Cairo University, Faculty of Computers
and Artificial Intelligence

Supervised Learning Project – CNN & ANN on MNIST

AI322-Supervised learning

Name	ID
Rana Esmail Zekery	20220131
Ahmed Abdelaziz	20220025
Ammar Mohamed Mahmoud	20220216
Marwan Osama	20220324

Deep Learning Hyperparameter Exploration: MNIST Project

Step 1: Setup and Baseline Models

We used the mnist data set which has 60000 training samples and 10,000 testing samples that is pre-divided , has images of handwritten digits from 0 to 9.

1.2 Artificial Neural Network (ANN)

We used a simple ANN model:

Flatten Layer : to convert the 28*28 image to 784 length vector.

Hidden Dense Layer (128 neurons): Activated using ReLU.

Output Dense Layer (10 neurons): Activated using Softmax for multi-class classification.

We trained the model using:

- **Optimizer:** Adam
- **Loss Function:** Categorical Crossentropy
- **Epochs:** 5
- **Batch Size:** 16
- **Validation Split:** 10% of the training data
- **Shuffling:** Enabled (Keras default)

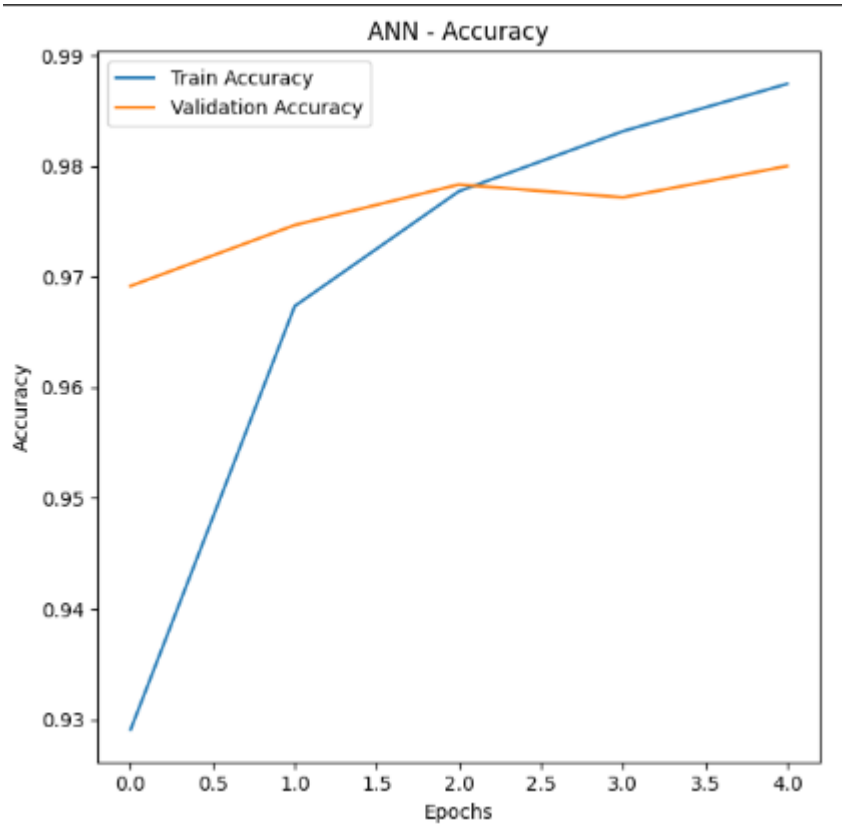
The training and evaluation results were:

- **Accuracy on Test Set:** 97.52%
- **Accuracy in First 5 Epochs**:

1. Epoch 1: 88.51%
2. Epoch 2: 96.54%
3. Epoch 3: 97.93%
4. Epoch 4: 98.39%
5. Epoch 5: 98.83%

- **Total Parameters:** 101,770
- **Avg Training Time per Epoch**: 14.76 seconds
- **Training Time:** 73.78 seconds
- **Test Time:** 0.84 seconds

- **Observations**: The model performed excellently on the test set, with high accuracy in all epochs. There was a slight decrease in accuracy after the first epoch, but it stabilized and continued to



improve in the following epochs.

