CS 519 Applied Machine Learning I

HW7: Convolutional Neural Networks

Submitted by: Indronil Bhattacharjee

1 Convolutional Neural Network:

A Convolutional Neural Network (CNN) is a type of artificial neural network designed specifically to process and analyze visual data, such as images and videos. CNNs have gained immense popularity and become the standard approach for various computer vision tasks due to their ability to automatically and adaptively learn spatial hierarchies of features from the input data.

1.1 Key components and how CNNs work:

1.1.1 Convolutional Layers:

Convolutional layers are the building blocks of CNNs. They consist of a set of learnable filters (also known as kernels) that are applied to small, local regions of the input data. Each filter detects specific features, such as edges, textures, or patterns, by performing a convolution operation (element-wise multiplication followed by summation) between the filter weights and the input data. Multiple filters are typically used in parallel to capture different features, resulting in multiple feature maps as output.

1.1.2 Pooling Layers:

Pooling layers are interspersed between convolutional layers to progressively reduce the spatial dimensions of the feature maps while retaining important information. The most common pooling operation is max pooling, which partitions each feature map into non-overlapping regions and outputs the maximum value from each region. This helps in making the representation smaller and more manageable. Pooling layers also introduce a degree of translation invariance, making the network more robust to small shifts in the input data.

1.1.3 Activation Functions:

Non-linear activation functions like ReLU (Rectified Linear Unit) are applied element-wise after each convolutional and pooling operation to introduce non-linearity into the network. ReLU is commonly used due to its simplicity and effectiveness in addressing the vanishing gradient problem.

1.1.4 Fully Connected Layers:

Fully connected layers are typically placed towards the end of the CNN architecture. They consist of neurons that are fully connected to neurons in the preceding layer, similar to traditional neural networks. Fully connected layers perform high-level reasoning and decision-making based on the features extracted by the convolutional and pooling layers.

1.1.5 Flattening:

Before passing the output of the convolutional and pooling layers to the fully connected layers, the feature maps are flattened into a 1-dimensional vector. This flattening step is necessary because fully connected layers require one-dimensional inputs.
Overall, CNNs leverage the hierarchical pattern recognition capabilities of convolutional layers, the spatial hierarchy reduction of pooling layers, and the high-level reasoning of fully connected layers to automatically learn and extract features from input images, enabling tasks such as image classification, object detection, and image segmentation.

2 CNN used in the experiment:

2.1 CNN Layers Description:

The convolutional neural network (CNN) architecture implemented in the experiment is designed for image classification tasks. It consists of several layers, each with a specific role in processing and extracting features from input images.

2.1.1 Convolutional Layer 1 (Conv1):

The first convolutional layer (Conv1) takes a grayscale image with dimensions 28x28 as input. It applies four different filters with a kernel size of 3x3 to convolve across the input image. Each filter extracts local patterns and features, such as edges or textures, from the input image.

2.1.2 Max-Pooling Layer 1 (Pool1):

Following Conv1, a max-pooling layer (Pool1) with a kernel size of 2x2 and a stride of 2x2 is applied. This layer reduces the spatial dimensions of the feature maps produced by Conv1 by half, effectively downsampling the features while retaining important information.

2.1.3 Convolutional Layer 2 (Conv2):

The output of Pool1 serves as the input to the second convolutional layer (Conv2). Conv2 applies two filters with a kernel size of 3x3 to extract higher-level features from the feature maps generated by Conv1. The stride of 3x3 in this layer results in larger strides between filter applications, further reducing the size of the feature maps.

2.1.4 Max-Pooling Layer 2 (Pool2):

After Conv2, another max-pooling layer (Pool2) is applied with a kernel size of 4x4 and a stride of 4x4. This layer further downsamples the feature maps, reducing their spatial dimensions while preserving important features.

2.1.5 Flattening:

The output of Pool2, which consists of two feature maps, is flattened into a one-dimensional vector. This flattened feature vector serves as the input to the fully connected layer.

2.1.6 Fully Connected Layer (FC1):

The flattened feature vector is then passed through a fully connected layer (FC1), which consists of 2 input neurons corresponding to the flattened feature vector and 10 output neurons representing the classes of

the dataset. The fully connected layer performs high-level reasoning and decision-making based on the extracted features.

2.1.7 Output Layer:

Finally, the output of FC1 is passed through a log softmax function, producing a probability distribution over the classes. Each output neuron corresponds to a class, and the class with the highest probability is selected as the predicted class for the input image.

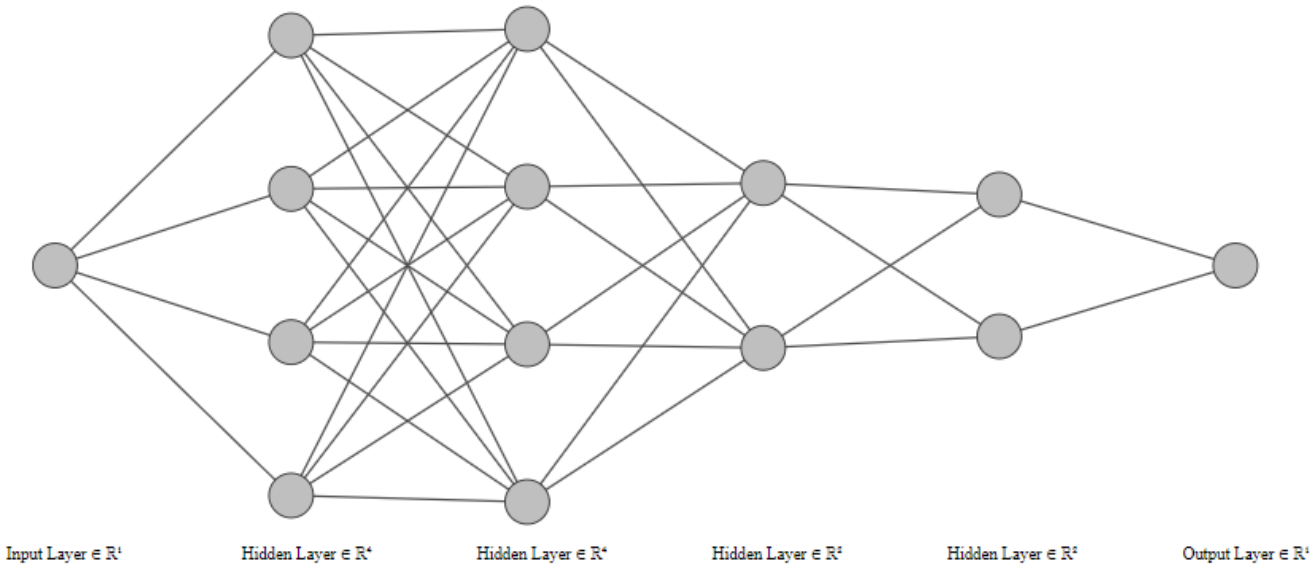


Figure 1: Fully Connected Neural Network representation style layer by layer

2.2 CNN Architecture:

2.2.1 First Convolutional Layer (conv1):

- Input:

- **Depth (number of input channels):** 1 (since it's a grayscale image)
- **Height:** 28
- **Width:** 28
- **Dimension:** batch_size x 1 x 28 x 28

- Kernel size: 3×3

- Stride: 1×1

- Padding: 0 (valid padding)

- Output:

- **Depth (number of output channels):** 4
- **Height:** 28 – 2 = 26
- **Width:** 28 – 2 = 26
- **Dimension:** batch_size x 4 x 26 x 26

2.2.2 First Max Pooling Layer (pool1):

- Input:

- **Depth (number of input channels):** 4
- **Height:** 26
- **Width:** 26
- **Dimension:** batch_size x 4 x 26 x 26

- Kernel size: 2×2

- Stride: 2×2

- Output:

- **Depth (number of output channels):** 4
- **Height:** $\left\lfloor \frac{26}{2} \right\rfloor = 13$
- **Width:** $\left\lfloor \frac{26}{2} \right\rfloor = 13$
- **Dimension:** batch_size x 4 x 13 x 13

2.2.3 Second Convolutional Layer (conv2):

- Input:

- **Depth (number of input channels):** 4
- **Height:** 13
- **Width:** 13
- **Dimension:** batch_size x 4 x 13 x 13

- Kernel size: 3×3

- Stride: 3×3

- Padding: 0 (valid padding)

