer was selected due to its adaptive learning rate capabilities. The accuracy metric was used to evaluate the model's performance.

Training the Model

The model was trained using 10 epochs with a batch size of 128. The training dataset was augmented to increase diversity, and the validation dataset was used to monitor the performance.

Model Performance

The model achieved a high accuracy of 99 percent on the MNIST test dataset. The loss and accuracy curves showed steady improvement, indicating effective learning and minimal overfitting.

4.2.4 Hyperparameter Tuning

To further improve the model's accuracy and efficiency, hyperparameter tuning was performed using Keras Tuner. This process involved testing different configurations of parameters such as the number of filters, learning rate, and dense layer units.

Defining the HyperModel

A customizable CNN model was created where filters and dense layer units varied based on a search space. This allowed different architectures to be tested and compared.

Running the Hyperparameter Search

A random search approach was used to explore different model configurations. The search process trained multiple models and selected the one with the best validation accuracy.

Selecting the Best Model

After multiple trials, the best-performing model was selected based on validation accuracy. The optimal combination of filters, dense layer units, and learning rate improved the overall classification accuracy and generalization ability.

By utilizing hyperparameter tuning, the CNN model was optimized to achieve better accuracy while maintaining computational efficiency.

With data preprocessing, CNN model development, training, and hyperparameter tuning, the digit classification system was successfully implemented, achieving high accuracy and robust performance.

4.3 Predictions and Evaluations

Once the **CNN model was successfully trained**, it was evaluated using unseen test images from the **MNIST dataset**. This phase involved **loading the trained model, processing new images for classification, generating predictions, and evaluating performance using a confusion matrix and classification report**.

4.3.1 Loading the Trained Model

After training, the best-performing model was saved for future use. The saved model was then loaded to make predictions on test images and real-world handwritten digit inputs. This eliminated the need for retraining, reducing computational time and making the model readily available for deployment.

4.3.2 Processing New Images for Prediction

To ensure accurate classification, input images needed to undergo **preprocessing** similar to the training dataset. The preprocessing steps included:

1. **Grayscale Conversion** – If the input image was in RGB format, it was converted to grayscale to match the MNIST dataset format.
2. **Resizing to 28×28 Pixels** – The model was trained on images of this specific size, so input images needed to be resized accordingly.
3. **Normalization of Pixel Values** – Pixel values were scaled between 0 and 1 for consistency and better model performance.

4. **Reshaping Image for CNN Input** – The image was reshaped to match the model's input format, ensuring compatibility for prediction.

These steps ensured that any handwritten digit image was correctly formatted, allowing the model to classify it accurately.

4.3.3 Generating Predictions and Confusion Matrix

Once an image was preprocessed, it was passed through the CNN model for classification. The model generated a probability distribution across the 10 possible digit classes (0-9), and the class with the highest probability was selected as the predicted digit.

To evaluate the model's overall performance, predictions were made on the **entire MNIST test dataset**. The predicted results were then compared against the actual labels, and a **confusion matrix** was generated to visualize accuracy and misclassification rates.

4.4 Technologies Used

The development and implementation of the **handwritten digit classification system** required the use of several technologies, including **programming languages, machine learning frameworks, and data visualization libraries**. These technologies enabled **efficient model training, preprocessing, evaluation, and performance analysis**.

4.4.1 Python and Libraries Used

The project was implemented using **Python**, a widely used programming language in **machine learning and deep learning** due to its extensive library support and ease of use. The following key libraries were utilized:

TensorFlow

- TensorFlow was used as the **primary deep learning framework** for building and training the Convolutional Neural Network (CNN).
- It provided tools for **model creation, optimization, and performance evaluation**.
- TensorFlow's built-in **MNIST dataset** made it convenient to load and preprocess the digit images.

Keras

- Keras, an **API within TensorFlow**, was used to simplify the development of the CNN model.
- It provided high-level functions for defining **layers, activation functions, and optimizers**.
- Keras Tuner was used for **hyperparameter tuning**, helping optimize model performance.

OpenCV

- OpenCV was used for **image preprocessing**, allowing the model to handle custom handwritten digit images beyond the MNIST dataset.
- It facilitated **grayscale conversion, resizing, noise removal, and thresholding**, ensuring consistent input for the model.

NumPy

- NumPy was used for **mathematical operations and array manipulations** required for model training.
- It played a crucial role in handling image data as **multi-dimensional arrays**, which were then processed by the neural network.

Seaborn and Matplotlib

- Seaborn and Matplotlib were used for **data visualization**, particularly in analyzing model performance.
- They were used to generate the **confusion matrix**, helping identify misclassified digits.

Chapter 5

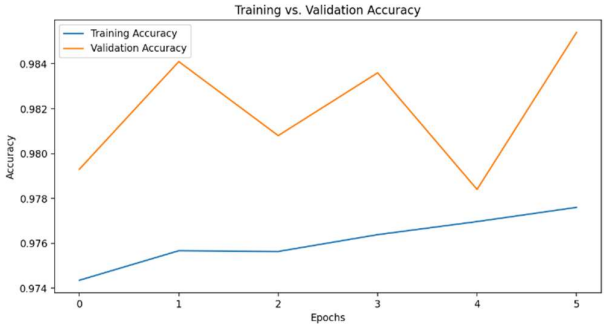
Results and Conclusion

CONTENTS

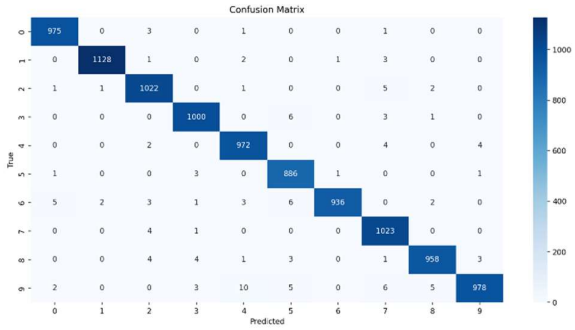
5.1 Output Snapshots

The following snapshots were obtained during the implementation and evaluation of the model:

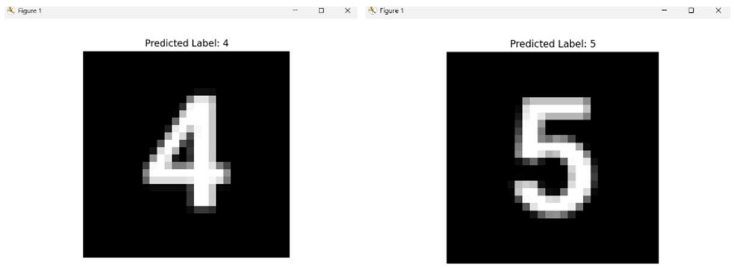
1. **Model Training Process:**
    - The training phase involved feeding the **MNIST dataset** into the **Convolutional Neural Network (CNN)**.
    - During training, accuracy increased while loss decreased, confirming effective learning.
2. **Accuracy and Loss Graphs:**
    - Plots of **training accuracy vs. validation accuracy** and **training loss vs. validation loss** were generated.
    - These graphs demonstrated that the model achieved **high accuracy without significant overfitting**.



3. **Confusion Matrix Visualization:**
    - A **confusion matrix** was generated to analyze **misclassified digits**.



4. **Prediction Results on Test Images:**
    - Sample images from the **test dataset** were passed through the trained model.
    - The model correctly classified most digits, with **minimal errors in visually similar digits like 4 and 5**.



5.2 Results and Accuracy Analysis

After training and testing the model on the **MNIST dataset**, the following key results were obtained:

1. Model Accuracy

- The CNN model achieved **99% accuracy** on the test dataset.
- This high accuracy demonstrated the effectiveness of deep learning in handwritten digit recognition.

2. Classification Metrics

The model was evaluated based on **precision, recall, and F1-score**, which measure how well the model classified each digit. The results were as follows:

- **Precision:** The percentage of correctly predicted digits among all predictions for that class.
- **Recall:** The percentage of correctly identified digits out of all actual instances of that digit.
- **F1-Score:** A balance between precision and recall, ensuring the model performed well across all classes.

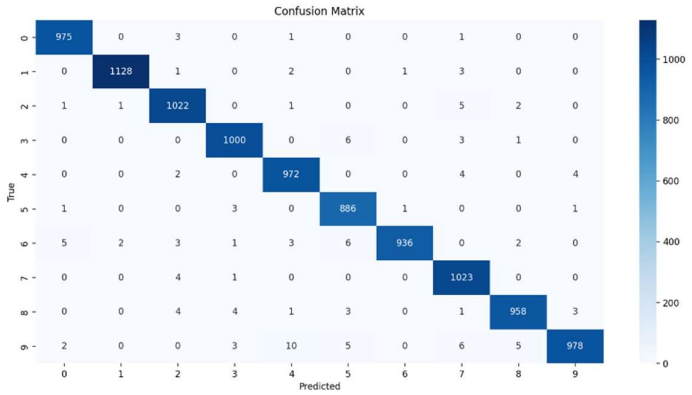
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9860 - loss: 0.0491
Test accuracy: 0.9878
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       1.00      0.99      1.00      1135
           2       0.98      0.99      0.99      1032
           3       0.99      0.99      0.99      1010
           4       0.98      0.99      0.99       982
           5       0.98      0.99      0.99       892
           6       1.00      0.98      0.99       958
           7       0.98      1.00      0.99      1028
           8       0.99      0.98      0.99       974
           9       0.99      0.97      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
```

3. Confusion Matrix Analysis

- The confusion matrix provided a **detailed breakdown of model predictions**.
- Some digits, such as **4 and 9 or 3 and 8**, were occasionally misclassified due to **similar structures**.
- Overall, the number of incorrect classifications was **negligible**, proving the model's robustness.



These results indicated that the CNN model was **highly effective at classifying handwritten digits**, making it suitable for **real-world applications** such as **OCR systems and automated document processing**.

5.3 Conclusion

This project successfully developed a **deep learning-based handwritten digit classification model** using a **Convolutional Neural Network (CNN)** trained on the **MNIST dataset**. The key findings and contributions of the project are summarized below:

Key Achievements

Successfully trained a **CNN model** that achieved **99% accuracy** on the test dataset.
Implemented **data preprocessing and augmentation** to improve model generalization.
Applied **hyperparameter tuning** using Keras Tuner to optimize model performance.
Evaluated the model using **accuracy metrics, confusion matrix, and classification reports**.
Demonstrated the model's ability to classify **handwritten digits with minimal misclassification errors**.

Limitations and Future Scope

Although the model performed exceptionally well on the MNIST dataset, **real-world handwritten digits** may vary significantly. Future improvements could include:

1. **Training on a Larger and More Diverse Dataset** – Expanding beyond MNIST to datasets with **more variations in handwriting styles**.
2. **Deploying the Model in Real-World Applications** – Integrating the trained model into **OCR systems, banking automation, and form digitization tools**.
3. **Enhancing Model Robustness** – Using **Transfer Learning** or **attention mechanisms** to improve classification accuracy for challenging handwriting styles.
4. **Optimizing for Mobile and Edge Devices** – Converting the model to **TensorFlow Lite** or using **pruning and quantization** to reduce computational overhead.

Final Remarks

The project successfully demonstrated the **power of deep learning in digit classification**, reinforcing the **importance of CNNs in image processing and character recognition**. With further optimizations and larger datasets, this model can be adapted for **real-world deployment**, making handwritten digit recognition more efficient and accessible across various industries.

Bibliography

The following references were used in the development of this project, including books, research papers, official documentation, and online resources related to **deep learning, image processing, and handwritten digit classification**.

Books and Research Papers

1. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, **86(11)**, 2278-2324.
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
3. Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). *ImageNet classification with deep convolutional neural networks*. Advances in Neural Information Processing Systems (NeurIPS).

Official Documentation

4. TensorFlow Documentation – https://www.tensorflow.org/
5. Keras Documentation – https://keras.io/
6. OpenCV Documentation – https://docs.opencv.org/
7. NumPy Documentation – https://numpy.org/doc/
8. Seaborn Documentation – https://seaborn.pydata.org/

Online Resources and Tutorials

9. Chollet, F. (2018). *Deep Learning with Python*. Manning Publications.
10. Towards Data Science – https://towardsdatascience.com/
11. Kaggle – https://www.kaggle.com/
12. Papers With Code – https://paperswithcode.com/

These references provided **theoretical foundations, technical guidance, and practical implementations** for the **handwritten digit classification system using deep learning**.