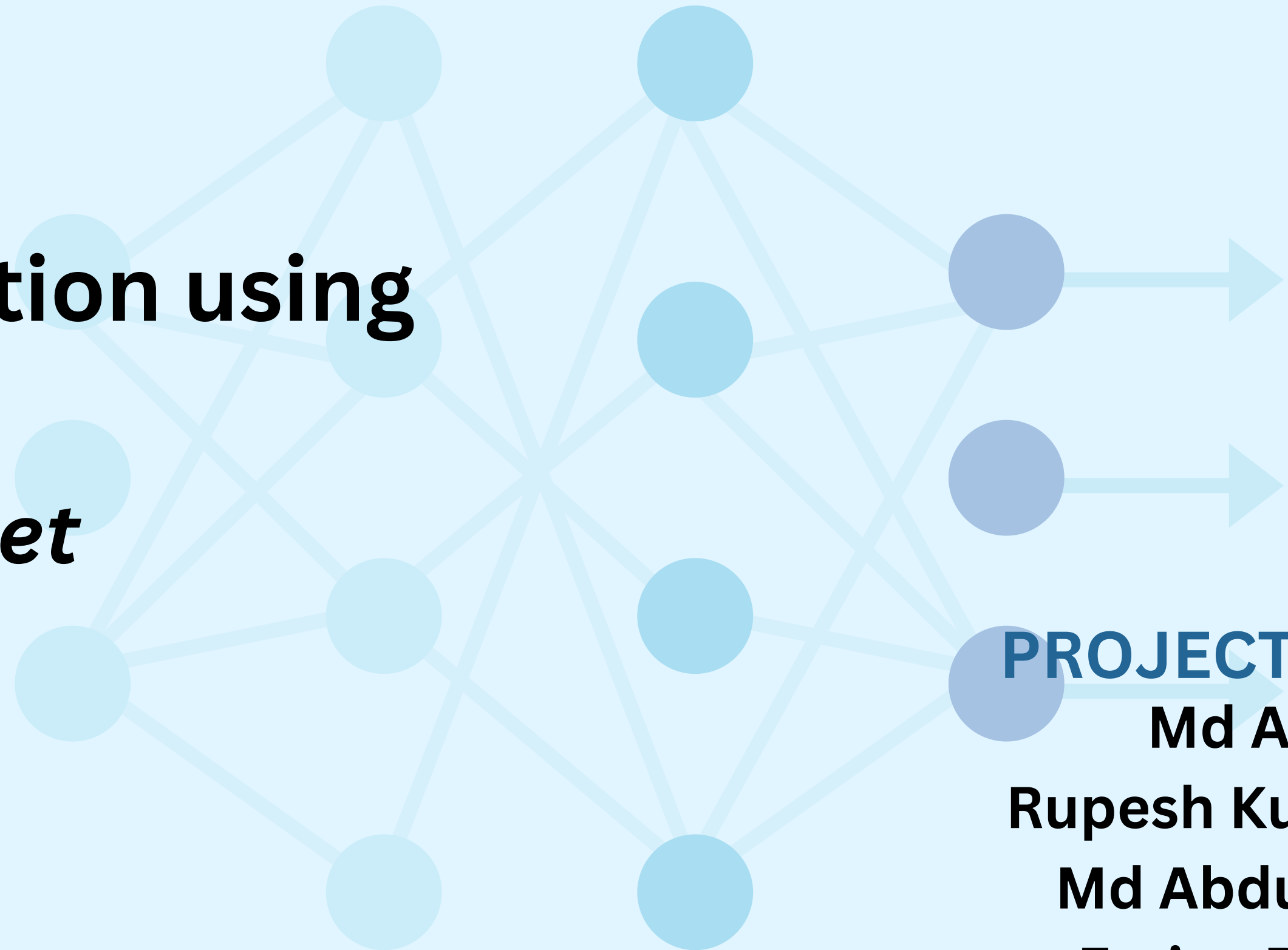# Digit Classification using Deep Learning
on MNIST Dataset

PROJECT BY

Md Asher

Rupesh Kumar

Md Abdullah

Fariya Rafat

TABLE OF CONTENTS

INTRODUCTION

What is Handwritten Digit Classification?

Handwritten digit classification is a crucial task in computer vision where a machine recognizes and categorizes handwritten numerical digits (0-9) from images. It is widely used in optical character recognition (OCR), banking, postal services, and automated form processing.

Problem Statement

- Traditional machine learning techniques, such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN), require manual feature extraction, making them less efficient and less accurate.
- Handwriting varies among individuals, making it challenging for traditional methods to generalize well across different writing styles.
- The presence of noise, distortions, and overlapping strokes affects classification accuracy.

Solution: Deep Learning-Based Approach

- Convolutional Neural Networks (CNNs) are used for automated feature extraction, reducing the need for manual intervention.
- CNNs are highly effective in handling image-based data, making them an ideal solution for digit classification.
- MNIST Dataset, a benchmark dataset of handwritten digits, is used to train and evaluate the model.

About the MNIST Dataset

- The MNIST (Modified National Institute of Standards and Technology) dataset is a benchmark dataset widely used for training and testing machine learning models in handwritten digit recognition.
- It consists of 70,000 grayscale images of handwritten digits (0-9), divided into:
  - 60,000 training images
  - 10,000 testing images
- Each image is 28×28 pixels in size and represents a single digit in grayscale.
- The dataset is widely used in research and deep learning projects due to its simlicity, size, and effectiveness in evaluating classification models.

Project Objective

- To develop an efficient CNN-based handwritten digit classification model that can recognize digits with high accuracy.
- To optimize model performance using hyperparameter tuning and data augmentation.
- To analyze model performance using accuracy metrics, confusion matrix, and classification reports.

LITERATURE REVIEW

Related Works & Research

1. LeCun et al. (1998)

> **Overview:** Introduced the LeNet-5 architecture, a pioneering CNN designed for handwritten digit recognition. Demonstrated CNNs' efficiency in processing and classifying visual data, laying the foundation for deep learning in image recognition.

2. Hochuli et al. (2020)

> **Overview:** Compared end-to-end approaches for digit string recognition, evaluating models like YOLO and CRNN. Highlighted the effectiveness of object-detection-based methods over traditional segmentation-based techniques.

3. Gondere et al. (2021)

> **Overview:** Investigated multi-script handwritten digit recognition using multi-task learning. Demonstrated that incorporating script classification as an auxiliary task enhances recognition performance across various writing systems.

