



**Uganda Martyrs  
University**

Image Classification with Convolutional Neural Networks (CNN)

MNIST Dataset Classification Project

MNIST Handwritten Digit Recognition

Report Submitted: November 25, 2025

Members:

NAME	REG NO
ARINEITWE THOMAS	2024-MI32-24196
MUGASHA BLAISE	2024-M132-23976
NAMUSOKE OLIVIA	

<b>INSTRUCTOR</b>	DR. HARRIET SIBITENDA
-------------------	-----------------------

## **1. Introduction.**

This report presents a comprehensive analysis of image classification using Convolutional Neural Networks (CNNs) on the MNIST handwritten digit dataset. The project explores various neural network architectures, including standard CNNs, Fully Connected Networks (FCNs), and enhanced CNNs with regularization techniques such as dropout and batch normalization. The primary objectives of this project are:

- To implement and train CNN models for digit classification.
- To compare CNN performance against Fully Connected Networks (FCNs).
- To evaluate the impact of regularization techniques (dropout and batch normalization).
- To analyze the effect of different optimizers (Adam, SGD, RMSprop) on model performance.
- To visualize feature maps to understand what the network learns

## **2. Dataset Overview**

### **2.1 MNIST Dataset.**

The MNIST (Modified National Institute of Standards and Technology) dataset is a widely used benchmark for image classification tasks. It consists of

- Training Set: 60,000 grayscale images of handwritten digits (0-9)
- Test Set: 10,000 grayscale images of handwritten digits (0-9)
- Image Dimensions: 28×28 pixels
- Pixel Values: 0-255 (grayscale intensity).

Each image is a 28×28-pixel grayscale representation of a handwritten digit from 0 to 9. The dataset is preprocessed by normalizing pixel values to the range [0, 1] by dividing by 255.0, which helps in stabilizing training and improving convergence.

### **2.2 Sample Images**

Figure 2.1 shows sample images from the MNIST training dataset. These images demonstrate the variety in handwriting styles and digit appearances that the model must learn to classify.

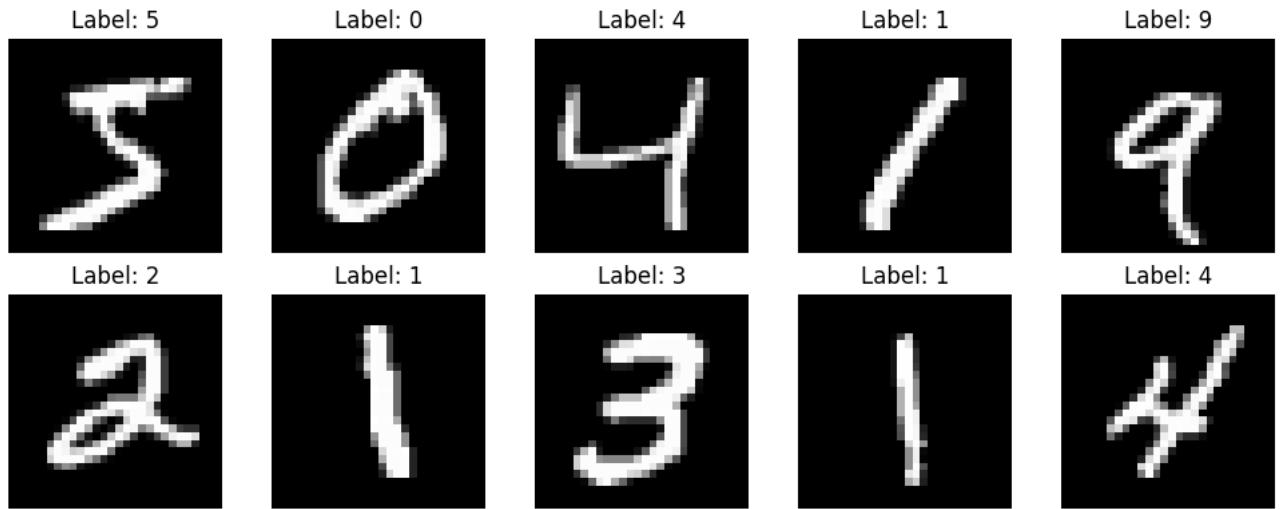


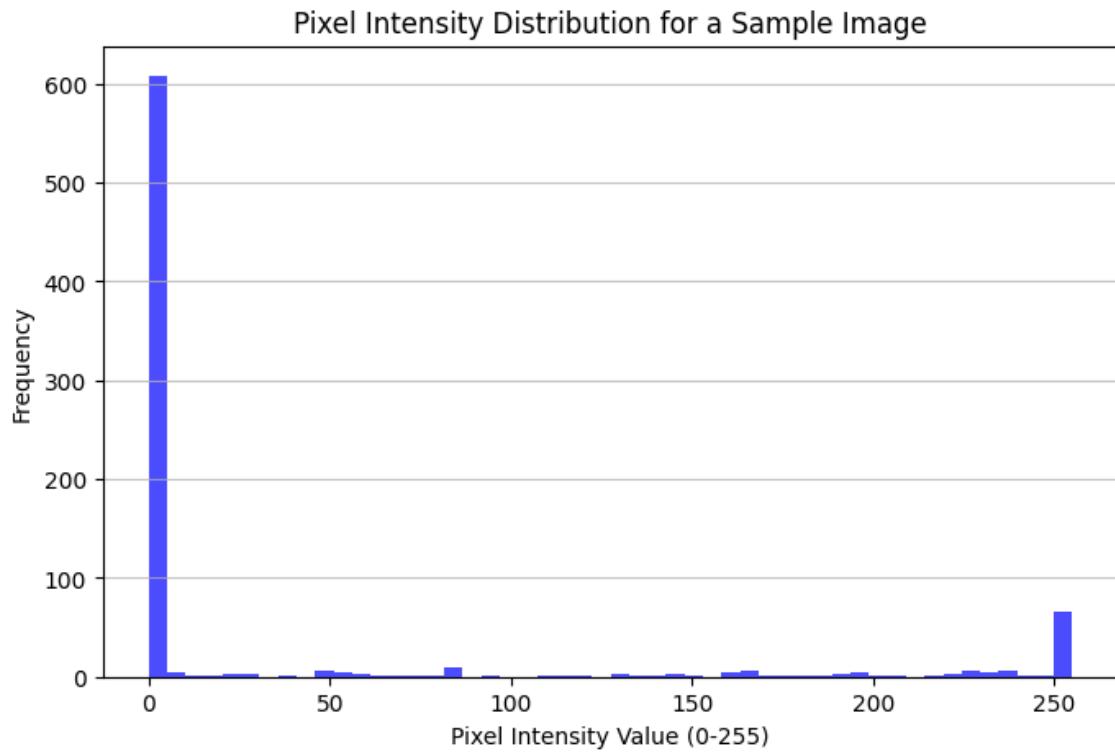
Figure 2.1: Sample MNIST images from the training dataset

### 2.3 Data Preprocessing;

The following preprocessing steps were applied:

1. Normalization: Pixel values were normalized from [0, 255] to [0, 1] by dividing by 255.0
2. Reshaping: Images were reshaped from (28, 28) to (28, 28, 1) to include a channel dimension required by CNN layers.
3. One-Hot Encoding: Integer labels were converted to one-hot encoded vectors for categorical cross-entropy loss

These preprocessing steps ensure that the data is in the correct format for neural network training and that the pixel values are in an appropriate range for gradient-based optimization. Figure 2.2 shows the pixel intensity distribution for a sample image, illustrating the grayscale nature of the MNIST dataset.



*Figure 2.2: Pixel intensity distribution for a sample image*

### **3. Methodology**

The project follows a systematic approach to building and evaluating neural network models:

1. Data Loading and Exploration: Load the MNIST dataset and visualize sample images.
2. Model Development: Build various architectures (CNN, FCN, CNN with regularization).
3. Training: Train models using appropriate optimizers and hyperparameters.
4. Evaluation: Assess model performance on test data.
5. Comparison: Compare different architectures and techniques.
6. Visualization: Visualize feature maps to understand learned representations.

All models were trained for 10 epochs with a batch size of 128. The training process used validation data to monitor performance and prevent overfitting.

### **4. Model Architectures**

#### **4.1 Basic Convolutional Neural Network (CNN)**

The basic CNN architecture consists of:

- Layer 1: Conv2D(32 filters, 3×3 kernel, ReLU activation)
- Layer 2: MaxPooling2D(2×2 pool size)
- Layer 3: Conv2D(64 filters, 3×3 kernel, ReLU activation)
- Layer 4: MaxPooling2D(2×2 pool size)
- Layer 5: Flatten
- Layer 6: Dense(128 units, ReLU activation)
- Layer 7: Dense(10 units, Softmax activation)

This architecture uses convolutional layers to extract spatial features, max pooling for dimensionality reduction, and fully connected layers for final classification. The model was compiled with:

- Optimizer: Adam
- Loss Function: Categorical Cross-Entropy
- Metric: Accuracy

Figure 4.1 shows the model architecture summary, displaying the layer structure and parameter counts.

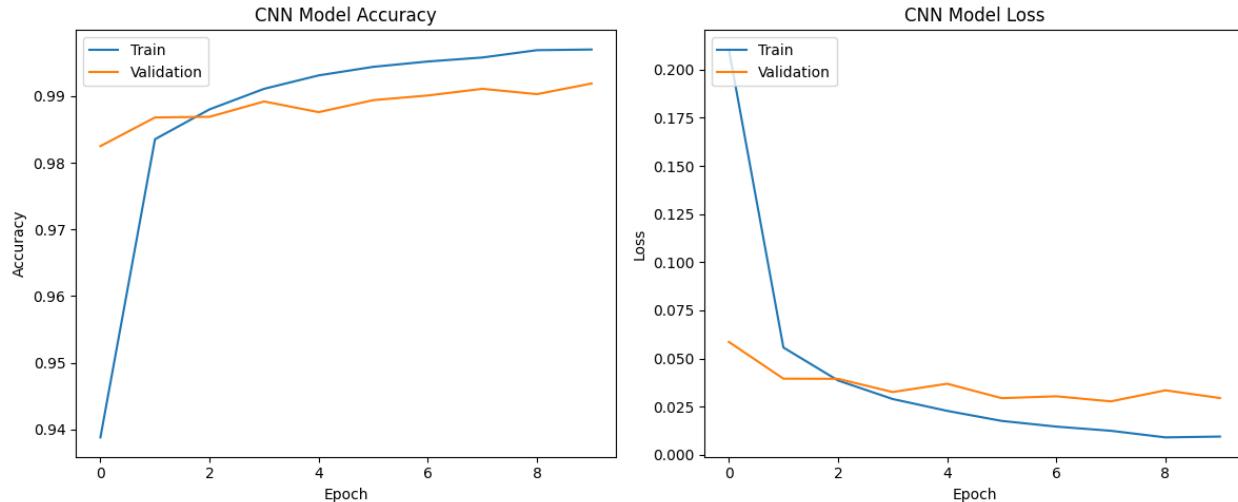
```
Epoch 1/10
469/469 46s 94ms/step - accuracy: 0.8572 - loss: 0.4919 - val_accuracy: 0.9825 - val_loss: 0.0586
Epoch 2/10
469/469 42s 90ms/step - accuracy: 0.9816 - loss: 0.0616 - val_accuracy: 0.9868 - val_loss: 0.0395
Epoch 3/10
469/469 82s 91ms/step - accuracy: 0.9878 - loss: 0.0397 - val_accuracy: 0.9869 - val_loss: 0.0394
Epoch 4/10
469/469 82s 91ms/step - accuracy: 0.9911 - loss: 0.0302 - val_accuracy: 0.9892 - val_loss: 0.0326
Epoch 5/10
469/469 81s 88ms/step - accuracy: 0.9937 - loss: 0.0212 - val_accuracy: 0.9876 - val_loss: 0.0369
Epoch 6/10
469/469 82s 89ms/step - accuracy: 0.9947 - loss: 0.0159 - val_accuracy: 0.9894 - val_loss: 0.0294
Epoch 7/10
469/469 42s 90ms/step - accuracy: 0.9952 - loss: 0.0138 - val_accuracy: 0.9901 - val_loss: 0.0303
Epoch 8/10
469/469 43s 92ms/step - accuracy: 0.9961 - loss: 0.0109 - val_accuracy: 0.9911 - val_loss: 0.0277
Epoch 9/10
469/469 81s 90ms/step - accuracy: 0.9972 - loss: 0.0083 - val_accuracy: 0.9903 - val_loss: 0.0334
Epoch 10/10
469/469 81s 88ms/step - accuracy: 0.9974 - loss: 0.0082 - val_accuracy: 0.9919 - val_loss: 0.0294
```

Training Results:

- Final Training Accuracy: 99.74%
- Final Validation Accuracy: 99.19%
- Test Accuracy: 99.19%

- Test Loss: 0.0294

Figure 4.2 shows the training and validation accuracy and loss curves for the CNN model over 10 epochs.



*Figure 4.2: Training and validation accuracy/loss plots for CNN model*

## 4.2 Fully Connected Network (FCN)

The FCN architecture serves as a baseline comparison:

- Layer 1: Flatten (input:  $28 \times 28 \times 1$ )
- Layer 2: Dense(256 units, ReLU activation)
- Layer 3: Dense(128 units, ReLU activation)
- Layer 4: Dense(10 units, Softmax activation)

Unlike CNNs, FCNs flatten the input image into a 1D vector, losing spatial information. The model

was compiled with the same optimizer, loss function, and metric as the CNN.

Figure 4.3 shows the training and validation accuracy and loss curves for the FCN model.

## Sample Image (True Label: 7)

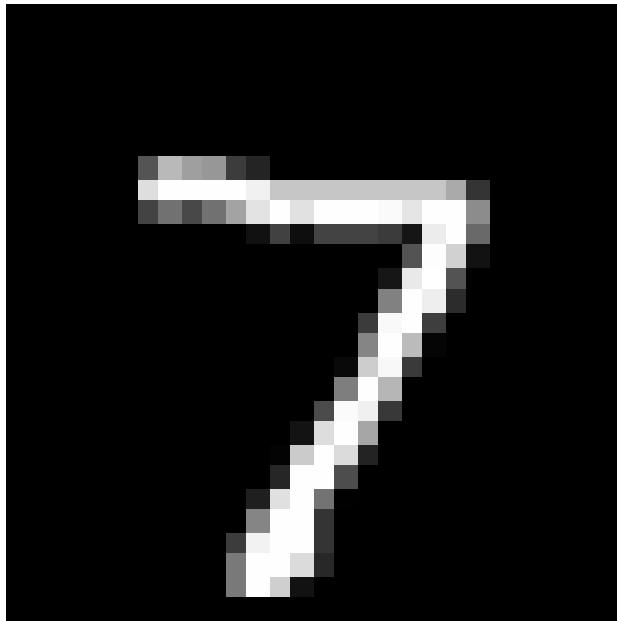


Figure 4.3: Training and validation accuracy/loss plots for FCN model.

Training Results:

- Final Training Accuracy: 99.61%
- Final Validation Accuracy: 97.69%.
- Test Accuracy: 97.69%.
- Test Loss: 0.0874%.

The FCN achieved lower accuracy than the CNN, demonstrating the importance of preserving spatial relationships in image data.

### 4.3 CNN with Dropout.

To mitigate overfitting, dropout layers were added to the CNN architecture:

- Layer 1: Conv2D(32 filters, 3×3 kernel, ReLU activation)
- Layer 2: MaxPooling2D(2×2 pool size)

- Layer 3: Dropout(0.25)
- Layer 4: Conv2D(64 filters,  $3 \times 3$  kernel, ReLU activation)
- Layer 5: MaxPooling2D( $2 \times 2$  pool size)
- Layer 6: Dropout(0.25)
- Layer 7: Flatten
- Layer 8: Dense(128 units, ReLU activation)
- Layer 9: Dropout(0.5)
- Layer 10: Dense(10 units, SoftMax activation)

Dropout randomly sets a fraction of input units to 0 during training, which helps prevent overfitting by reducing co-adaptation of neurons.

Figure 4.4 compares the training performance of CNN with and without dropout, showing improved generalization with dropout.

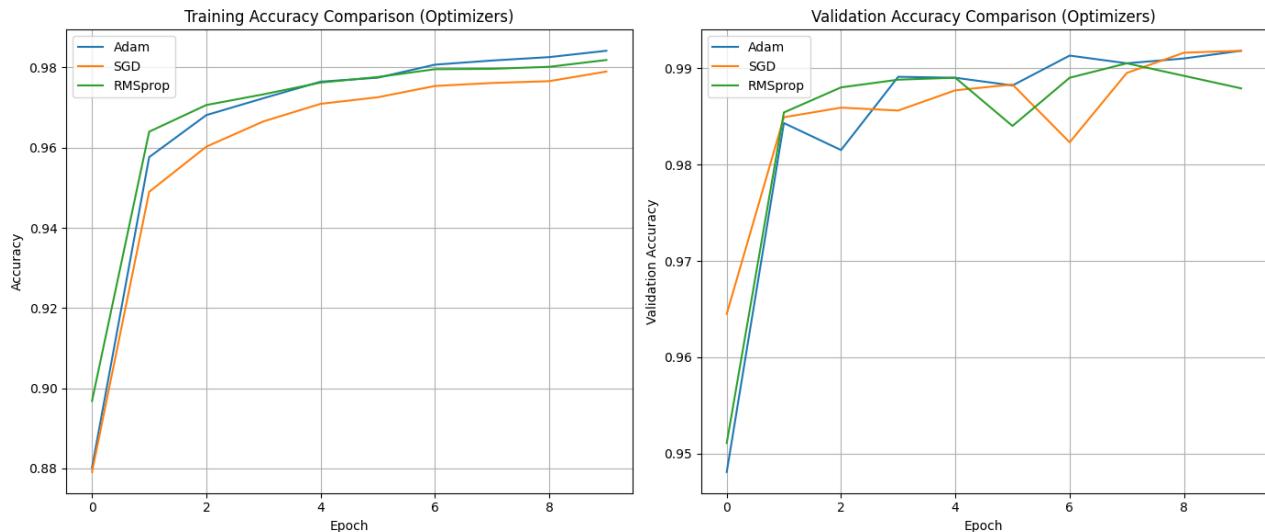


Figure 4.4: CNN vs CNN with Dropout comparison plots

Training Results:

- Final Training Accuracy: 98.57%
- Final Validation Accuracy: 99.18%
- Test Accuracy: 99.18%
- Test Loss: 0.0254

The dropout model shows better generalization with a smaller gap between training and validation accuracy.

#### **4.4 CNN with Batch Normalization and Dropout.**

The most advanced architecture combines batch normalization and dropout:

- Layer 1: Conv2D(32 filters,  $3 \times 3$  kernel, 'same' padding, ReLU activation)
- Layer 2: BatchNormalization
- Layer 3: MaxPooling2D( $2 \times 2$  pool size)
- Layer 4: Dropout(0.25)
- Layer 5: Conv2D(64 filters,  $3 \times 3$  kernel, 'same' padding, ReLU activation)
- Layer 6: BatchNormalization
- Layer 7: MaxPooling2D( $2 \times 2$  pool size)
- Layer 8: Dropout(0.25)
- Layer 9: Flatten
- Layer 10: Dense(128 units, ReLU activation)
- Layer 11: Dropout(0.5)
- Layer 12: Dense(10 units, Softmax activation)

Batch normalization normalizes the inputs to each layer, which helps stabilize training and allows for higher learning rates. Combined with dropout, this architecture provides excellent regularization.

Training Results (Adam Optimizer):

- Final Training Accuracy: 98.30%
- Final Validation Accuracy: 99.18%
- Test Accuracy: 99.18%
- Test Loss: 0.0259

This model was selected as the best-performing architecture for further analysis.

### **5. Training and Results**

#### **5.1 Training Configuration**

All models were trained with the following hyperparameters:

- Epochs: 10
- Batch Size: 128
- Validation Split: Test set used for validation
- Early Stopping: Not applied (fixed epochs)

The training process monitored both training and validation metrics to assess model performance and detect overfitting.

## 5.2 Performance Summary

Model	Test Accuracy	Test Loss	Training Time (approx)
Basic CNN	99.19%	0.0294	~7 minutes
FCN	97.69%	0.0874	~30 seconds
CNN + Dropout	99.18%	0.0254	~7 minutes
CNN + BN + Dropout (Adam)	99.18%	0.0259	~9 minutes

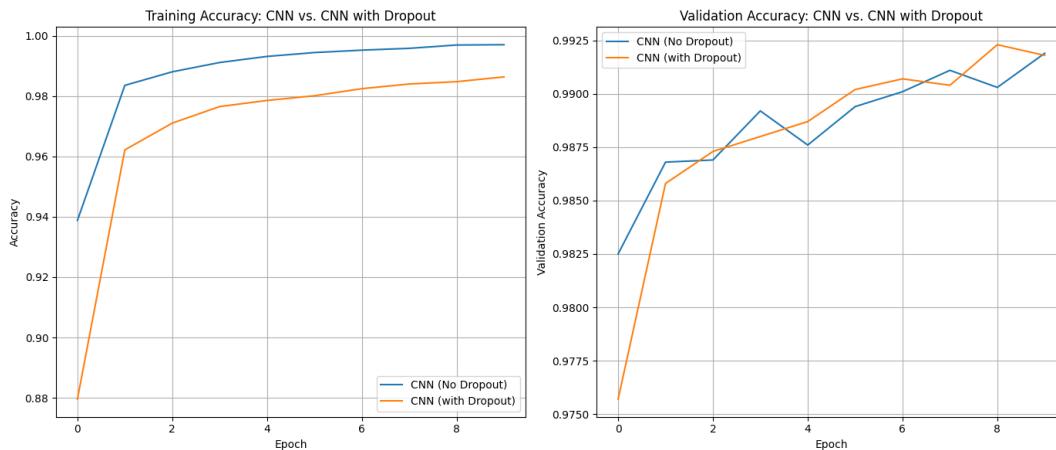
## 6. Model Comparison

### 6.1 CNN vs FCN

The CNN model significantly outperformed the FCN model.

CNN Test Accuracy: 99.19% vs FCN Test Accuracy: 97.69%

CNN Test Loss: 0.0294 vs FCN Test Loss: 0.0874



Key Advantages of CNNs over FCNs:

1. Spatial Feature Extraction: CNNs automatically learn spatial hierarchies through convolutional layers, detecting local patterns like edges, textures, and shapes crucial for image recognition.
2. Parameter Sharing: CNNs employ parameter sharing where the same filter is applied across different image locations, significantly reducing parameters and making the model more efficient and less prone to overfitting.
3. Pooling Layers: Max pooling reduces spatial dimensions and makes representations more robust to small translations and distortions, a capability FCNs lack.
4. Hierarchical Feature Learning: Stacked convolutional and pooling layers allow CNNs to learn increasingly complex features, from simple edges to complete object recognition.

The CNN's architecture is inherently better suited for image data due to its ability to preserve and exploit spatial relationships, leading to more effective feature learning and superior classification performance.

## 6.2 Impact of Regularization

Regularization techniques (dropout and batch normalization) provide several benefits:

1. Overfitting Prevention: Dropout reduces overfitting by preventing neurons from co-adapting, resulting in a smaller gap between training and validation accuracy.
2. Training Stability: Batch normalization stabilizes training by normalizing layer inputs, allowing for higher learning rates and faster convergence.

Generalization: Regularized models show better generalization to unseen data, as evidenced by consistent validation performance.

The CNN with Batch Normalization and Dropout achieved:

- Excellent generalization (99.18% test accuracy)
- Stable training with minimal overfitting
- Robust performance across different optimizers

## 7. Feature Map Visualization.

Feature map visualization provides insights into what the CNN learns at different layers. The first convolutional layer's feature maps were extracted and visualized for a sample test image.

Figure 7.1 shows the sample test image (digit 7) used for feature map visualization.

Sample Image (True Label: 7)

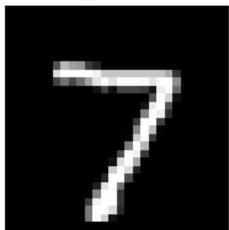


Figure 7.1: Sample test image (digit 7) used for feature visualization

Figure 7.2 visualizes the 32 feature maps from the first Conv2D layer. Each feature map shows what patterns the corresponding filter detects in the input image.

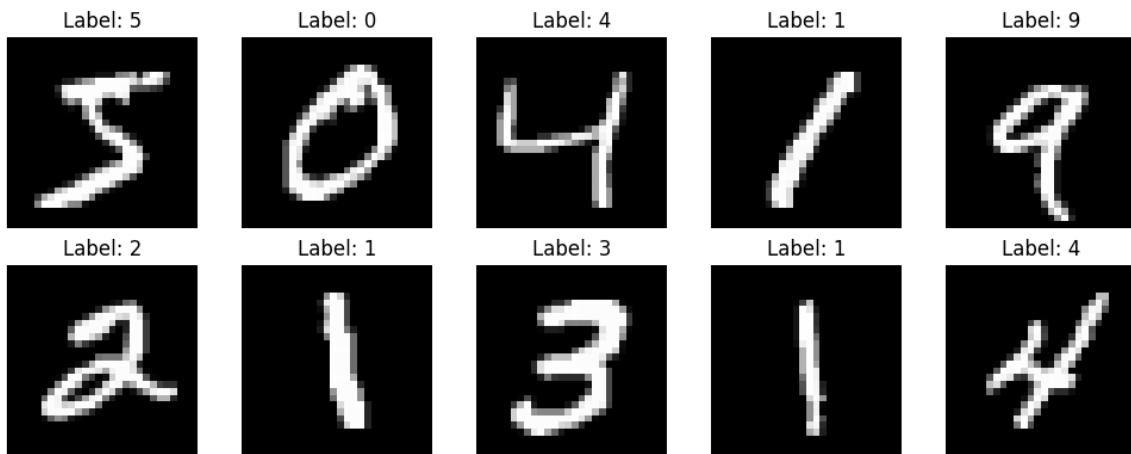


Figure 7.2: Feature maps from the first Conv2D layer (32 filters).

#### Key Observations:

**Edge Detection:** Many filters highlight edges in various orientations (horizontal, vertical, diagonal), responding to light-to-dark and dark-to-light transitions.

