Convolution Neural Networks

Introduction:

In this problem, we perform a classification task on the MINST and CIFAR dataset using convolution neural network. Convolution Neural Networks take images as input, passes the image through a convolution network to learn feature embedding to perform clustering, classification, or other machine learning tasks.

Initially, the image is passed through a convolution network to learn feature vector embeddings. The feature vector is flattened and sent as input to a fully connected layer for further processing. The convolution networks not only enable to learn the feature vectors but also enables to reduce the dimensionality complexity of the problem /images. The most common used operations performed in the convolution networks are convolution, max pooling, padding Fully connected layers.

Dataset:

We use two datasets in the problem – CIFAR100 and MNIST datasets. The Dataset are downloaded and preprocessed batchwise using the Data loader class.

CIFAR100:

This dataset has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 super classes.

MNIST:

The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset. It is a dataset of 60,000 small square 28×28-pixel grayscale images of handwritten single digits between 0 and 9.

```
transform = transforms.Compose(
[transforms.ToTensor()])

Train_Data = torchvision.datasets.MNIST(root='./data', train=True,download=True, transform=transform)
Train_Loader = torch.utils.data.DataLoader(Train_Data, batch_size=batch_size,shuffle=True)

Test_Data = torchvision.datasets.MNIST(root='./data', train=False,download=True, transform=transform)
Test_Loader = torch.utils.data.DataLoader(Test_Data, batch_size=batch_size,shuffle=False)
```

Model architecture:

The Table indicates the operation in the network sequentially, number of input channels, number of output channels, Kernel size, stride, input shape, output shape and activation functions.

```
class CNN_Module(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1=nn.Conv2d(in_channels=1, out_channels=32, kernel_size=(3,3), stride=(1,1))
        self.pool1=nn.MaxPool2d(2,2)
        self.pool2=nn.MaxPool2d(2,2)
        self.conv2=nn.Conv2d(in_channels=32, out_channels=64, kernel_size=(5,5), stride=(1,1))
        self.fc1=nn.Linear(1024,1024)
        self.fc2=nn.Linear(1024,10)

        self.relu1=nn.ReLU()
        self.relu2=nn.ReLU()
        self.relu3=nn.ReLU()
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu1(x)
        x = self.pool1(x)
        x = self.conv2(x)
        x = self.relu2(x)
        x = self.pool2(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = self.relu3(x)
        x = self.fc2(x)
        x = self.softmax(x)
        return x
```

```
class CNN_Module(nn.Module):
def __init__(self):
    super().__init__()
    self.conv1=nn.Conv2d(in_channels=3, out_channels=64, kernel_size=(3,3), stride=(1,1))
    self.pool1=nn.MaxPool2d(2,2)
    self.pool2=nn.MaxPool2d(2,2)
    self.conv2=nn.Conv2d(in_channels=64, out_channels=128, kernel_size=(5,5), stride=(1,1))
    self.fc1=nn.Linear(3200,1024)
    self.fc2=nn.Linear(1024,100)

    self.relu1=nn.ReLU()
    self.relu2=nn.ReLU()
    self.relu3=nn.ReLU()
    self.softmax = nn.Softmax(dim=1)

def forward(self, x):
    x = self.conv1(x)
    x = self.relu1(x)
    x = self.pool1(x)
    x = self.conv2(x)
    x = self.relu2(x)
    x = self.pool2(x)
    x = torch.flatten(x, 1)
    x = self.fc1(x)
    x = self.relu3(x)
    x = self.fc2(x)
    x = self.softmax(x)
    return x
```

Convolution Network Sequence for MINST dataset:

Layer	I/P channels	O/P channels	Kernel Size	Stride	I/P shape	O/P Shape	Activation
Conv1	1	32	3,3	1,1	28*28	26*26	Relu
Maxpool2d-1	32	32	2,2		26*26	13*13	
Conv2	32	64	5,5	1,1	13*13	9*9	Relu
Maxpool2d-2	32	64	2,2		9*9	4*4	
Flatten	64				4*4	1024	
Fully Connected-1	1	1			1024	1024	Relu
Fully Connected-2	1	1			1024	10	Relu
SoftMax Layer	1	1			10	10	