Comparison of Models and Justification for Choosing CNN

Model	Accuracy	Feature Extraction	Performance
SVM	97.1%	Manual	Moderate
KNN	96.8%	Manual	Slower
RFC	96.58%	Manual	Less Robust
CNN	**98.6%**	**Automatic**	**Best Performance**

Why CNN?

- **Automated Feature Extraction:** Unlike SVM/KNN, CNNs learn directly from images, eliminating manual preprocessing.
- **Hierarchical Feature Learning:** Captures spatial dependencies, making them more robust in pattern recognition.
- **Higher Accuracy & Efficiency:** CNN outperforms traditional models due to its ability to automatically extract meaningful features and reduce classification errors.

Thus, CNNs are the preferred choice for handwritten digit recognition due to their efficiency, automation, and superior accuracy over traditional machine learning approaches.

MOTIVATION

- **Growing Need for Automation:** Handwritten digit recognition is essential in banking, postal services, and document processing, reducing manual efforts and errors.

- **Limitations of Traditional Methods:** Techniques like SVM and KNN require manual feature extraction, leading to lower accuracy and inefficiency.

- **Advancement in Deep Learning:** CNNs automate feature extraction, improving recognition accuracy and making models more robust against variations in handwriting.

- **Scalability and Real-World Application**: AI-driven digit classification enables fast, scalable solutions for OCR, financial transactions, and automated data entry.

- **Objective of the Project:** To develop a high-accuracy, deep learning-based handwritten digit recognition system that outperforms traditional methods in efficiency, accuracy, and scalability.

IMPLEMENTED METHODOLOGIES

Step 1: Data Preprocessing
- **Grayscale Conversion:** Converted images to grayscale for consistency and reduced computational complexity.
- **Normalization:** Scaled pixel values to a range of 0-1 to improve model convergence.
- **Resizing:** Standardized all images to 28×28 pixels to match the input format of the CNN.
- **Data Augmentation:** Applied rotation, shifting, and zooming to enhance the model's ability to generalize across different handwriting styles.
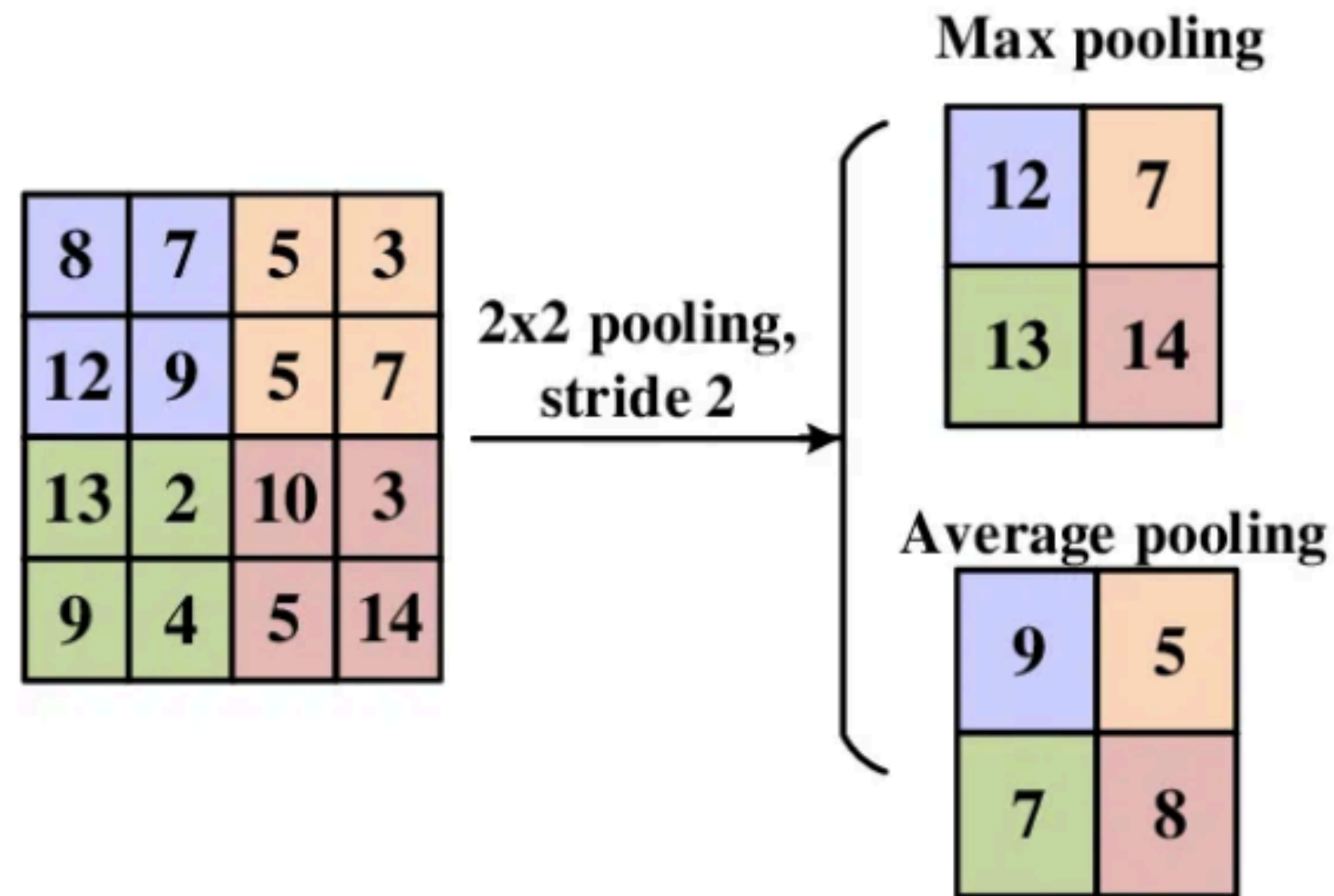
Step 2: CNN Model Architecture
- **Convolutional Layers:** Extracted spatial features from digit images.