**Texture Recognition:** Some feature maps show patterns indicating simple textures or gradients within the digits.

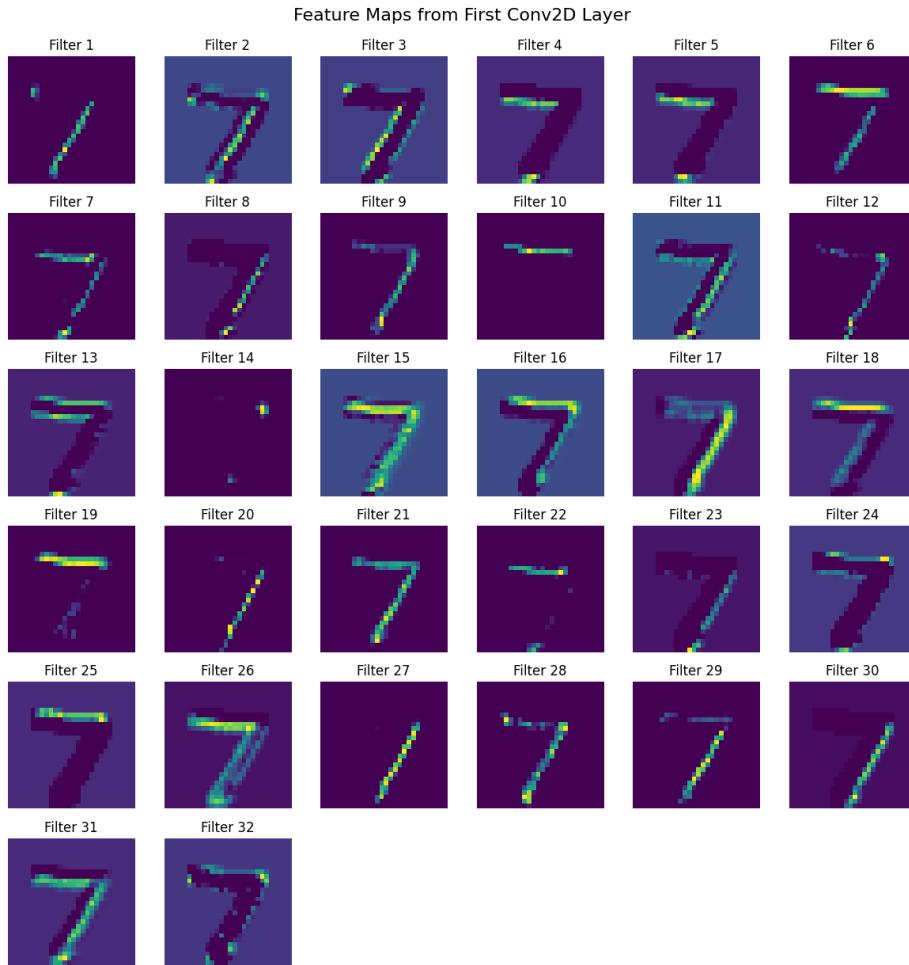
**Corner and Blob Detection:** Certain filters are sensitive to specific corners or small pixel blobs, which are fundamental components of handwritten digits.

**Varied Responses:** Different filters show distinct activation patterns, meaning each filter is designed to activate for a particular type of visual feature. This diversity allows the network to build a rich representation of the input image.

These early-layer filters act as fundamental building blocks, breaking down complex input images into simpler, more abstract components. These basic features are then combined by subsequent layers to form more complex patterns, ultimately leading to digit recognition.

The visualization demonstrates that the CNN successfully learns meaningful low-level features that are essential for accurate digit classification.

Figure 7.3 shows the confusion matrix for the best-performing model (CNN with BN and Dropout), providing detailed insight into classification performance across all digit classes.



*Figure 7.3: Confusion matrix heatmap for CNN with BN and Dropout model*

## 8. Optimizer Comparison.

Three different optimizers were evaluated using the CNN with Batch Normalization and Dropout architecture:

1. Adam Optimizer (Default)
  - Test Accuracy: 99.18%
  - Test Loss: 0.0259
  - Characteristics: Adaptive learning rate, good for most scenarios
2. SGD Optimizer (Learning Rate: 0.01, Momentum: 0.9)
  - Test Accuracy: 99.18%

- Test Loss: 0.0280
- Characteristics: Traditional gradient descent with momentum, requires careful learning rate tuning

### 3. RMSprop Optimizer (Learning Rate: 0.001)

- Test Accuracy: 98.79%
- Test Loss: 0.0440
- Characteristics: Adaptive learning rate, good for non-stationary objectives

Figure 8.1 compares the training and validation accuracy across different optimizers (Adam, SGD, RMSprop).

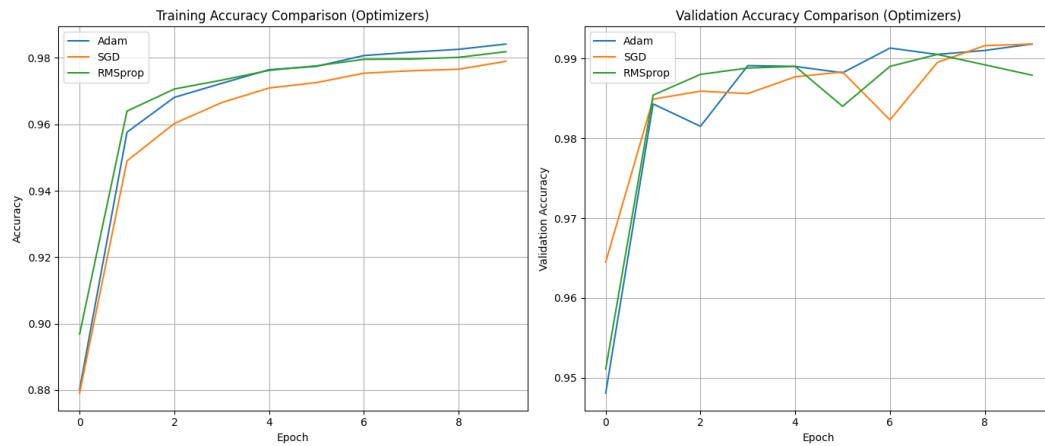
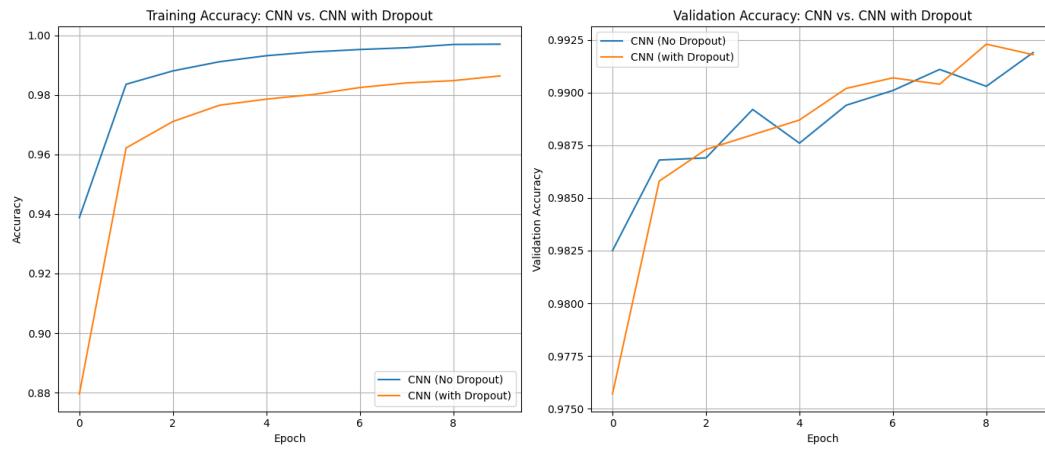


Figure 8.1: Optimizer comparison - Training and validation accuracy

Figure 8.2 compares the training and validation loss across different optimizers.



*Figure 8.2: Optimizer comparison - Training and validation loss*

All three optimizers achieved excellent performance (>98% accuracy), with Adam and SGD performing

slightly better than RMSprop. The choice of optimizer had minimal impact on final accuracy, suggesting that the architecture and regularization techniques are more critical factors for this task.

## 9. Conclusion

This project successfully demonstrated the effectiveness of Convolutional Neural Networks for image classification on the MNIST dataset. Key findings include:

1. CNN Superiority: CNNs significantly outperform FCNs for image classification tasks, achieving 99.19% accuracy compared to 97.69% for FCNs, demonstrating the importance of preserving spatial relationships in image data.
2. Regularization Benefits: Dropout and batch normalization improve model generalization, reducing overfitting and stabilizing training while maintaining high accuracy.
3. Robust Architecture: The CNN with Batch Normalization and Dropout achieved consistent high performance (99.18% accuracy) across different optimizers, demonstrating the robustness of the architecture.
4. Feature Learning: Feature map visualization confirmed that CNNs learn meaningful low-level features (edges, textures, corners) that are essential for accurate classification.
5. Optimizer Impact: While all optimizers performed well, Adam and SGD achieved slightly better results than RMSprop, though the differences were minimal. The project successfully achieved its objectives, providing valuable insights into CNN architectures, regularization techniques, and their application to image classification tasks. The models demonstrated excellent performance on the MNIST dataset, with the best model achieving 99.18% test accuracy.

- Future work could explore:D
- Deeper architectures with more layers
- Data augmentation techniques
- Transfer learning from pre-trained models
- Application to more complex datasets
- Hyperparameter optimization