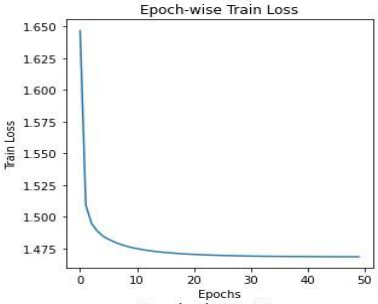
Convolution Network Sequence for CIFAR100 dataset:

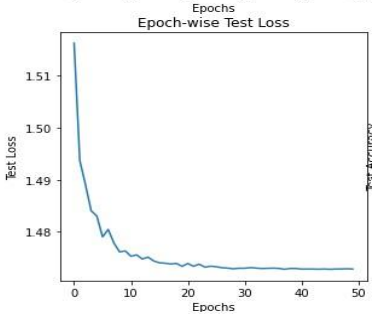
Layer	I/P channels	O/P channels	Kernel Size	Stride	I/P shape	O/P Shape	Activation
Conv1	3	64	3,3	1,1	32*32	30*30	Relu
Maxpool2d-1	64	64	2,2		30*30	15*15	
Conv2	64	128	5,5	1,1	15*15	11*11	Relu
Maxpool2d-2	64	128	2,2		11*11	5*5	
Flatten	128				5*5	3200	
Fully Connected-1	1	1			3200	1024	Relu
Fully Connected-2	1	1			1024	100	Relu
SoftMax Layer	1	1			100	100	

Evaluation:

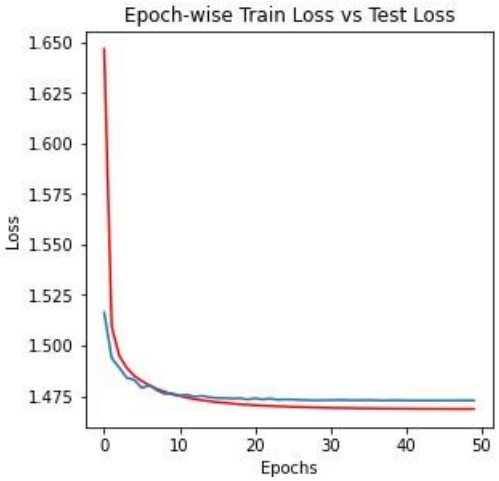
The model parameters used for the evaluation are Batch Size = 32, learning rate = 1e-4 Epochs = 50. Training and testing Loss is calculated for each epoch. The loss is plotted with respect to epochs. As the number of epochs increases the loss values decrease and flatline after certain number of epochs.

MNIST:



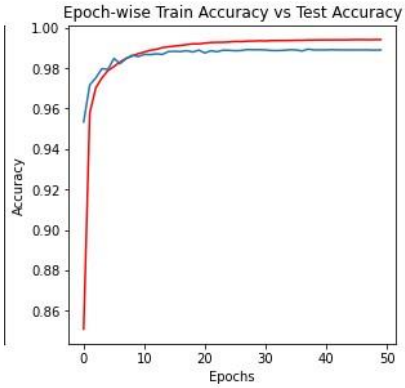


The comparison between the Training and validation loss indicates no overfitting between the training and testing.



In each epoch, train dataset and testing datasets are processed. The accuracy is calculated for each epoch.

MNIST:



Performance of the model on the testing dataset is evaluated. Precision, Recall, F1-Score, and accuracy are calculated.

MNIST:

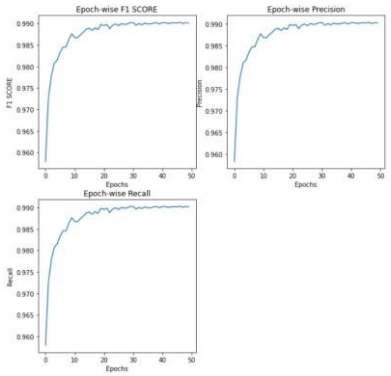
The accuracy reported by the model:

`Train Accuracy: 0.9945166666666667 Test Accuracy: 0.9901`

`F1 SCORE: 0.9900974650973656 Precision: 0.9901163235115688 Recall: 0.9901`

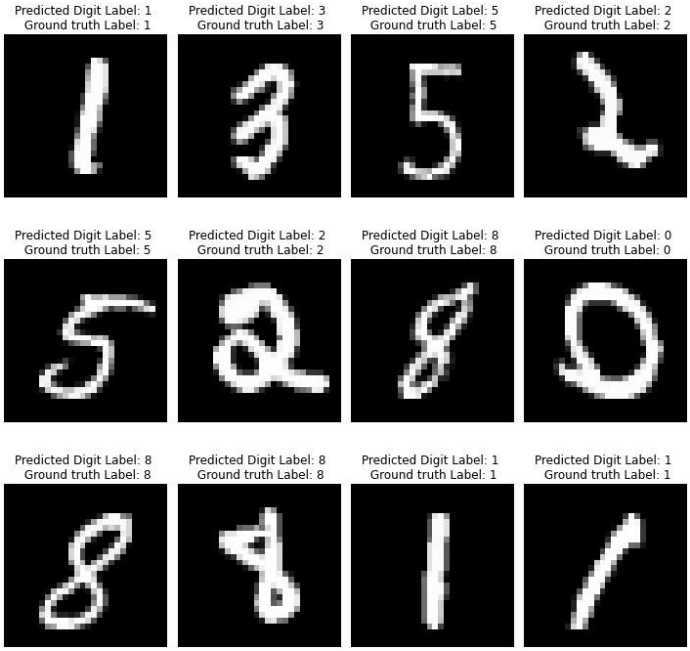
We also plot graphs of the F1-score, Precision, Recall, Accuracy as the epochs increases to analyze the trend.

MNIST:



Sample Example Plot:

In this section, we plot 10 examples with their ground truth and predicted class. **MNIST:**



Feature Maps and Kernel Filters:

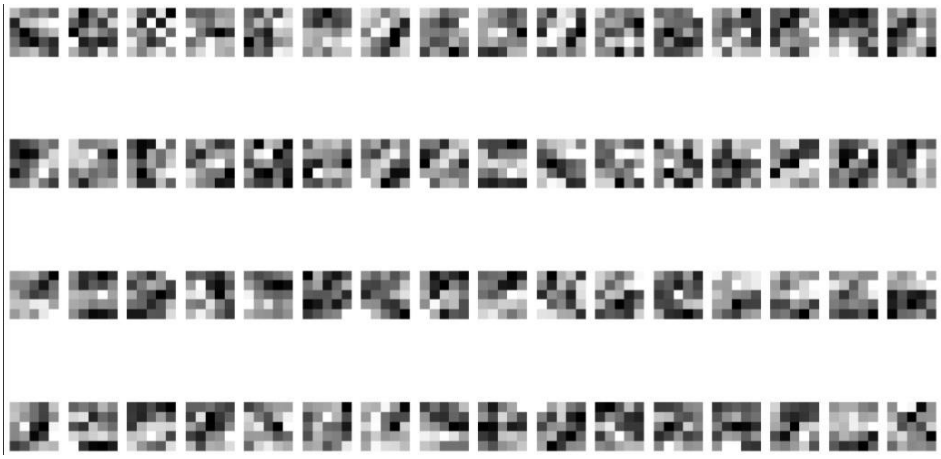
In this section, we visualize the feature maps and Kernel filters.

Shown below Kernel filters of the first and second convolution layers.

MNIST:

Convolution filters of Layer1 of 3*3 filter size:

Convolution filters of Layer2 of 5*5 filter size:



Shown below Feature Maps of the second convolution layers for four data inputs.