- Output:

- **Depth (number of output channels):** 2
- **Height:** $\left\lfloor \frac{(13-2)}{3} \right\rfloor = 4$
- **Width:** $\left\lfloor \frac{(13-2)}{3} \right\rfloor = 4$
- **Dimension:** batch_size x 2 x 4 x 4

2.2.4 Second Max Pooling Layer (pool2):

- Input:

- **Depth (number of input channels):** 2
- **Height:** 4
- **Width:** 4
- **Dimension:** batch_size x 2 x 4 x 4

- Kernel size: 4×4

- Stride: 4×4

- Output:

- **Depth (number of output channels):** 2

- **Height:** $\left\lfloor \frac{4}{4} \right\rfloor = 1$
- **Width:** $\left\lfloor \frac{4}{4} \right\rfloor = 1$
- **Dimension:** batch_size x 2 x 1 x 1

2.2.5 Fully Connected Layer (fc1):

Input:

- **Depth (number of input channels):** 2
- **Height:** 1
- **Width:** 1
- **Dimension:** batch_size x 2 x 1 x 1

Output:

- **Depth (number of output channels):** 10
- **Dimension:** batch_size x 10

```
----------------------------------------------------------------
    Layer (type)               Output Shape         Param #
================================================================
        Conv2d-1          [-1, 4, 26, 26]              40
     MaxPool2d-2          [-1, 4, 13, 13]               0
        Conv2d-3           [-1, 2, 4, 4]              74
     MaxPool2d-4           [-1, 2, 1, 1]               0
        Linear-5                [-1, 10]              30
================================================================
```

Figure 2: CNN Architecture Output Dimension Snapshot

2.3 Tabular representation of the CNN Architecture:

Layer	Input	Output	Kernel	Stride
Conv1	batch_size x 1 x 28 x 28	batch_size x 4 x 26 x 26	3 x 3	1 x 1
Pool1	batch_size x 4 x 26 x 26	batch_size x 4 x 13 x 13	2 x 2	2 x 2
Conv2	batch_size x 4 x 13 x 13	batch_size x 2 x 4 x 4	3 x 3	3 x 3
Pool2	batch_size x 2 x 4 x 4	batch_size x 2 x 1 x 1	4 x 4	4 x 4
Fc1	batch_size x 2 x 1 x 1	batch_size x 10	-	-

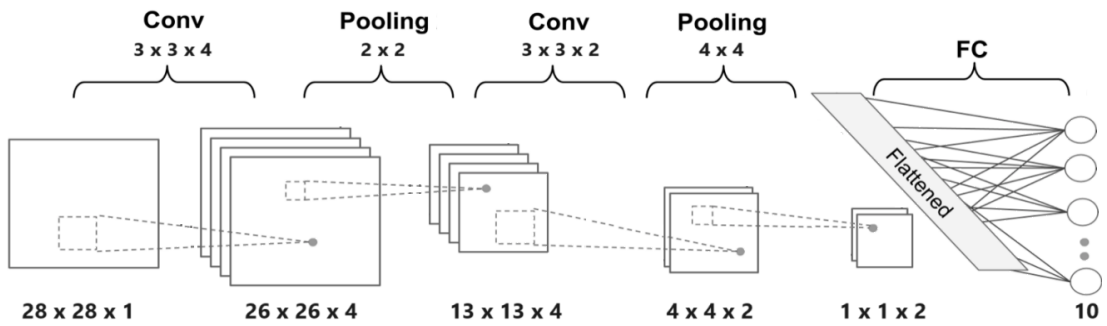


Figure 3: CNN Architecture

3 MNIST Dataset:

The MNIST dataset is a widely used benchmark dataset in the field of machine learning and computer vision. It consists of a large collection of handwritten digits, ranging from 0 to 9, each digit being represented as a grayscale image of size 28x28 pixels. The dataset is split into two main subsets: a training set and a test set.

The training set contains 60,000 examples, while the test set consists of 10,000 examples. Each example in the dataset consists of an image of a handwritten digit and the corresponding label indicating the true digit represented in the image.

The MNIST dataset is commonly used for training and evaluating machine learning models, particularly in tasks such as digit recognition and image classification. Due to its simplicity and accessibility, it has become a standard benchmark for evaluating the performance of various machine learning algorithms, especially those related to deep learning and convolutional neural networks (CNNs).

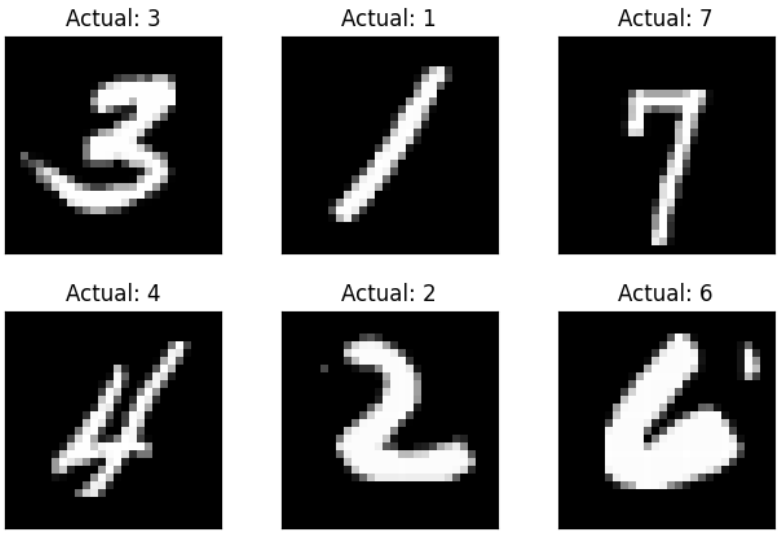


Figure 4: Some instances of MNIST dataset with actual labels

4 Testing the CNN using MNIST dataset:

To test the CNN with the MNIST dataset, we first train the network using the training set of MNIST images and corresponding labels. During training, the network learns to classify the images into their respective digit classes. After training, we evaluate the performance of the trained network using the test set of MNIST images and labels.

During testing, we pass each image through the trained network and compare the predicted class label with the true label. We compute the accuracy of the network by dividing the number of correctly classified images by the total number of images in the test set.

The MNIST dataset consists of 28x28 grayscale images of handwritten digits from 0 to 9. It contains a training set of 60,000 images and a test set of 10,000 images. This dataset is widely used for benchmarking image classification algorithms.

After testing the CNN with the MNIST dataset, we can analyze the accuracy of the network and visualize any misclassifications or patterns in the predictions. This evaluation helps us assess the performance of the CNN and identify areas for improvement if necessary.
These settings were chosen to ensure a thorough exploration of the model's performance while maintaining consistency and reproducibility across experiments. The choice of 50 epochs allows for sufficient training time, enabling the model to converge to an optimal solution. A training batch size of 64 was selected to balance computational efficiency and model convergence, while a larger testing batch size of 1000 ensures robust evaluation metrics.

The learning rate of 0.01 with momentum of 0.5 was used for optimizing the model parameters during training, and a random seed of 1 was set to ensure reproducibility of results across different runs. These experimental settings provide a comprehensive evaluation of the CNN model's performance on the MNIST dataset.

4.1 Accuracy analysis

Epoch	Train Accuracy	Test Accuracy		Epoch	Train Accuracy	Test Accuracy
1	11.88%	14.08%		26	38.78%	39.59%
2	18.90%	20.17%		27	38.94%	39.54%
3	20.27%	20.84%		28	39.23%	39.96%
4	21.38%	21.55%		29	39.37%	39.52%
5	22.25%	22.83%		30	39.53%	39.11%
6	22.80%	23.37%		31	39.47%	39.79%
7	23.40%	23.64%		32	39.50%	39.45%
8	23.77%	24.31%		33	39.55%	39.20%
9	24.30%	24.82%		34	39.74%	38.71%
10	24.66%	25.16%		35	39.82%	40.59%
11	25.28%	25.87%		36	39.90%	39.58%
12	25.53%	25.39%		37	40.08%	40.07%
13	25.54%	26.22%		38	39.98%	40.67%
14	25.91%	25.37%		39	40.19%	39.54%
15	29.17%	33.12%		40	39.91%	40.54%
16	34.31%	35.16%		41	40.18%	40.69%
17	34.96%	35.15%		42	40.60%	40.80%
18	35.33%	35.09%		43	41.56%	42.33%
19	35.62%	36.06%		44	42.06%	42.24%
20	36.05%	36.01%		45	42.44%	43.22%
21	36.43%	36.45%		46	42.81%	43.05%
22	37.08%	37.24%		47	43.30%	43.48%
23	37.55%	37.91%		48	43.45%	43.84%
24	38.01%	38.03%		49	43.64%	44.17%
25	38.44%	38.82%		50	43.69%	44.51%

The table above displays the training and testing accuracies achieved by the Convolutional Neural Network (CNN) model over 50 epochs. It is evident that both the training and testing accuracies increase gradually as the number of epochs progresses.