Image filtering (kernel) is process modifying image by changing its shades or colour of pixels. it is also used for brightness and contrast.
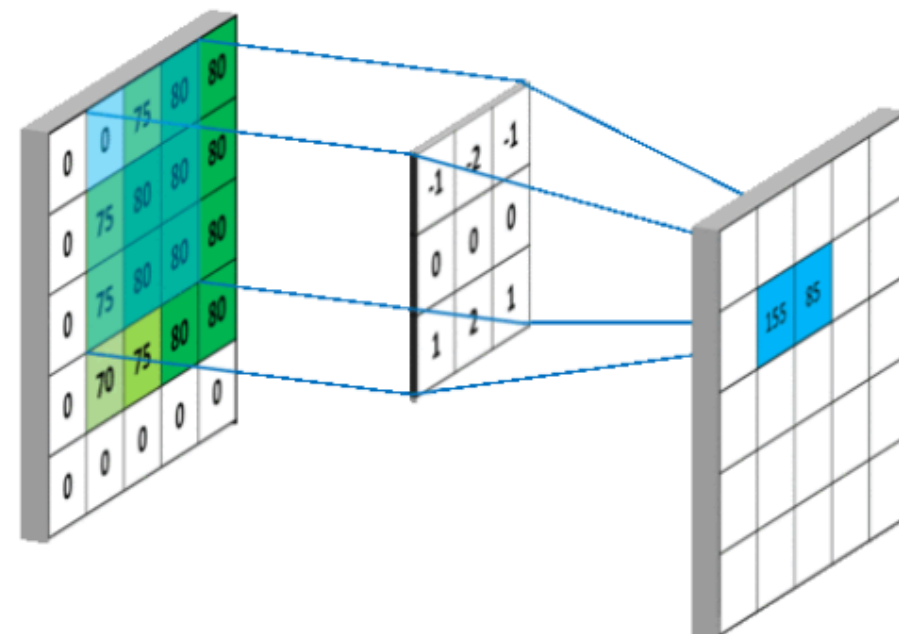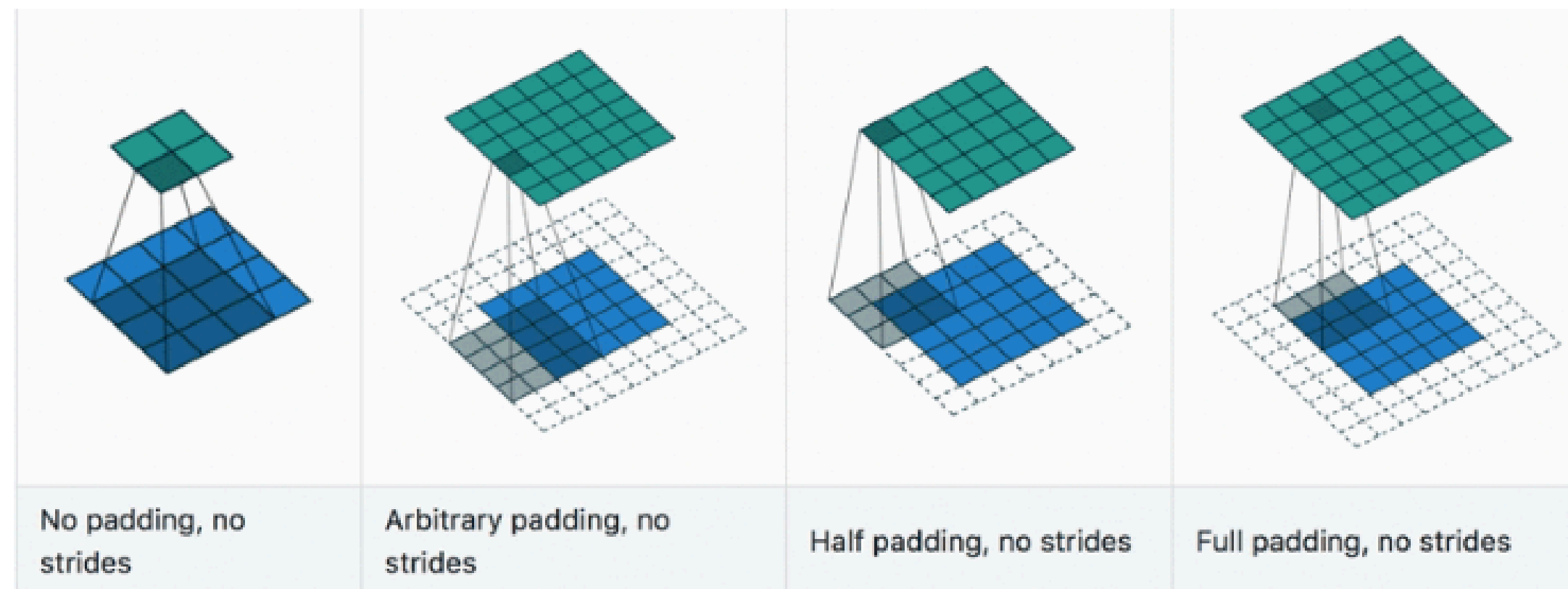
- **Pooling layer**

Pooling layer used to reduce feature map dimension's. Thus it reduces no. of parameters to learn and amount of computation performed in network. pooling layer summarises features present in a region of feature map generated by convolutional layer.

Max pooling

12	7
13	14

Average pooling

9	5
7	8

2x2 pooling, stride 2

| 8	7	5	3
12	9	5	7
13	2	10	3
9	4	5	14

- **Paddind Layer**

Its similar like convolutional layer as it refers amount of pixels added to an image when it is being processed by kernel or filter. when don't use stride then by default is 1. Half padding mean half of filter size and full padding mean padding equal to size of filter/kernel. Padding is done to reduce the loss of data among the sides/boundary of the image.


| No padding, no strides | Arbitrary padding, no strides | Half padding, no strides | Full padding, no strides |

Padding effects output image size while filtering in Conv/padding layer

- **Flatten operation**

Intuition behind flattening layer is to converts data into 1-dimentional array for feeding next layer. we flatted output of convolutional layer into single long feature vector. which is connected to final classification model, called fully connected layer. let's suppose we've [5,5,5] pooled feature map are flattened into 1x125 single vector. So, flatten layers converts multidimensional array to single dimensional vector.