MNIST:

Figure shows the activation maps of the 4 training dataset images at conv1:

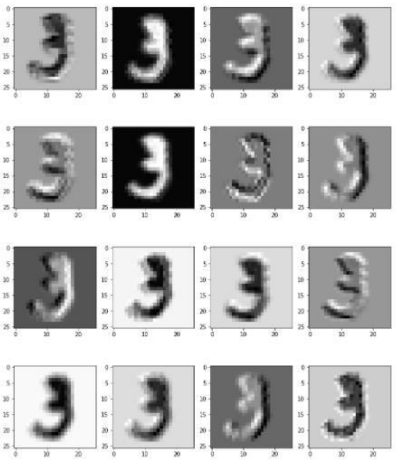
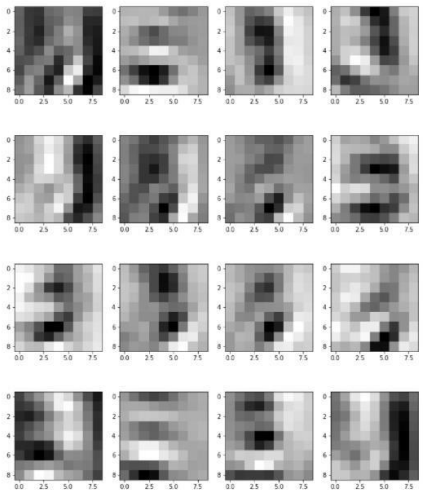


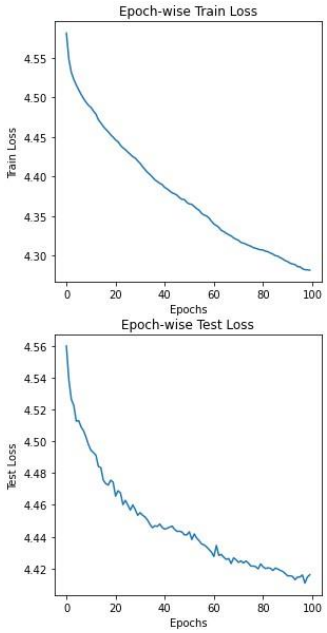
Figure shows the activation maps of the 4 training dataset images at conv2:



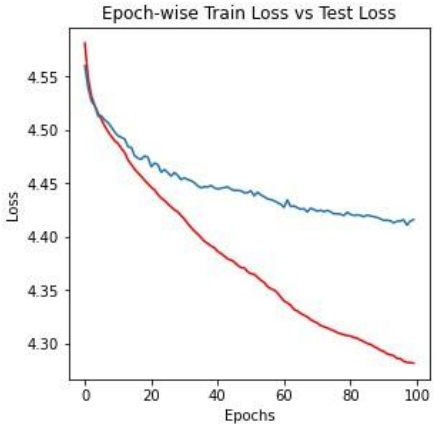
Evaluation:

The model parameters used for the evaluation are Batch Size = 32, learning rate = 1e-4 Epochs = 50. Training and testing Loss is calculated for each epoch. The loss is plotted with respect to epochs. As the number of epochs increases the loss values decreases.

CIFAR100:



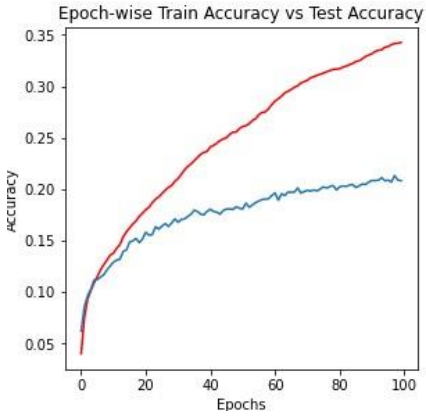
The comparison between the Training and testing loss indicates overfitting between the training and testing as there is huge difference between the training and testing loss after ~20 epochs.



In each epoch, train dataset and testing datasets are processed. The accuracy is calculated for each epoch. There is overfitting between the training and testing as there is huge difference between the

training and testing accuracy after ~20 epochs this is due to the model simplicity. If more convolutions are added, then the model learns better performs with a better accuracy and less overfitting.

CIFAR100:



Performance of the model on the testing dataset is evaluated. Precision, Recall, F1-Score, and accuracy are calculated.