Initially, the model starts with low accuracies, indicating poor performance. However, as training continues, the accuracies improve steadily, reaching higher values towards the later epochs. This trend suggests that the model is learning from the training data and generalizing well to unseen data, as evidenced by the increasing testing accuracies.

The final testing accuracy achieved by the model after 50 epochs is 44.51%, which indicates that the model correctly classified approximately 44.51% of the images in the MNIST test dataset. Although this accuracy may seem modest, it demonstrates the capability of the CNN model to learn and make meaningful predictions on handwritten digit images.

Further analysis could involve fine-tuning hyperparameters, adjusting the model architecture, or employing more advanced techniques to enhance the model's performance and achieve higher accuracies.

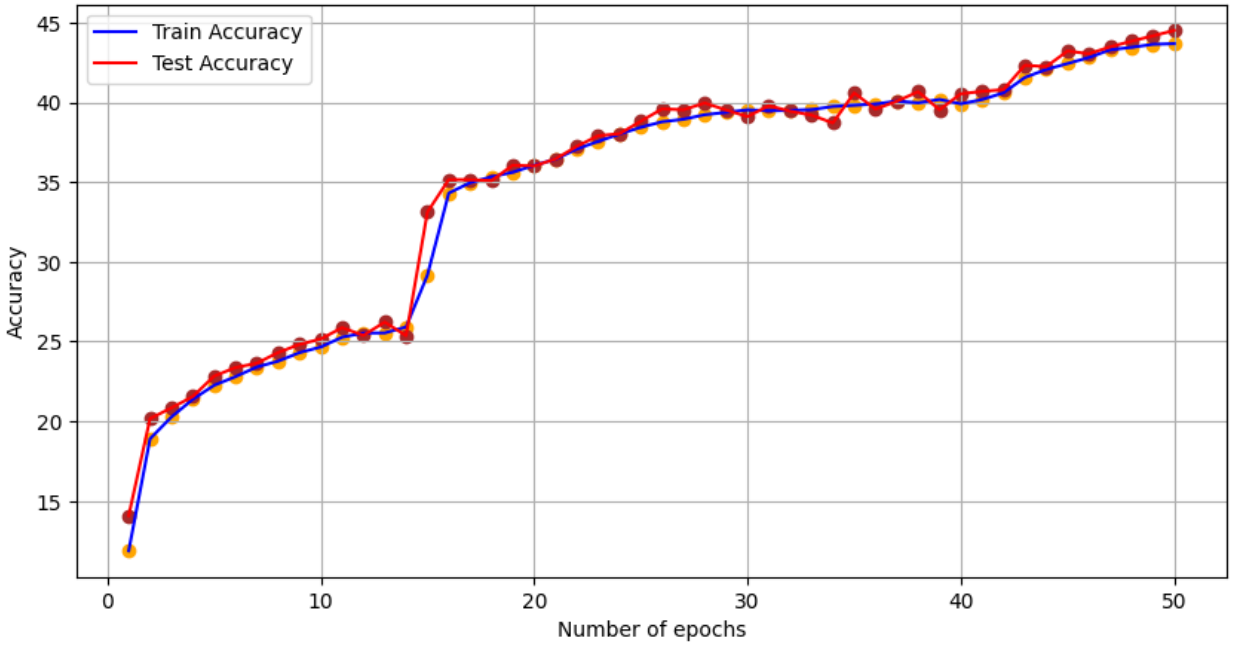


Figure 5: Accuracy vs Number of Epochs

4.2 Loss Analysis:

Epoch	Train Loss	Test Loss		Epoch	Train Loss	Test Loss
1	2.300381	2.2766		26	1.859358	1.8478
2	2.240192	2.1969		27	1.856626	1.8401
3	2.185945	2.1651		28	1.852676	1.8375
4	2.166061	2.1513		29	1.850037	1.8408
5	2.151178	2.1369		30	1.846593	1.8408

Epoch	Train Loss	Test Loss		Epoch	Train Loss	Test Loss
6	2.138571	2.1249		31	1.843248	1.8289
7	2.129344	2.1200		32	1.838186	1.8213
8	2.121814	2.1105		33	1.826120	1.8161
9	2.112158	2.1033		34	1.818935	1.8250
10	2.101153	2.1010		35	1.812604	1.7948
11	2.093191	2.0860		36	1.809341	1.8111
12	2.087049	2.0791		37	1.806346	1.7905
13	2.081887	2.0742		38	1.805053	1.7887
14	2.076647	2.0704		39	1.802753	1.7921
15	2.037060	1.9767		40	1.800386	1.7824
16	1.959070	1.9371		41	1.798579	1.7810
17	1.933703	1.9125		42	1.788094	1.7687
18	1.920665	1.9065		43	1.760673	1.7331
19	1.911357	1.8926		44	1.746487	1.7220
20	1.901810	1.8944		45	1.727890	1.7040
21	1.893336	1.8772		46	1.718973	1.7007
22	1.883923	1.8728		47	1.711202	1.6942
23	1.875210	1.8581		48	1.705452	1.6873
24	1.869960	1.8549		49	1.701452	1.6857
25	1.864386	1.8530		50	1.699412	1.6794

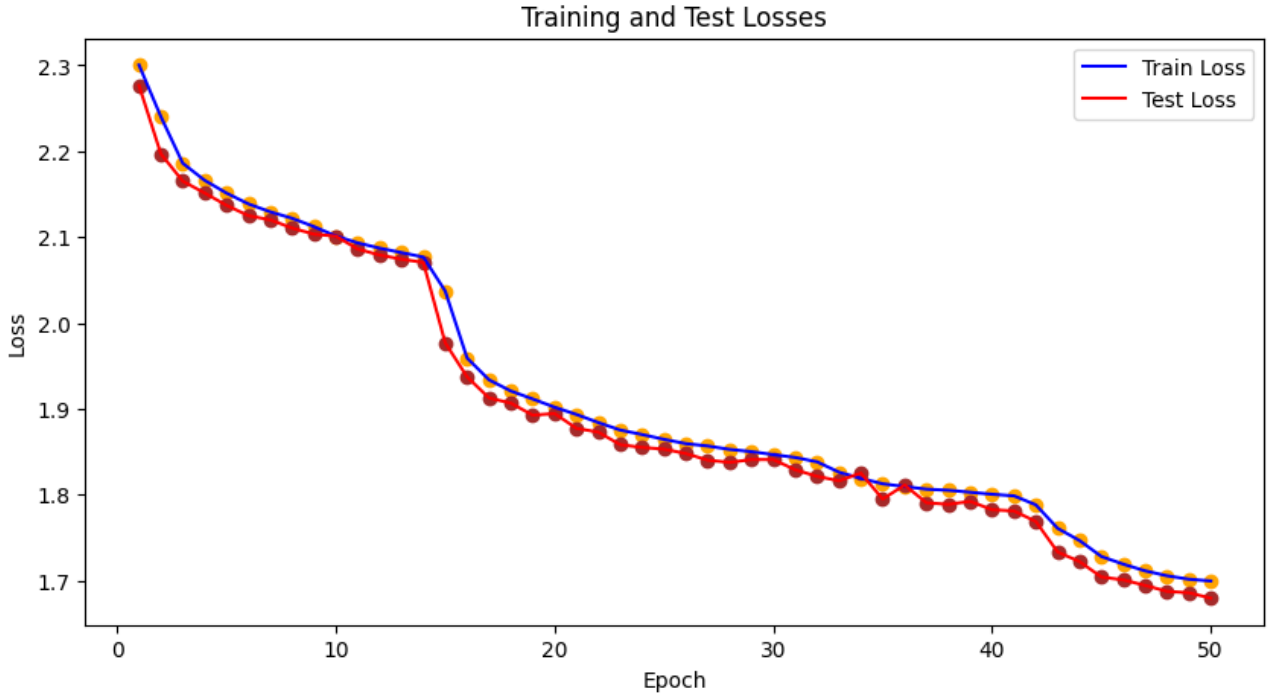


Figure 6: Loss vs Number of Epochs

The tables above show the training and testing loss values for the CNN model over each set of 10 epochs. The training loss decreases over time, indicating that the model is improving its fit to the training data. Similarly, the testing loss generally decreases, suggesting that the model is also performing better on unseen data as training progresses.

The decrease in both training and testing loss values demonstrates that the model is learning and becoming more effective at making predictions. However, it's important to monitor for signs of overfitting, where the training loss continues to decrease while the testing loss starts to increase or plateau. In this case, the narrowing gap between the training and testing loss values indicates that the model is not overfitting excessively and is achieving good generalization performance.

4.3 Number of predictions:

The number of correct predictions for the best accuracy achieved during testing is after epoch 50. For the testing dataset, with an accuracy of 44.51%, the number of correct predictions is near 4500 among 10000 instances of the total dataset, which is not much promising.

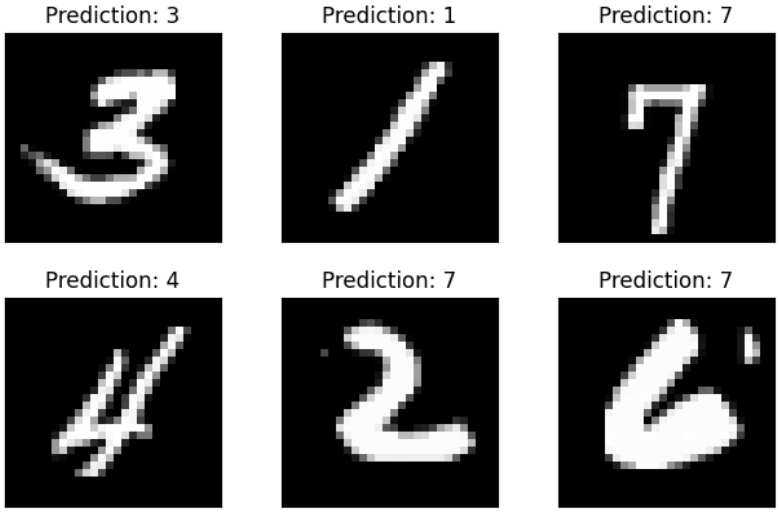


Figure 7: Some instances of MNIST dataset with predicted labels

5 Performance analysis of the CNN with parameter tuning using MNIST dataset:

5.1 Kernel Size

In the previous model, The first convolutional layer (conv1) had a kernel size of 3x3, which has been changed to 5x5 in the updated model. Now after updating the model with different kernel size, the updated model has both conv1 and conv2 with a kernel size of 5x5.

5.1.1 Loss Analysis with kernel 5x5:

Epoch	Train Accuracy	Test Accuracy		Epoch	Train Accuracy	Test Accuracy
1	71.89	9.31		26	83.82	84.89
2	80.62	80.27		27	83.82	83.25
3	80.88	81.99		28	83.98	82.46

Epoch	Train Accuracy	Test Accuracy		Epoch	Train Accuracy	Test Accuracy
4	81.02	80.73		29	83.93	82.11
5	81.18	79.68		30	83.81	82.33
6	81.24	81.68		31	83.80	83.17
7	81.31	80.93		32	83.98	83.23
8	81.48	80.61		33	83.86	82.86
9	81.46	80.87		34	83.82	84.38
10	81.36	81.69		35	83.77	84.29
11	81.60	82.40		36	84.00	84.50
12	81.73	81.61		37	83.92	83.74
13	81.65	82.38		38	83.93	83.56
14	81.74	82.83		39	83.94	84.48
15	81.83	83.25		40	83.89	81.67
16	82.17	81.61		41	83.93	83.52
17	82.48	82.74		42	83.94	81.24
18	83.08	83.32		43	83.95	84.22
19	83.13	82.68		44	83.98	81.99
20	83.33	83.53		45	83.96	84.05
21	83.45	83.78		46	83.89	84.95
22	83.50	83.87		47	83.93	84.77
23	83.48	82.44		48	84.00	82.62
24	83.68	83.32		49	83.86	84.21
25	83.56	84.19		50	84.03	84.75

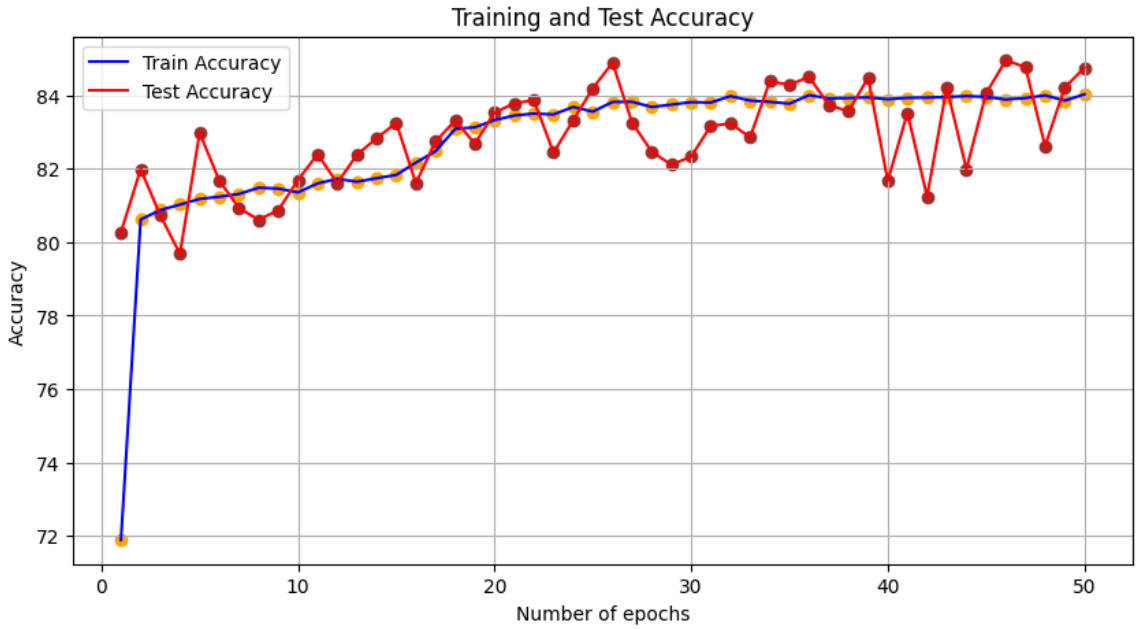


Figure 8: Accuracy vs Number of Epochs

5.1.2 Loss Analysis with kernel 5x5:

Epoch	Train Loss	Test Loss		Epoch	Train Loss	Test Loss
1	0.883	2.3228		26	0.488	0.4701
2	0.607	0.6201		27	0.486	0.4535
3	0.597	0.5424		28	0.490	0.5488
4	0.590	0.5906		29	0.489	0.5145
5	0.582	0.6287		30	0.487	0.5440
6	0.578	0.5199		31	0.487	0.5320
7	0.578	0.5897		32	0.483	0.5088
8	0.570	0.5801		33	0.484	0.5050
9	0.567	0.6840		34	0.486	0.5451
10	0.569	0.5776		35	0.484	0.4654
11	0.565	0.5782		36	0.483	0.4703
12	0.561	0.5342		37	0.483	0.4559
13	0.560	0.5880		38	0.484	0.4844
14	0.559	0.5337		39	0.481	0.4894
15	0.551	0.5188		40	0.482	0.4521
16	0.540	0.5122		41	0.481	0.5418
17	0.526	0.5590		42	0.483	0.4797
18	0.512	0.5043		43	0.482	0.5735
19	0.507	0.4950		44	0.483	0.4692
20	0.502	0.5187		45	0.485	0.5545
21	0.502	0.4858		46	0.483	0.4816
22	0.496	0.4779		47	0.481	0.4482
23	0.499	0.4716		48	0.482	0.4474
24	0.491	0.5402		49	0.483	0.5364
25	0.492	0.4949		50	0.480	0.4478

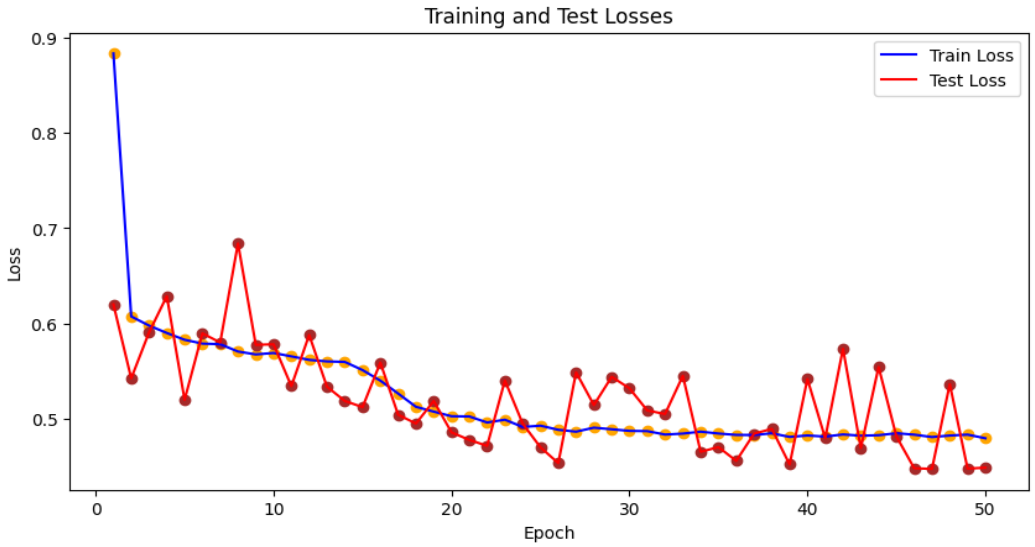


Figure 9: Loss vs Number of Epochs

5.1.3 Number of prediction:

The number of correct predictions for the best accuracy achieved during testing is after epoch 50. For the testing dataset, with an accuracy of 84.75%, the number of correct predictions is near 8500 among 10000 instances of the total dataset, which is better than the previous one.

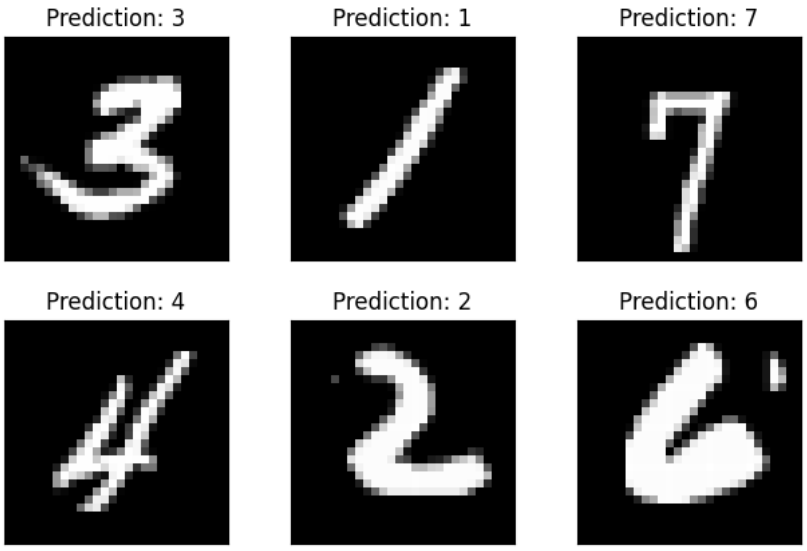


Figure 10: Some instances of MNIST dataset with actual labels

5.1.4 Comparative analysis:

Accuracy:

- The first model achieved a maximum test accuracy of 44.51% after 50 epochs.
- The second model achieved a higher maximum test accuracy of 84.75% after 50 epochs.

The second model demonstrates a significant improvement in accuracy compared to the first model.

Loss:

- The first model's test loss fluctuates throughout training, starting at 2.3228 and decreasing to 1.6794.
- The second model's test loss shows a consistent decreasing trend, starting at 2.3228 and decreasing to 0.4478.

The second model exhibits much lower test loss values compared to the first model, indicating better convergence and model fit.

Architecture:

- The first model utilizes a simpler architecture with fewer layers and parameters.
- The second model appears to have a more complex architecture with additional convolutional and pooling layers, potentially allowing it to learn more intricate patterns in the data.

Training Dynamics:

- The first model's training accuracy increases gradually over epochs, while the test accuracy fluctuates.
- The second model demonstrates a more consistent improvement in both training and test accuracy over epochs, indicating better generalization.

Training Time:

- The training time for the second model might be longer due to its more complex architecture and potentially larger number of parameters.

Overall, the second model outperforms the first model in terms of both accuracy and loss, suggesting that the architectural enhancements and training dynamics improvements have led to better model performance. However, it's essential to consider factors such as computational complexity and training time when choosing between these models.

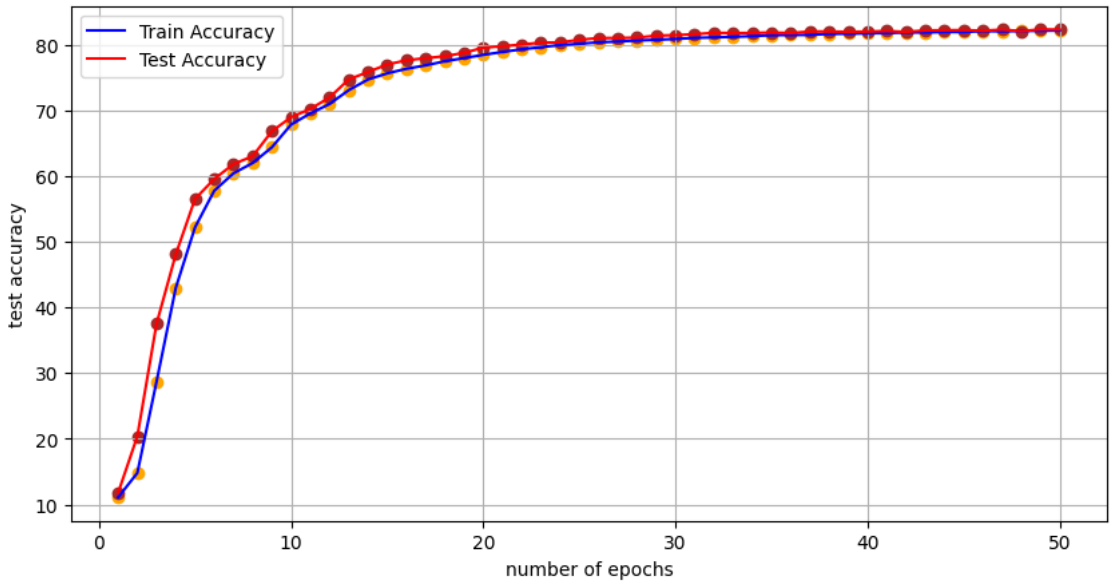
Metric	Previous Model	New Model
Conv Kernels	3 x 3	5 x 5
Train Accuracy	43.69%	84.03%
Test Accuracy	44.51%	84.75%
Train Loss	1.6994	0.4796
Test Loss	2.3228	0.4487
Time (mins)	18.16 min	18.41 min

5.2 Learning Rate

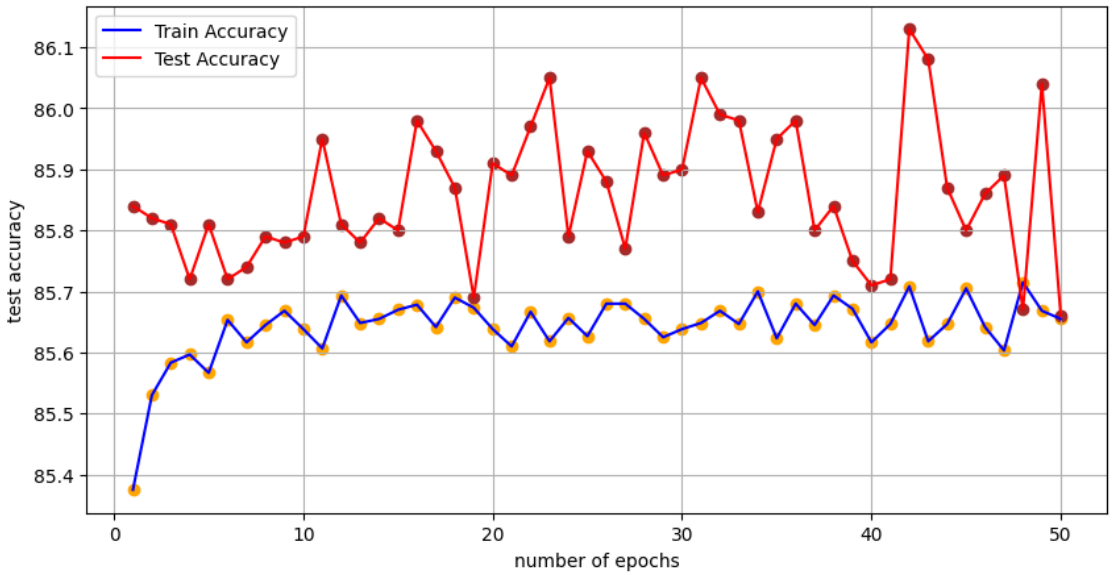
In the previous model, The first model had a learning rate of 0.001, which has been changed to 0.01 in the updated model. Now after updating the model with different learning rates, the updated model has the learning rate of 0.01.

5.2.1 Accuracy Analysis with different learning rates:

Learning rate = 0.001				Learning rate = 0.01		
Epoch Interval	Train Accuracy	Test Accuracy		Epoch Interval	Train Accuracy	Test Accuracy
1-10	67.85%	68.97%		1-10	85.38%	85.84%
11-20	69.55%	70.23%		11-20	85.69%	85.95%
21-30	78.91%	79.78%		21-30	85.64%	86.05%
31-40	81.04%	82.09%		31-40	85.69%	86.04%
41-50	82.17%	82.34%		41-50	85.71%	86.13%



(a)

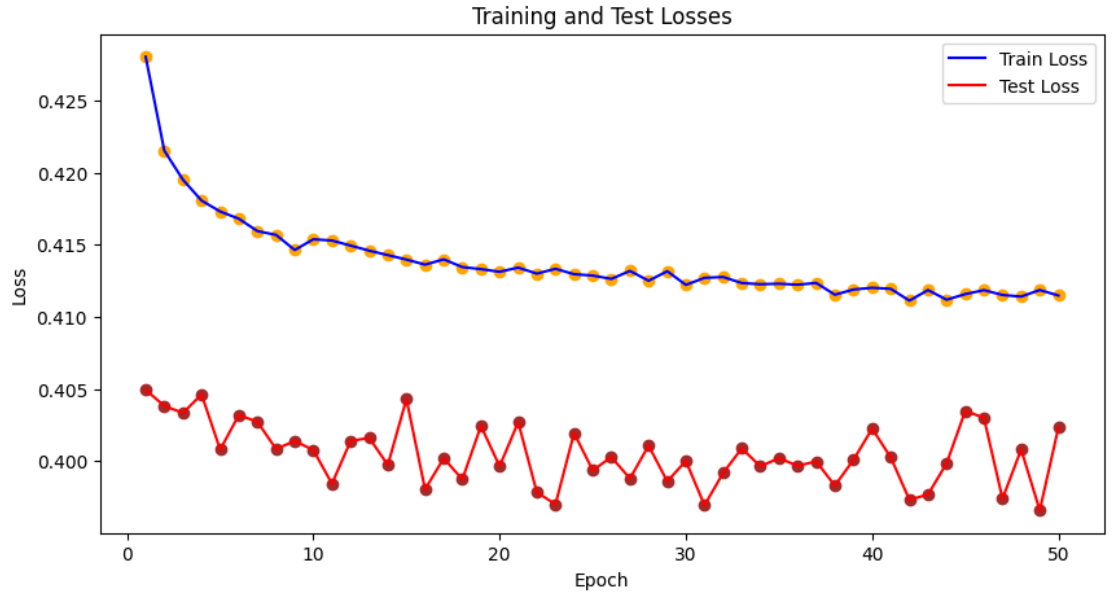


(b)

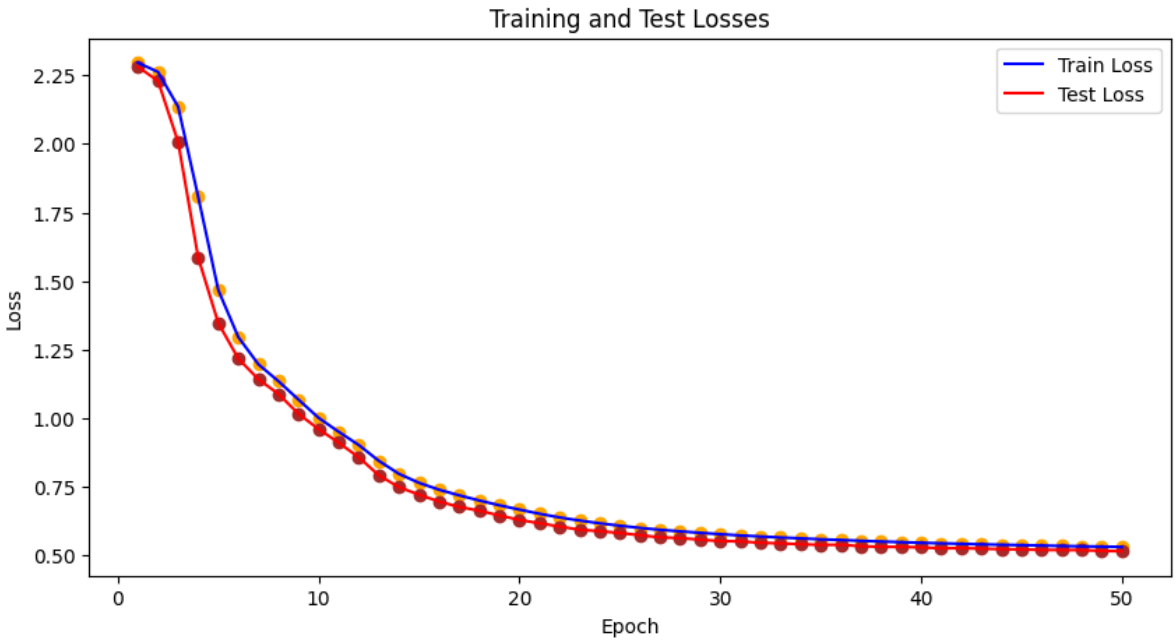
Figure 11: Accuracy vs Number of Epochs with learning rate = (a) 0.001 and (b) 0.01

5.2.2 Loss Analysis with different learning rates:

Epoch Interval	Train Loss	Test Loss		Epoch Interval	Train Loss	Test Loss
1-10	1.000840	0.9599		1-10	0.428037	0.4049
11-20	0.651502	0.6168		11-20	0.415290	0.3984
21-30	0.577129	0.5518		21-30	0.412213	0.3969
31-40	0.568759	0.5255		31-40	0.411529	0.3966
41-50	0.530636	0.5150		41-50	0.411119	0.3973



(a)

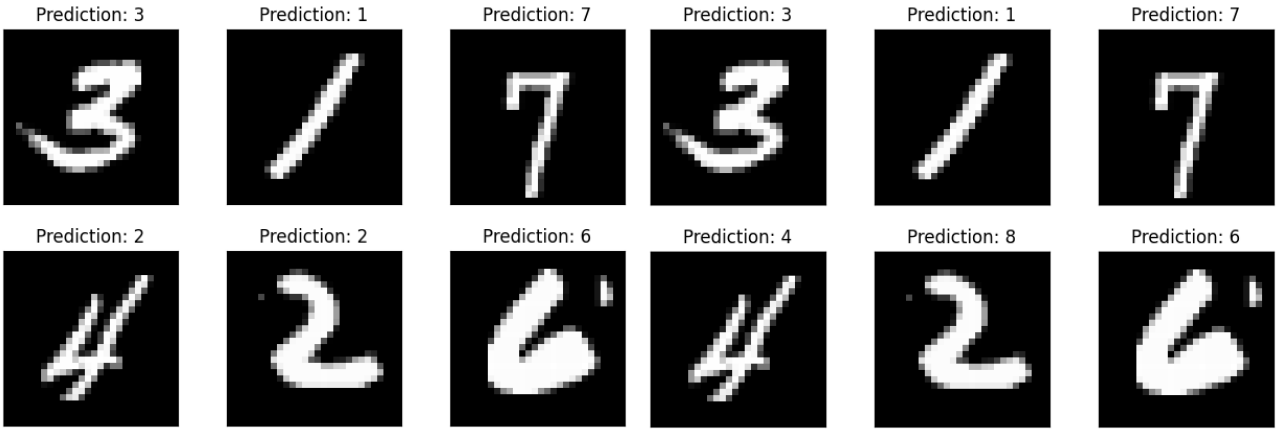


(b)

Figure 12: Loss vs Number of Epochs with learning rate = (a) 0.001 and (b) 0.01

5.2.3 Number of predictions for different learning rates:

The number of correct predictions for the best accuracy achieved during testing is after epoch 50. For the testing dataset, with an accuracy of 82.34% and 86.13% for learning rates 0.001 and 0.01 respectively, the number of correct predictions is near 8200 and 8600 among 10000 instances of the total dataset, which is better than the previous one.



(a) **(b)**

Figure 13: Some instances of with predicted labels with learning rate = (a) 0.001 and (b) 0.01

5.2.4 Comparative analysis:

Accuracy:

- The first model achieved a maximum test accuracy of 82.34% after 50 epochs.
- The second model achieved a higher maximum test accuracy of 86.14% after 50 epochs.

The second model with learning rate 0.01 demonstrates a significant improvement in accuracy compared to the first model.

Loss:

- The first model's test loss fluctuates throughout training, starting at 0.9599 and decreasing to 0.5150.
- The second model's test loss shows a consistent decreasing trend, starting at 0.4049 and decreasing to 0.3973.

The second model with 0.001 learning rate exhibits much lower test loss values compared to the first model, indicating better convergence and model fit.

Training Dynamics:

- The first model's training accuracy increases gradually over epochs, while the test accuracy fluctuates.
- The second model demonstrates a more consistent improvement in both training and test accuracy over epochs, indicating better generalization.

Training Time:

- The training time for the second model might be longer due to its more slow learning rate and potentially larger number of parameters.

Metric	Previous Model	New Model
Learning rate	0.001	0.01
Train Accuracy	82.17%	85.71%
Test Accuracy	82.34%	86.13%
Train Loss	0.5306	0.4111
Test Loss	0.5150	0.3973
Time (mins)	32.14 min	21.24 min

5.3 Adding Hidden layers

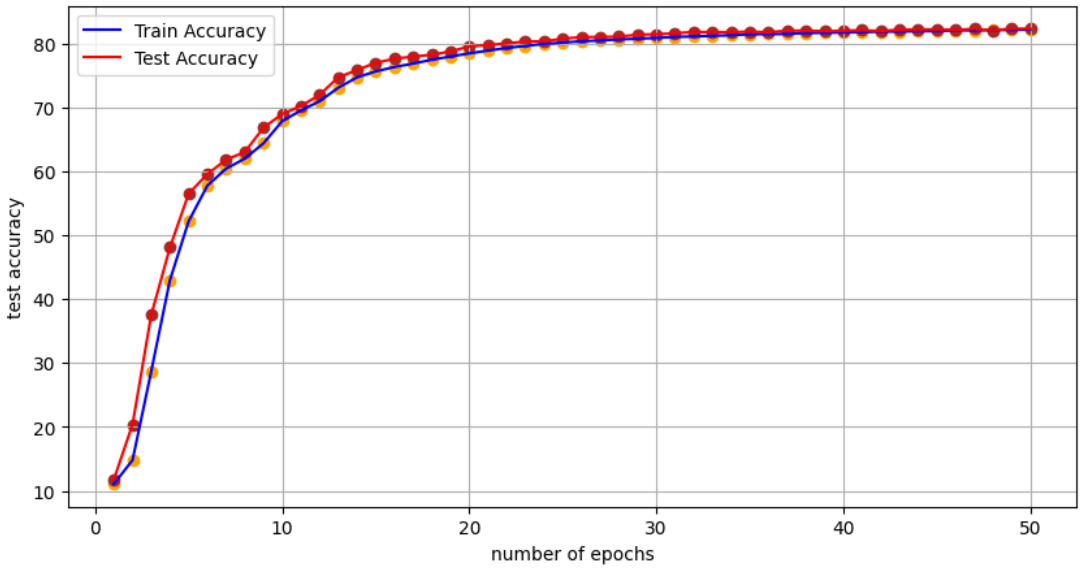
In the previous model, there was no dropout layers. Now after updating the model with addind dropout layers with a more fully connected layer, the updated model 2 more layers.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 10, 24, 24]             260
            Conv2d-2           [-1, 20, 8, 8]           5,020
         Dropout2d-3           [-1, 20, 8, 8]               0
            Linear-4                 [-1, 50]          16,050
            Linear-5                 [-1, 10]             510
================================================================
Total params: 21,840
Trainable params: 21,840
Non-trainable params: 0
```

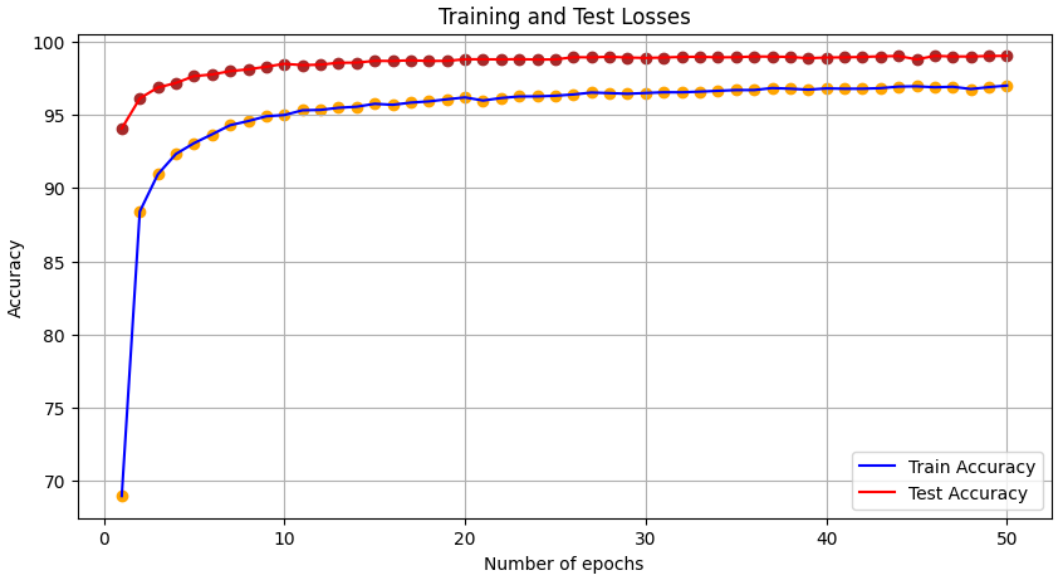
Figure 14: CNN Architecture Output Dimension Snapshot

5.3.1 Accuracy Analysis with added hidden layers:

No dropout layers				Added dropout and FC layer		
Epoch Interval	Train Accuracy	Test Accuracy		Epoch Interval	Train Accuracy	Test Accuracy
1-10	85.38%	85.84%		1-10	94.99%	98.49%
11-20	85.69%	85.95%		11-20	96.19%	98.80%
21-30	85.64%	86.05%		21-30	96.50%	98.89%
31-40	85.69%	86.04%		31-40	96.81%	98.92%
41-50	85.71%	86.13%		41-50	97.00%	99.03%



(a)

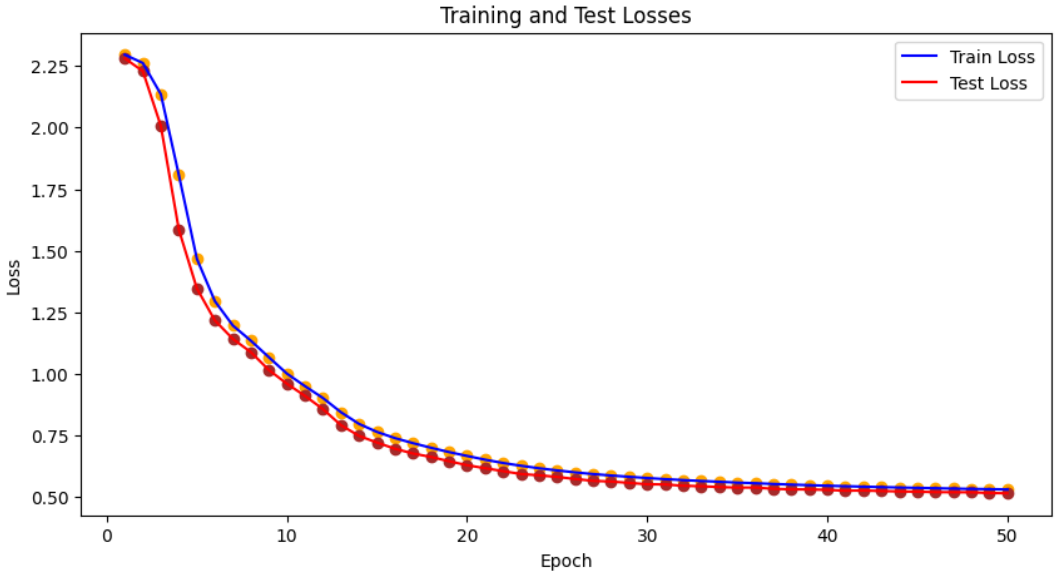


(b)

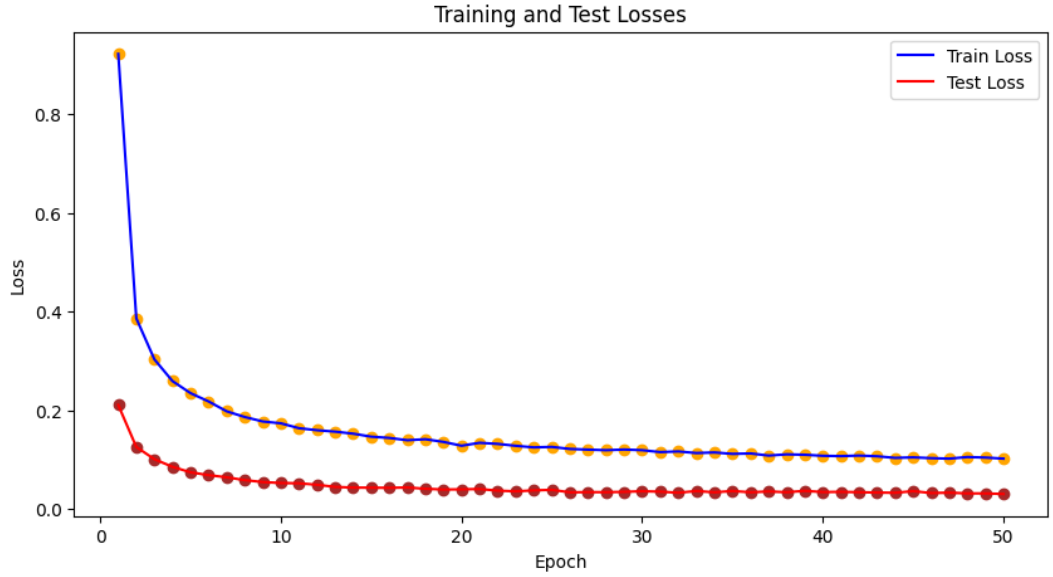
Figure 15: Accuracy vs Number of Epochs with adding Dropout layer = (a) False and (b) True

5.3.2 Loss Analysis with added hidden layers:

No dropout layers				Added dropout and FC layer		
Epoch Interval	Train Loss	Test Loss		Epoch Interval	Train Loss	Test Loss
1-10	0.4280	0.4049		1-10	0.1738	0.0529
11-20	0.4153	0.3984		11-20	0.1284	0.0401
21-30	0.4122	0.3969		21-30	0.1194	0.0363
31-40	0.4115	0.3966		31-40	0.1080	0.0346
41-50	0.4111	0.3973		41-50	0.1023	0.0307



(a)

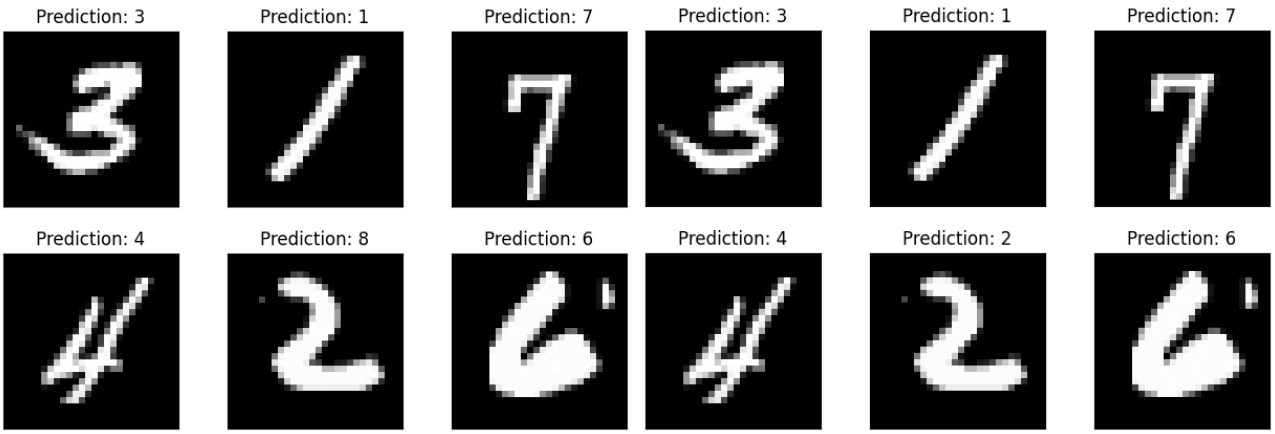


(b)

Figure 8: Loss vs Number of Epochs with adding Dropout layer = (a) False and (b) True

5.2.3 Number of predictions with added hidden layers:

The number of correct predictions for the best accuracy achieved during testing is after epoch 50. For the testing dataset, with an accuracy of 86.14% and 99.03% for no dropout layers and added dropout and fully connected layers, the number of correct predictions is near 8600 and 9900 among 10000 instances of the total dataset, which is better than the previous one and almost perfect.



(b) (b)

Figure 4: Some instances of with predicted labels with adding Dropout layer = (a) False and (b) True

5.2.4 Comparative analysis:

Accuracy:

- The first model achieved a maximum test accuracy of 86.14% after 50 epochs.
- The second model achieved a higher maximum test accuracy of 99.03% after 50 epochs.

The second model with added dropout and fully connected layers demonstrates a significant improvement in accuracy compared to the first model.

Loss:

- The first model's test loss fluctuates throughout training, starting at 0.4280 and decreasing to 0.4111.
- The second model's test loss shows a consistent decreasing trend, starting at 0.0529 and decreasing to 0.0307.

The second model with added dropout and fully connected layers exhibits much lower test loss values compared to the first model, indicating better convergence and model fit.

Training Dynamics:

- The first model's training accuracy increases gradually over epochs, while the test accuracy fluctuates.
- The second model demonstrates a more consistent improvement in both training and test accuracy over epochs, indicating better generalization.

Training Time:

- The training time for the second model might be longer due to its more complex model and potentially larger number of parameters.

Metric	Previous Model	New Model
Added Hidden Layers	None	Dropout and fully connected
Train Accuracy	85.71%	97.00%
Test Accuracy	86.13%	99.03%
Train Loss	0.4111	0.1023
Test Loss	0.3973	0.0307
Time (mins)	21.24 min	28.51 min

6 Conclusion

In summary, the model trained with a learning rate of 0.01 outperformed the one trained with a learning rate of 0.001, showcasing better accuracy and lower loss values. Additionally, the model with more hidden layers demonstrated superior performance compared to the simpler architecture with fewer layers. Furthermore, using a larger kernel size of 5x5 yielded better results than the 3x3 kernel size, indicating that the more complex architecture could capture more intricate patterns in the data. These findings emphasize the importance of tuning hyperparameters and designing architectures tailored to the specific task to

Reference:

[1] MNIST Dataset manual. https://conx.readthedocs.io/en/latest/MNIST.html