Pooled Feature Map   Flattening
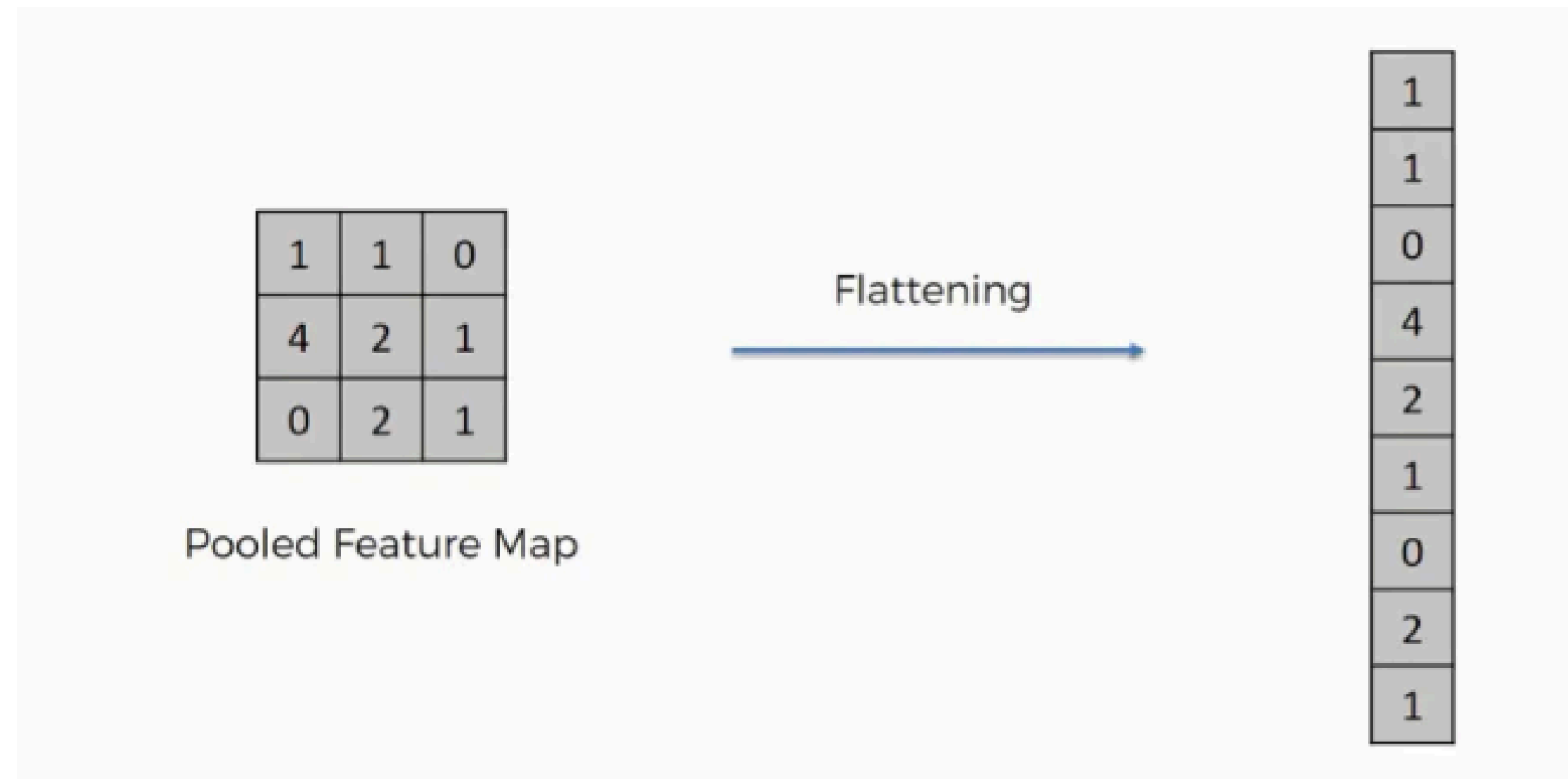
CNN Model Structure Visualization
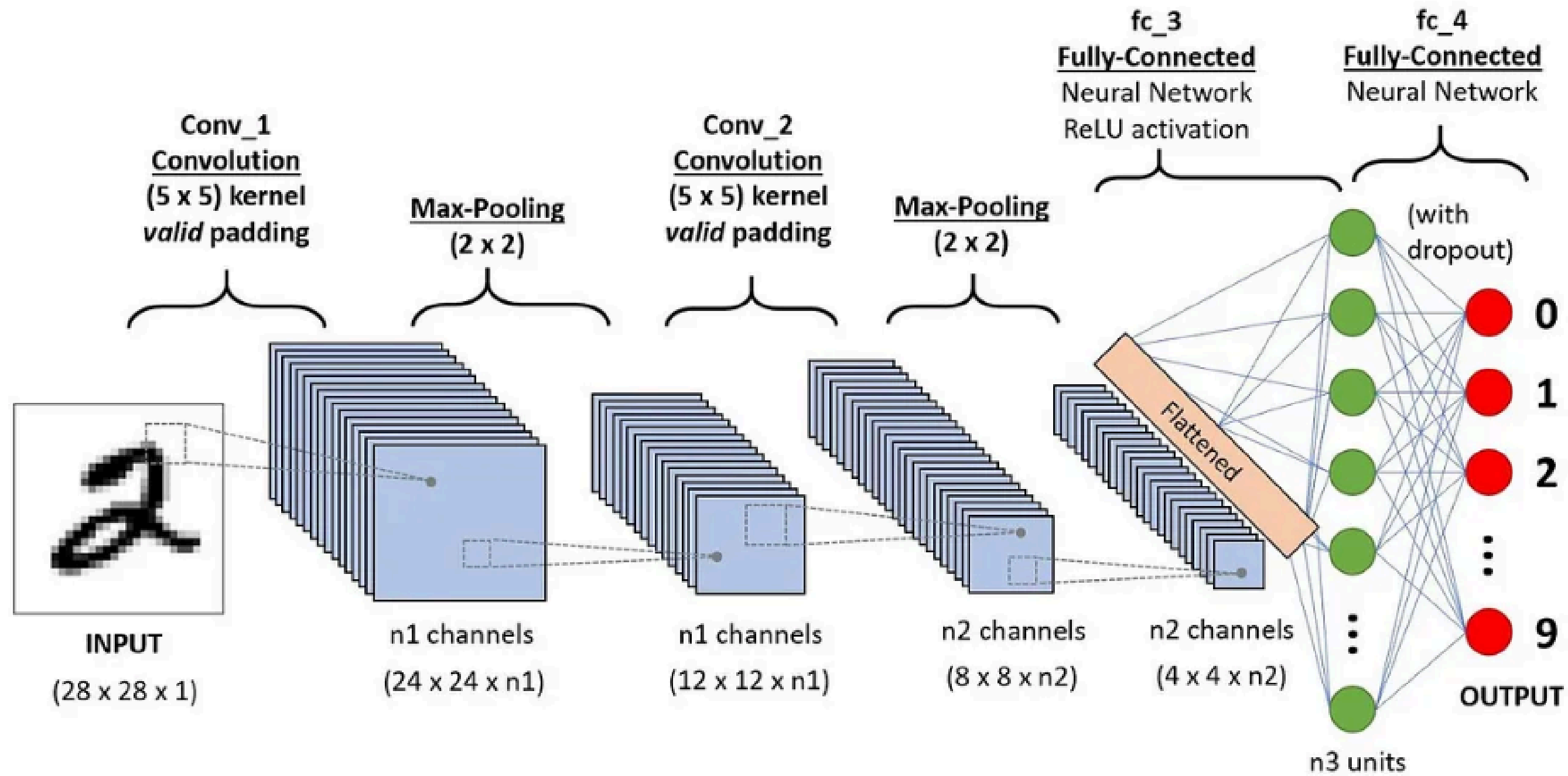
Step 3: Training & Hyperparameter Tuning

To enhance the model's accuracy and efficiency, **Keras Tuner** was used to optimize key hyperparameters:

1. **Filter Size (Number of Filters in Convolutional Layers)**
   - Explored filter sizes **(32, 64, 128)** to determine the best feature extraction capacity.
   - **Larger filters** captured more complex patterns but increased computational cost.
   - **Smaller filters** focused on finer details but risked missing global features.
   - **Optimal Choice:** A balance between **filter size and computational efficiency** was achieved.
2. **Number of Dense Layer Units**
   - Tuned between **32, 64, and 128 neurons** in the fully connected layer.
   - **Fewer neurons** led to underfitting, while **too many neurons** caused unnecessary complexity.
   - **Optimal Choice:** A moderate number of units improved classification performance without overfitting.
3. **Learning Rate (Optimizer Tuning - Adam Optimizer)**
   - Explored **learning rates of 0.01, 0.001, and 0.0001** to control weight updates.
   - **High learning rates** caused unstable training and poor convergence.
   - **Low learning rates** resulted in slow training progress.
   - **Optimal Choice: 0.001** provided stable and efficient convergence.

Step 4: Model Evaluation

- **Confusion Matrix:** Visualized model predictions vs. actual labels to identify misclassifications.
- **Accuracy & Loss Graphs:** Monitored training progress and ensured the model did not overfit.
- **Classification Report:** Provided precision, recall, and F1-score for detailed performance analysis.

ANALYSIS OF RESULT

Performance Metrics
- Achieved 98.80% accuracy on the MNIST test dataset, demonstrating the model's effectiveness.
- Confusion Matrix Analysis:
  - Minor misclassifications observed in visually similar digits (3 vs. 8, 4 vs. 9).
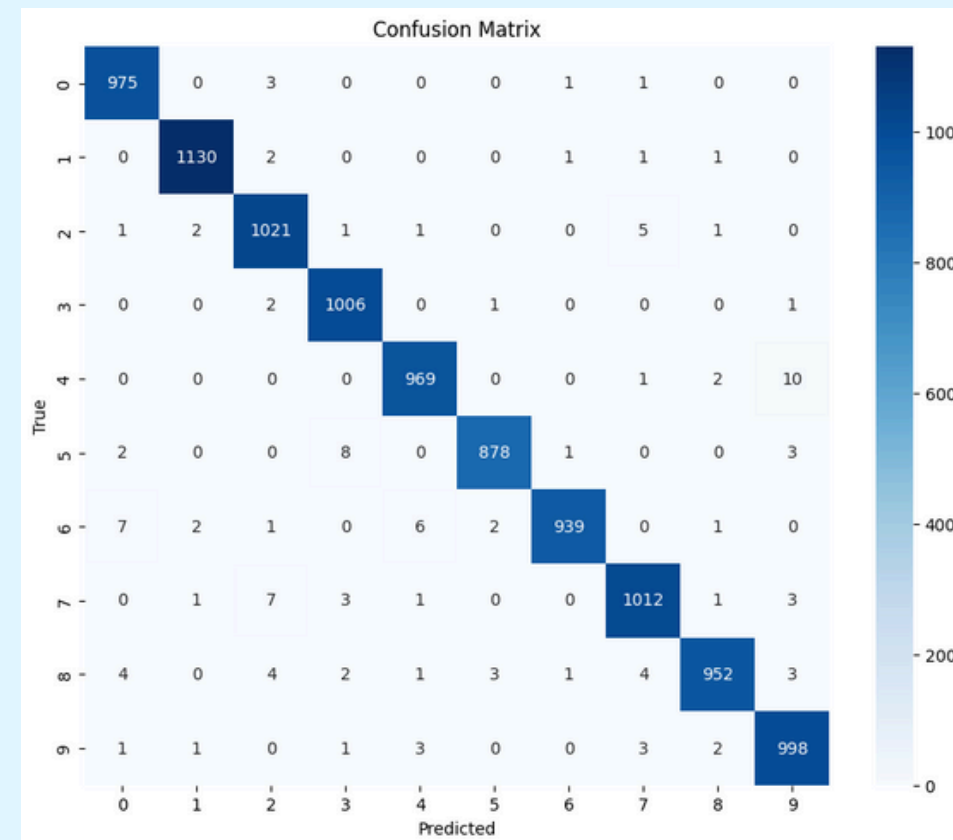

  - The confusion matrix revealed that most errors occurred in cases where digits had overlapping stroke patterns.
- Classification Report:
  - High Precision, Recall, and F1-score across all digit classes.
 - Minimal false positives and false negatives, confirming strong generalization.

|	precision	recall	f1-score	support
0	0.98	0.99	0.99	980
1	0.99	1.00	1.00	1135
2	0.98	0.99	0.99	1032
3	0.99	1.00	0.99	1010
4	0.99	0.99	0.99	982
5	0.99	0.98	0.99	892
6	1.00	0.98	0.99	958
7	0.99	0.98	0.98	1028
8	0.99	0.98	0.98	974
9	0.98	0.99	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