CIFAR100:

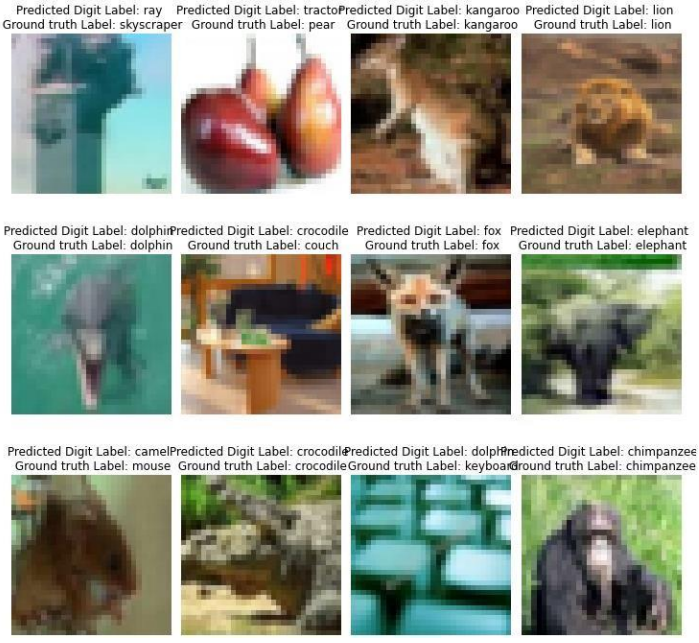
The accuracy reported by the model:

`Final Train Accuracy: 0.3621 Final Train Loss: 4.261431006824269 Final Test Accuracy: 0.2248 Final Test Loss: 4.397375644392269`

F1 SCORE: `0.15527068269575928` Precision: `0.12604492659560593` Recall: `0.22479999`

Sample Example Plot:

In this section, we plot 10 examples with their ground truth and predicted class. **CIFAR100:**



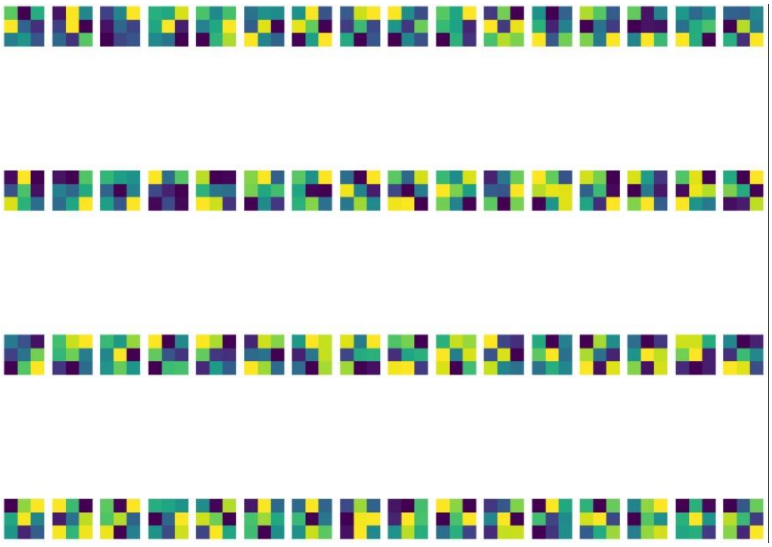
Feature Maps and Kernel Filters:

In this section, we visualize the feature maps and Kernel filters.

Shown below Kernel filters of the first and second convolution layers.

CIFAR100:

Convolution filters of Layer1 of 3*3 filter size:



Convolution filters of Layer2 of 5*5 filter size:

Shown below Feature Maps of the second convolution layers for four data inputs.

CIFAR100:

Figure shows the activation maps of the 4 training dataset images at conv1:

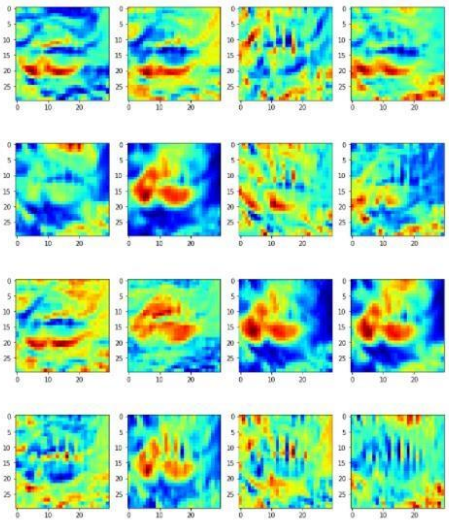


Figure shows the activation maps of the 4 training dataset images at conv2: