



**project  
AI322-Supervised learning**

|  |  |
| --- | --- |
| **ID** | **Name** |
| **20220131** | Rana Esmail Zekery |
| **20220025** | Ahmed Abdelaziz |
|  | Ammar Mohammed |
|  | Marawan Osama |

# 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.

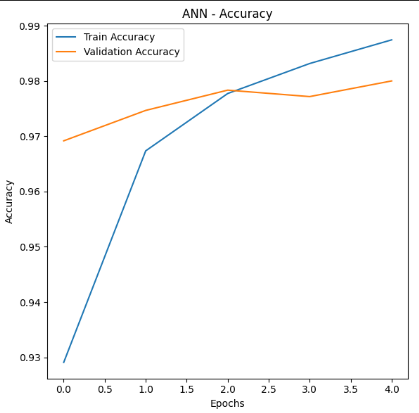
We trained the model using:

* **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

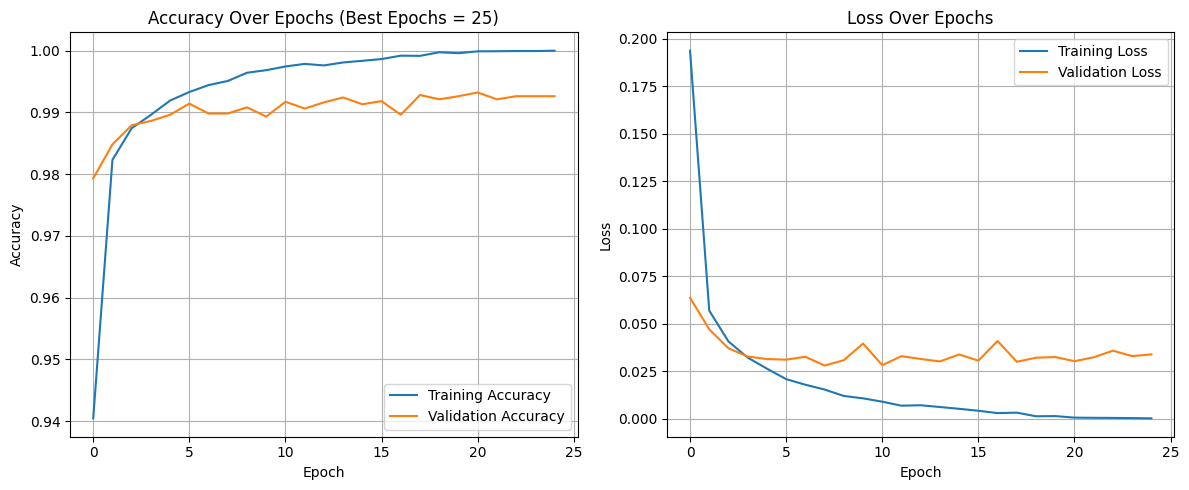
#### ****4. Base CNN (25 Epochs)****

* **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.

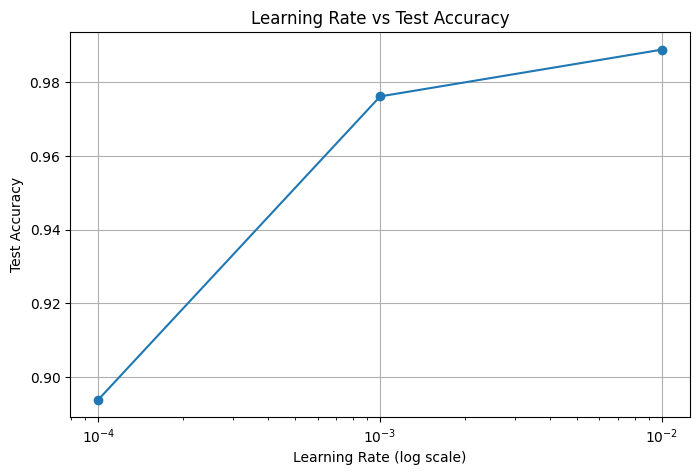
### ****What You Did:****

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.