1.3 Support Vector Machine (SVM)

Due to the computational cost of training an SVM on the full dataset, we used a subset of 10,000 samples for training and 2,000 for testing. The images were flattened to 784-length vectors and normalized between 0 and 1.

The SVM was trained with:

- **Kernel:** Radial Basis Function (RBF)
- **C:** 1.0
- **Gamma:** Scale

Results for the SVM model:

- **Accuracy on Subset Test Set:** 94.45%
- **Number of Support Vectors:** 3,744
- **Training Time:** 15.73 seconds
- **Test Time:** 9.20 seconds

Why ANN was faster?

The ANN achieved good accuracy quickly, showing steady improvements in the first few epochs. However, it took longer to train compared to SVM due to its more complex architecture, involving deeper layers and a larger number of parameters.

Step 2: Base CNN + Epoch Tuning

we built a CNN model with 3 convolutional layers and 2 max-pooling layers. We used the ReLU activation function and the SGD optimizer with a learning rate of 0.01 and momentum of 0.9. We also explored different epoch values (10–25) to identify the best epoch count for optimal performance.

2.1 CNN Architecture

The CNN model consists of the following layers:

- **Conv2D Layer 1**: 32 filters, kernel size (3, 3), ReLU activation
- **MaxPooling2D Layer 1**: Pool size (2, 2)
- **Conv2D Layer 2**: 64 filters, kernel size (3, 3), ReLU activation
- **MaxPooling2D Layer 2**: Pool size (2, 2)
- **Conv2D Layer 3**: 64 filters, kernel size (3, 3), ReLU activation
- **Flatten Layer**: Converts the 3D feature maps into a 1D vector
- **Dense Layer**: Output layer with 10 units (for MNIST classes), softmax activation function

B) Epoch Tuning

We tested different epoch values (10, 15, 20, and 25) to determine which one leads to the best model performance.

The results for each epoch count:

2.2 Epoch Tuning Results

1. Base CNN (10 Epochs)

- **Test Accuracy**: 99.16%
- **Number of Layers**: 7
- **Parameters**: 61,514
- **Training Time**: 75.21 seconds
- **Test Time**: 1.36 seconds

2. Base CNN (15 Epochs)

- **Test Accuracy**: 98.88%
- **Number of Layers**: 7
- **Parameters**: 61,514

- **Training Time**: 128.30 seconds
- **Test Time**: 1.35 seconds

3. Base CNN (20 Epochs)

- **Test Accuracy**: 99.25%
- **Number of Layers**: 7
- **Parameters**: 61,514
- **Training Time**: 186.57 seconds
- **Test Time**: 1.35 seconds

4. Base CNN (25 Epochs)

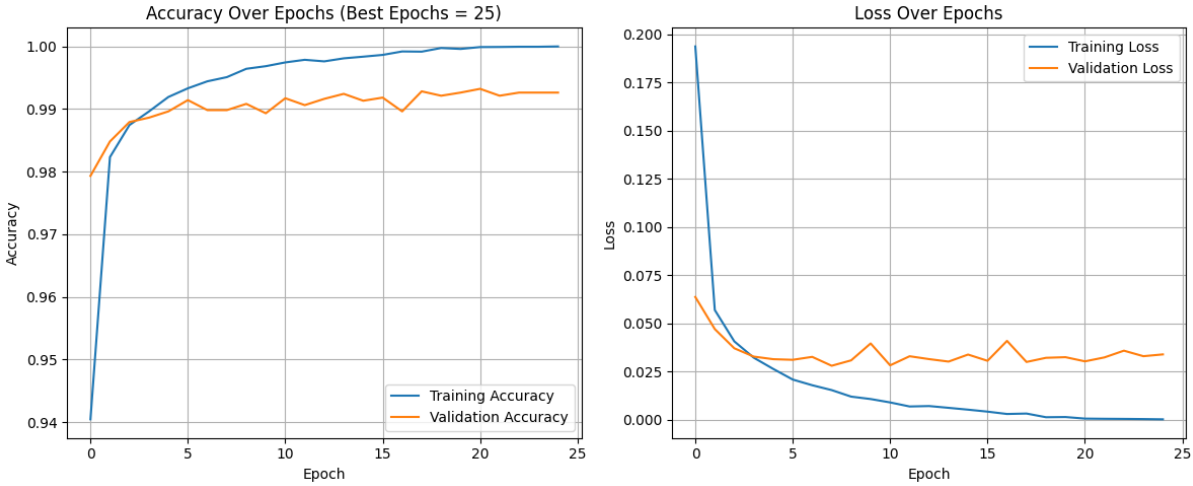
- **Test Accuracy**: 99.26%
- **Number of Layers**: 7
- **Parameters**: 61,514
- **Training Time**: 211.05 seconds
- **Test Time**: 0.74 seconds

Conclusion

After experimenting with various epoch values (10, 15, 20, 25), the highest test accuracy of **99.26%** was achieved using **25 epochs**. Although training time increased with more epochs, the improvement in performance made 20 epochs the optimal choice based on accuracy alone.

Accuracy and Loss Plots

To visualize the model's performance over time, the following plots show the accuracy and loss curves for the **best model (25 epochs)**:



1. **Accuracy Plot**: This graph shows the progression of both training and validation accuracy over the course of the training epochs.

2. **Loss Plot**: This graph displays the decrease in training and validation loss over time.

Step 3: Learning Rate Testing

Find the best learning rate for your base CNN model by testing different values and seeing how they affect performance.

What You Did:

1. **Fixed the number of epochs** (25 in our case).
2. **Tested 3 different learning rates:**
    - `0.01`
    - `0.001`
    - `0.0001`
3. **For each learning rate:**
    - Trained the CNN for 25 epochs.
    - Measured test accuracy and training time.
    - Logged the results.
    - Compared to find which learning rate gave the highest accuracy.

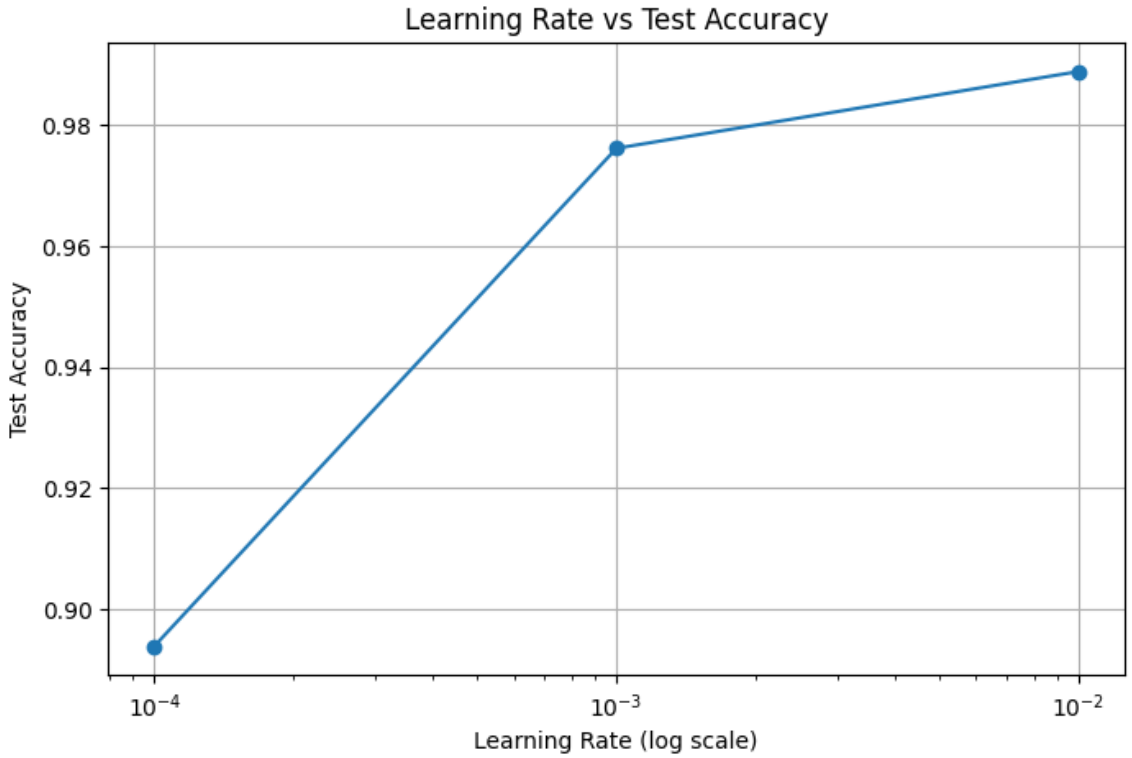
Results:

- **LR = 0.01**:
  Achieved **~98.89%** accuracy on test data.
  Training was **fast and stable**.
  Training Time: 203.15 seconds
- **LR = 0.001**:
  Slower learning, but eventually reached around **97.62%** accuracy.
  Slightly less accurate than 0.01.
- Training Time: 210.40 seconds
- **LR = 0.0001** :
  Likely to be **too slow**, often underperforms unless trained for many more epochs
- Test Accuracy: 89.39%.
- Training Time: 227.52 seconds

Conclusion :

- The **best learning rate** so far is `0.01`, which gave the highest test accuracy.

- You also plotted a graph of learning rate (on log scale) vs. accuracy, which helps visualize



how performance changes.

Step 4: Model Architecture Variation

In this step, we explored different CNN architectures by varying the number of convolutional and fully connected (FC) layers to identify a model that balances high accuracy with a reduced number of parameters. The goal was to **optimize architecture** without significantly sacrificing performance.

4.1 Approach

- We used the **best settings from Step 2 and Step 3**:
    - Epochs: Best value found from epoch tuning
    - Learning Rate: Best value from learning rate testing
- We limited architectures to:
    - Max 3 convolutional layers
    - Max 4 fully connected (hidden) layers, each with ≤ 512 neurons
- All models used ReLU activation in hidden layers and softmax in the output layer (10 classes).
- Optimizer: SGD with the chosen learning rate

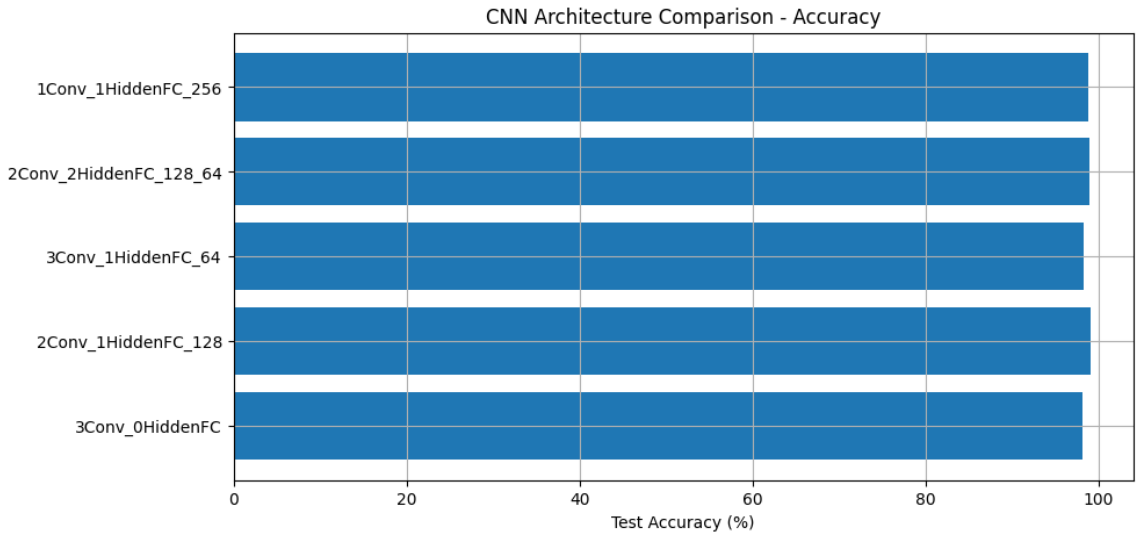
Step 4.2: Architecture Variation Results and Analysis

We experimented with five CNN architectures varying the number of convolutional and fully connected layers. The objective was to balance model accuracy and parameter count to optimize performance and efficiency.

- The highest accuracy (99.03%) was achieved by the **2Conv_1HiddenFC_128** model.
- Although the 1Conv_1HiddenFC_256 model has the largest number of parameters, it does not outperform smaller models in accuracy.
- Models with more convolutional layers but fewer FC neurons achieved reasonable accuracy with fewer parameters.
- Training times were roughly comparable across models, with minor variations.

Visualization:

- The bar chart below compares the test accuracies of each architecture.



- The scatter plot illustrates the trade-off between the number of parameters and accuracy.

Conclusion for Step 4

The **2Conv_1HiddenFC_128** architecture offers the best balance of accuracy and complexity, making it the preferred choice for further experimentation and tuning in Step 5.

Step 5: Batch Size and Activation Function Tuning

Experimental Setup

We tuned the batch size and activation functions to further improve the CNN model performance. The architecture and learning rate were fixed from the best configuration found in previous steps.

- **Batch sizes tested:** 32 (baseline), 128, 192
- **Activation functions tested:** ReLU (baseline), Sigmoid, Tanh, ELU

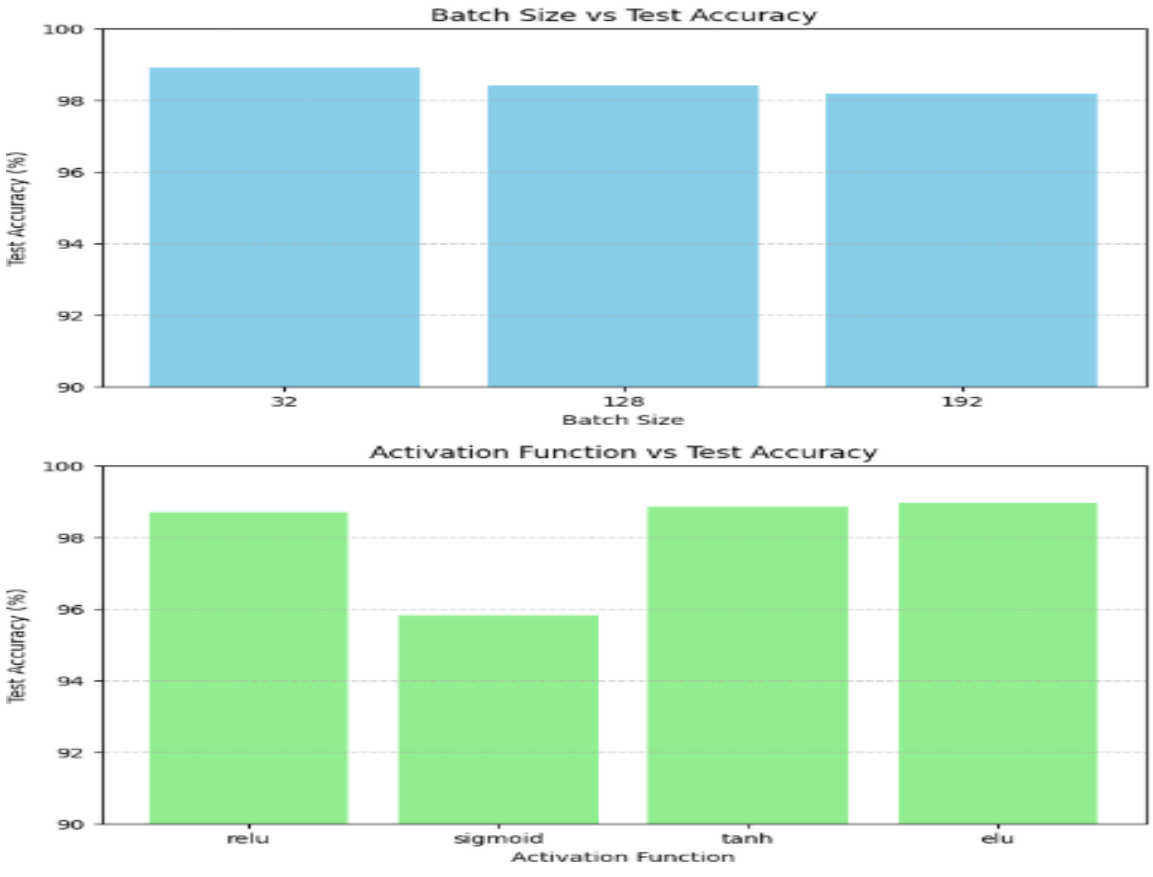
- Training epochs and learning rate remained constant for fair comparison.

Conclusion and Final Settings

- TSmaller batch size (32) gave the highest test accuracy (~98.90%) but took the longest training time (~210 seconds).
- Larger batch sizes (128, 192) reduced training time significantly (to ~66 and ~45 seconds) but slightly lowered accuracy (98.42% and 97.90%).
- Smaller batches may improve model generalization, while larger batches speed up training.
- The eLU activation function consistently enabled good learning performance across all batch sizes.
- Choice of batch size should consider application needs—whether accuracy or training efficiency is more critical.

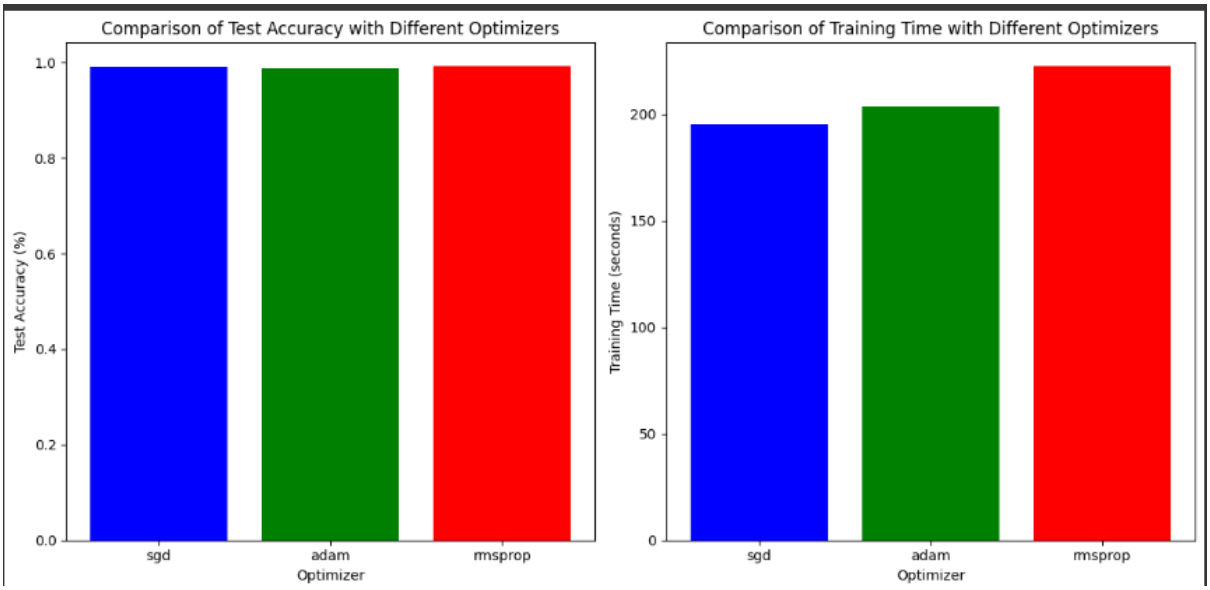
Visualizations

- **Figure 1:** Batch Size vs Test Accuracy
  A bar chart showing test accuracy across batch sizes, highlighting the peak at 32.
- **Figure 2:** Activation Function vs Test Accuracy
  A bar chart comparing test accuracies for each activation function, with ELU



Step 6 Conclusion (Optimizers + Dropout)

- **Optimizers tested:** SGD (with tuned LR), Adam (default LR), RMSProp (default LR).
- **Best optimizer:** RMSProp gave the highest test accuracy (~99.18%) and balanced training time.
- **Dropout layers:** Tested dropout in two positions:
    - After first max-pooling layer
    - After first fully connected (FC) hidden layer (if present)
- **Dropout rates tried:** 0.25 and 0.5.
- **Effect of dropout:**
    - None of the dropout configurations improved test accuracy beyond the baseline (no dropout).
    - Dropout slightly increased training time due to added computations.
    - Dropout did not significantly enhance generalization in this setting, possibly because the model was already well-regularized or the dataset is simple (MNIST).
- **Final choice:**
    - Use **RMSProp** optimizer without dropout layers for best performance and efficiency in this architecture.



Step 7: Final Model & Report Writing

Final Model Configuration

- **Convolutional Layers:** [(32 filters, 3x3 kernel), (64 filters, 3x3 kernel)]
- **Fully Connected Layers:** [128 units]
- **Epochs:** 25
- **Batch Size:** 32
- **Activation Function:** ELU
- **Optimizer:** RMSprop
- **Dropout:** None (dropout was tested but did not improve the model for the chosen optimizer)

Training the Final Model

The final CNN model was built using the above configuration and trained on the MNIST dataset for 25 epochs with a batch size of 32. The model summary and training progress are shown below:

- Total Parameters: 225,034
- Training Time: Approximately 203 seconds

Final Model Performance

- **Test Accuracy:** 99.22%
- **Test Loss:** 0.0839

Final Analysis and Summary of Choices

The final model was selected based on a systematic and incremental hyperparameter tuning process:

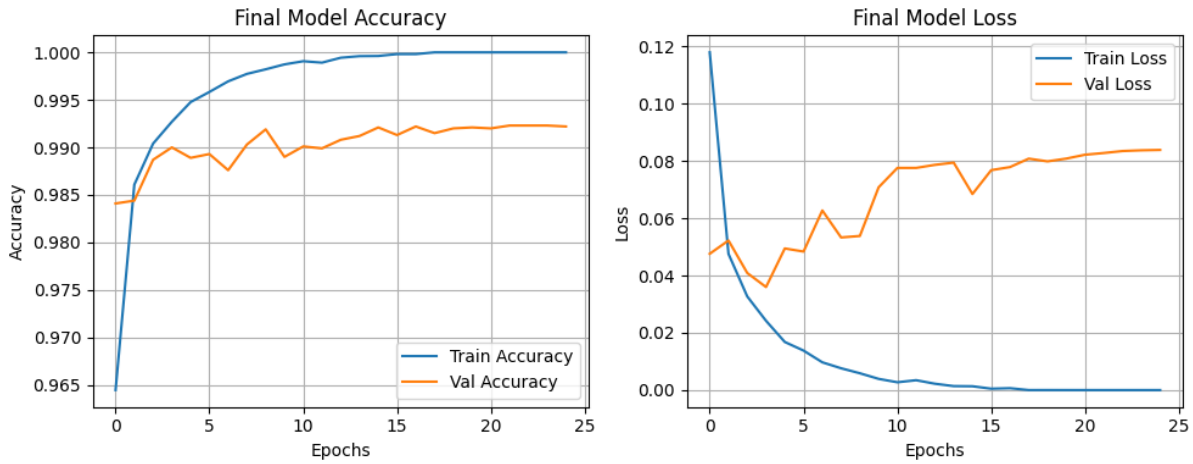
1. **Baseline Models:** Initial benchmarks were set by training basic ANN and SVM models.
2. **Epoch Tuning:** Tested various epochs ([10, 15, 20, 25]) and settled on 25 epochs for best performance.
3. **Learning Rate Testing:** For SGD optimizer, learning rates of [0.01, 0.001, 0.0001] were tested; 0.01 was optimal.
4. **Architecture Exploration:** Different convolutional and fully connected layer combinations were evaluated. The final architecture with two convolutional layers and one fully connected layer (128 units) provided a good balance between accuracy and complexity.
5. **Batch Size & Activation Functions:** Tested batch sizes [32, 128, 192] and activations ['relu', 'sigmoid', 'tanh', 'elu']. Batch size 32 and ELU activation yielded the best results.

6. **Optimizers & Dropout:** Evaluated optimizers ['sgd', 'adam', 'rmsprop']; RMSprop gave the best validation accuracy. Dropout layers were tested but did not improve the model performance with RMSprop.

This methodical approach allowed us to isolate the impact of each hyperparameter and achieve a highly accurate CNN model for MNIST digit classification. Further improvements could be explored through more extensive hyperparameter searches or advanced techniques.

Visualizations

Below are the accuracy and loss plots for training and validation over the 25 epochs:



Conclusion

The final CNN model achieved excellent accuracy on the MNIST test set, demonstrating effective hyperparameter tuning and architecture design. The model balances complexity with performance, and the chosen hyperparameters reflect a well-optimized configuration for this task.