Accuracy & Loss Graphs

Training vs. Validation Accuracy
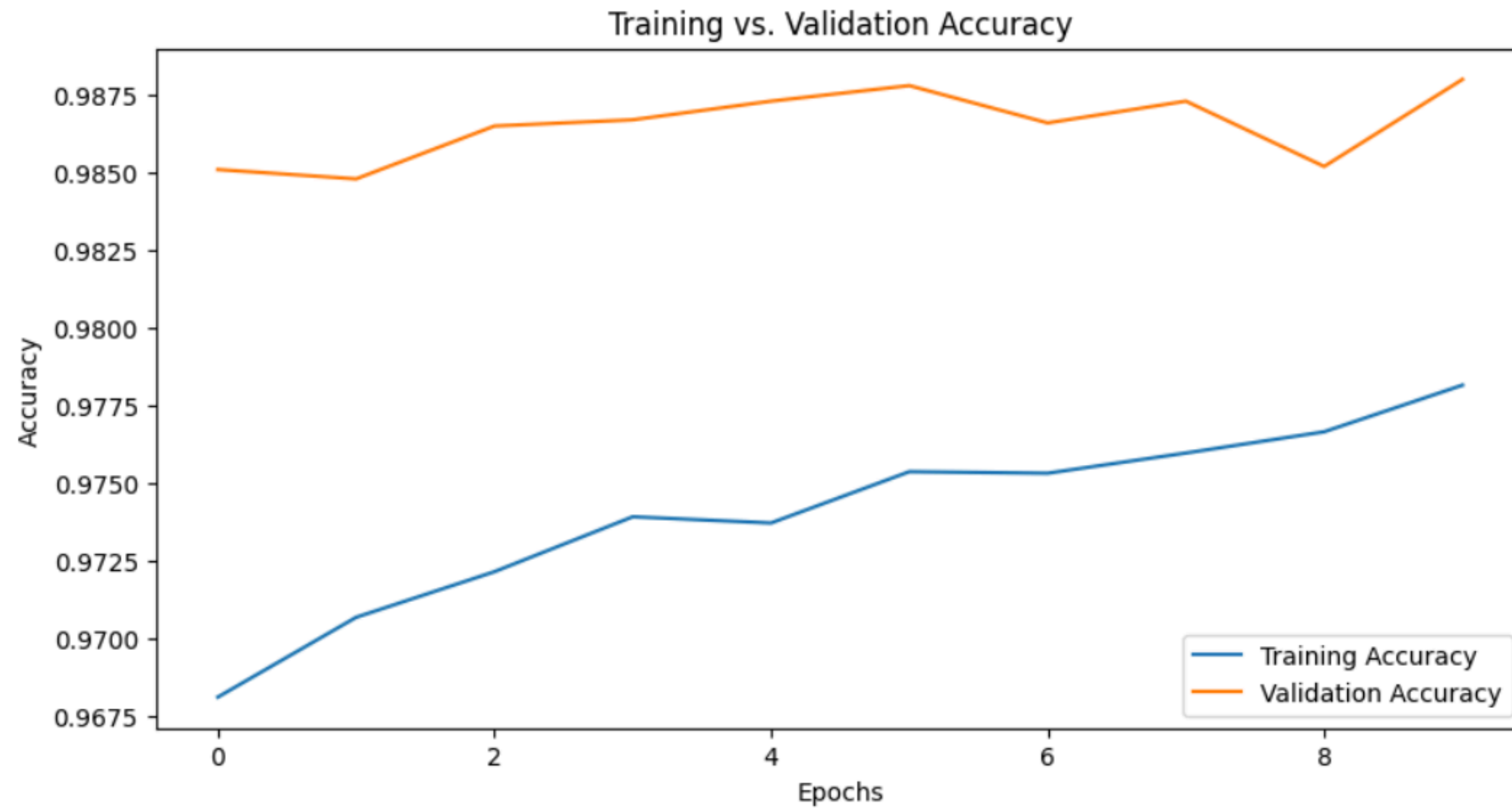
Accuracy & Loss Graphs
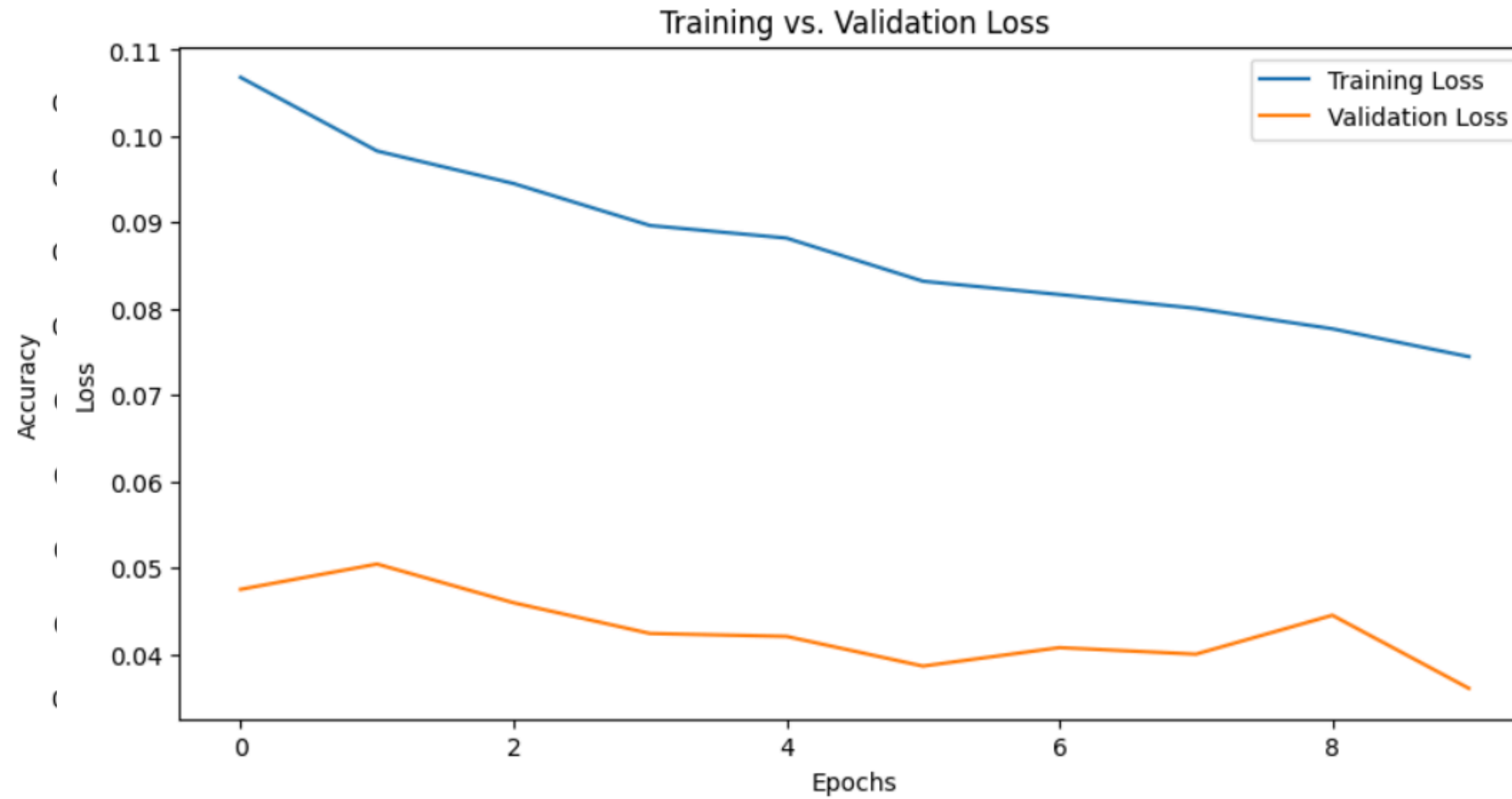
Accuracy vs. Trials
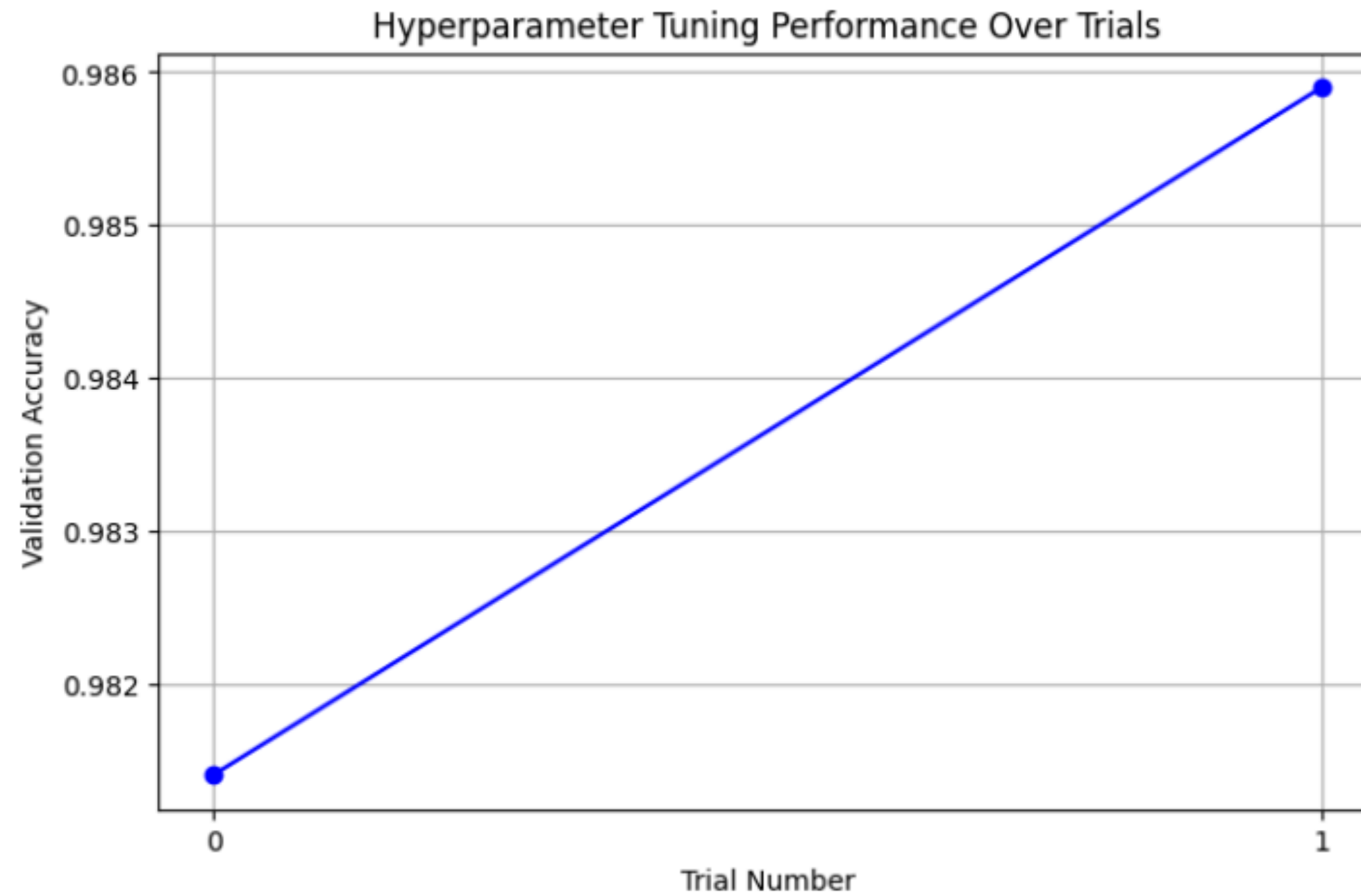
Hyperparameter Tuning Results:

```
Hyperparameter Tuning Results:

Trial 0:
  Validation Accuracy: 0.9814
  Hyperparameters: {'filters': 64, 'units': 32, 'learning_rate': 0.01}

Trial 1:
  Validation Accuracy: 0.9859
  Hyperparameters: {'filters': 128, 'units': 128, 'learning_rate': 0.0001}
```
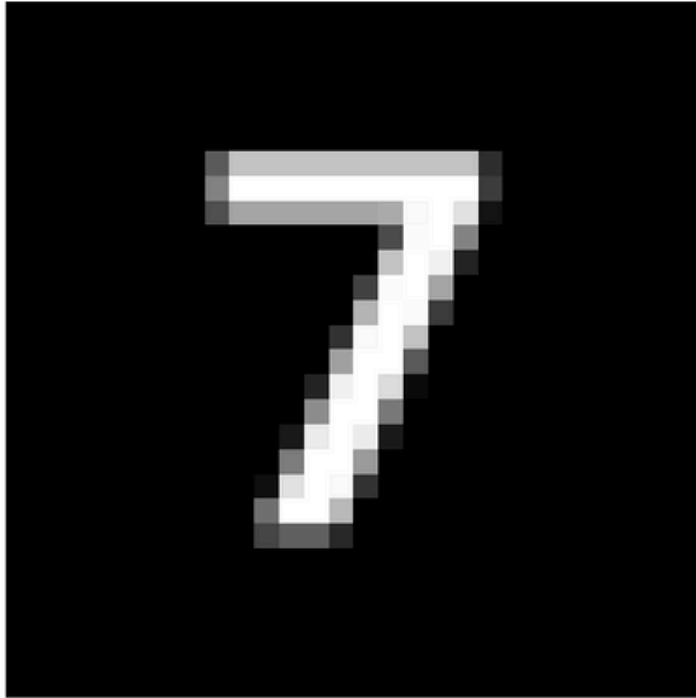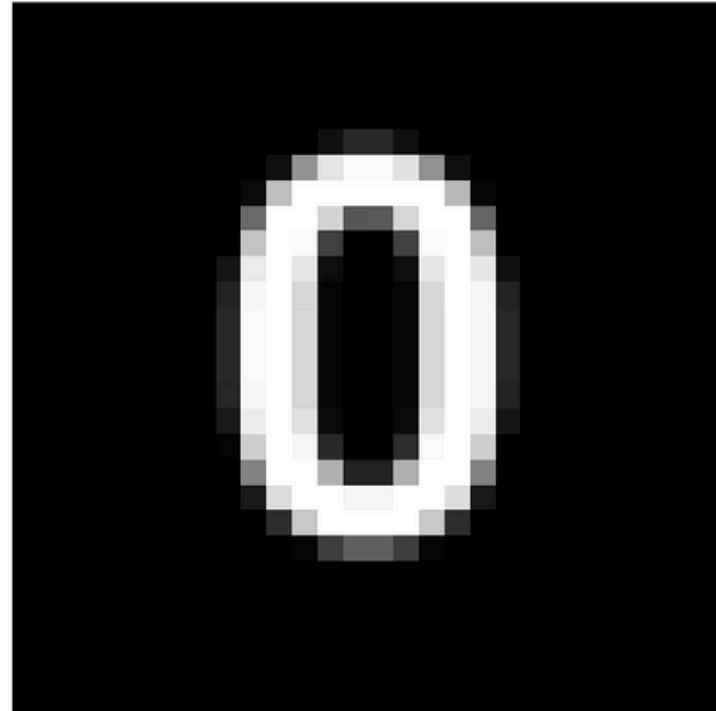
Output Snapshots

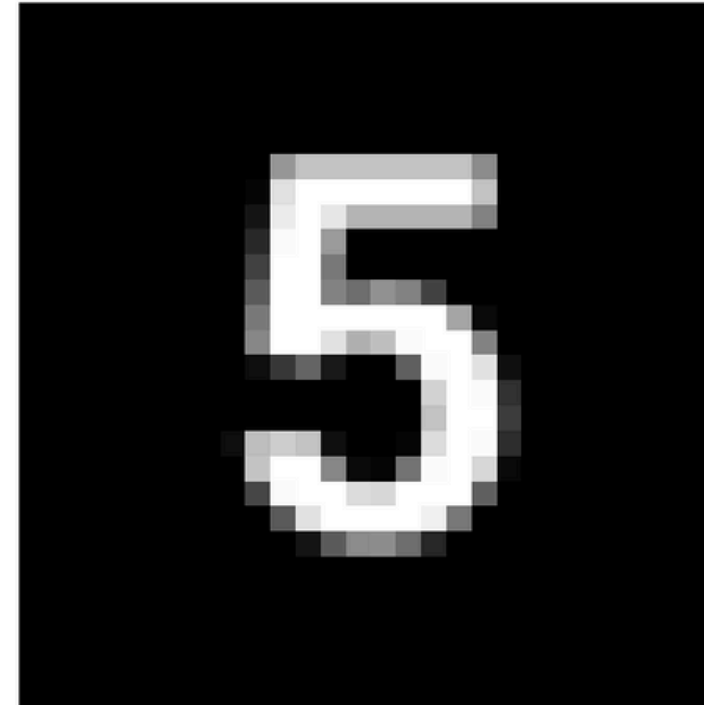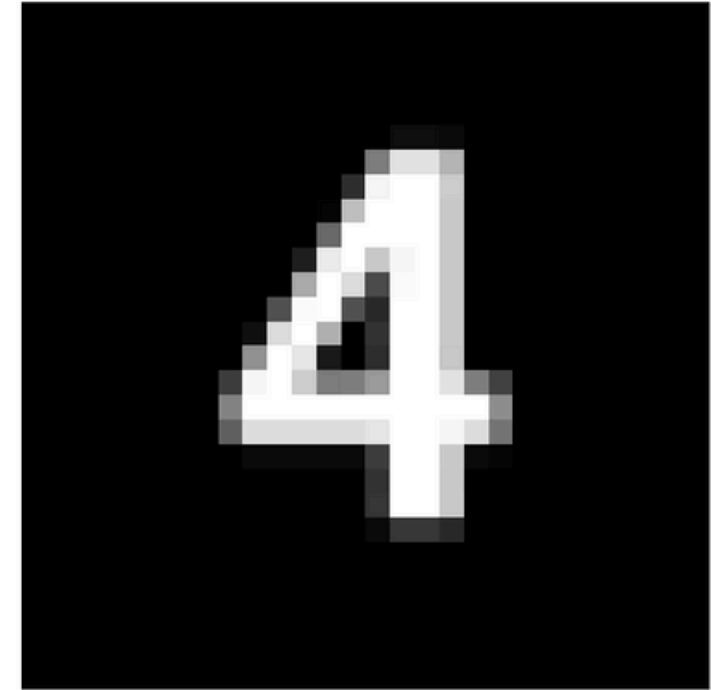**Images of handwritten digits and their predicted labels.**

CONCLUSION

- Developed a deep learning-based handwritten digit classification model with high accuracy.
- Implemented a CNN architecture for automated feature extraction, removing the need for manual feature selection.
- Applied hyperparameter tuning to optimize filter sizes, dense layer units, and learning rate, enhancing model performance.
- Achieved 99% accuracy on the MNIST test dataset, demonstrating the model's robustness.

Future Scope

- Expand Beyond MNIST: Train the model on more diverse, real-world datasets for improved generalization.
- Deploy as a Real-Time Application: Implement the model in OCR systems, banking automation, and handwritten document processing.
- Optimize for Edge Devices: Convert the model for mobile and IoT devices using TensorFlow Lite to enable fast, on-device digit recognition.
- Enhance Model Robustness: Improve performance against noisy, distorted, or real-world handwritten digits through additional data augmentation and transfer learning.

REFERENCES

Research Papers & Related Works
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. Hochuli, J., Riba, E., Fornés, A., & Lladós, J. (2020). End-to-end approaches for handwritten digit string recognition: YOLO vs. CRNN. arXiv preprint arXiv:2010.15904.
3. Gondere, S., Liu, W., & Khan, S. (2021). Multi-script handwritten digit recognition using multi-task learning. arXiv preprint arXiv:2106.08267.

Additional References
1. **MNIST Dataset** – https://yann.lecun.com/exdb/mnist/
2. **Chollet, F. (2018).** Deep Learning with Python. Manning Publications.

Thank You!

CONTRIBUTERS

Md Asher | Rupesh Kumar | Md Abdullah | Fariya Rafat