Name: Shubham Sarda Entry Number: 2018TT10958

Report: Assignment 2.2

PART A

In this part, using the model specified in assignment statement, just after 8 epochs, public test accuracy of **0.974** was achieved. Train loss (cross entropy loss) was reduced to **0.048.**

Training Specifications: Adam optimizer, learning rate=1e-4, epochs=8, batch size=200, loss function: Cross entropy loss, Model Parameters: ~1.4 Million

Time required for the training was 331s on hpc GPU

Dataset Used: Devanagri Handwritten Character Dataset

Same data was also trained with the best artificial neural network model found D part of Assignment 2.1

ANN Model Specifications:

Neural Network Architecture: [512,256,256,46], Activation Function: relu, Model Parameters: ~.74 million

Training Specifications: epochs=8, Momentum Optimizer, learning rate=.03, batch size=200, loss function: Cross entropy loss

Results: Train loss= **.051**, Test accuracy=**.929**

Time required for training was 74 seconds on personal CPU.

Hence, we see that while ANN are faster to train in this case due to simplified architecture and lesser parameters, but they aren't able to capture features of image data as good as CNN models since they flatten the image in the first step itself resulting in loss of important information. CNNs provide much better accuracy for same number of epochs and without and tuning

PART B

In this part, using the model specified in assignment statement, after 5 epochs, public test accuracy of **0.703** was achieved. Train loss (cross entropy loss) was reduced to **0.63.**

Training Specifications: Adam optimizer, learning rate=1e-4, epochs=85 batch size=200, loss function: Cross entropy loss

Model Parameters: ~2.6 Million

Time required for the training was 214s on hpc GPU

Dataset Used: CIFAR 10

Same data was also trained with the best artificial neural network model found D part of Assignment 2.1

ANN Model Specifications:

Neural Network Architecture: [512,256,256,10], Activation Function: relu

Model Parameters: ~.74 Million

Training Specifications: epochs=5, Momentum Optimizer, learning rate=.03, batch size=200, loss function: Cross entropy loss

Results: Train loss= **1.41**, Test accuracy=**.46**

Time required for training was 78 seconds on personal CPU.

Here again, we see that while ANN are faster to train because of the simplified architecture and lesser parameters. Here the difference in accuracy is also very large compared to previous case due to increased complexity of the dataset. Hence, CNNs provide much better accuracy for same number of epochs and without and tuning.

PART C

Dataset Used: CIFAR 10

In this competitive part, initially several feature engineering/data augmentation techniques were tested on the model used in part B and trained for 25 minutes. Training parameters were same as those used in Part B. **num_workers** can be changed according to system specifications (currently set at 0).

Reference: https://pytorch.org/vision/stable/transforms.html

Note: Most of these transformations require PIL Image as input therefore in those cases, image was first converted to PIL Image, then then these transformations were performed and finally converted back to tensor.

RandomChoice: applies one out of given list of transformations

RandomRotation(a): Rotates the image by a random angle between -a to a

RandomAffine : used for random translation and shearing

AutoAugment: Does default augmentation used in ImageNet

ColorJitter: Used to modify brightness, saturation, contrast and hue

RandomHorizontalFlip: Randomly flips the image horizontally with probability 0.5

RandomVerticalFlip: Randomly flips the image vertically with probability 0.5

RandomErasing: Random erasing some part of the image

Data Augmentation Performed	Test Accuracy
No augmentation	0.714
transforms.RandomChoice([transforms.RandomRotation(5), transforms.RandomAffine(0,translate=(.1,.1))])	**0.793**
transforms.RandomChoice([transforms.RandomRotation(45), transforms.RandomAffine(0,translate=(.5,.5))])	.733
transforms.AutoAugment()	.755
transforms.RandomRotation(5),transforms.RandomAffine(0,translate=(.1,.1)), transforms.ColorJitter(.1,.1,0,0)	.7895
transforms.RandomChoice([transforms.RandomRotation(10), transforms.RandomAffine(0,translate=(.2,.2))])	0.785
transforms.RandomChoice([transforms.RandomRotation(5), transforms.RandomAffine(0,translate=(.1,.1)), transforms.RandomHorizontalFlip(), transforms.RandomVerticalFlip(), transforms.ColorJitter(.1,.1,.1,.1)	0.77
transforms.RandomErasing()	0.732
transforms.RandomRotation(5), transforms.RandomAffine(0,translate=(.1,.1)), transforms.RandomGrayscale(.05)	.792

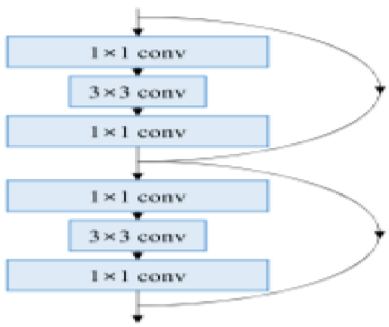
Augmentation with highest accuracy was selected and grid search was used to select the best learning rate. Optimal Learning rate was found to be 1e-3, gave an accuracy of **0.81** with the model from part B and data augmentation of random rotation and affine translation.

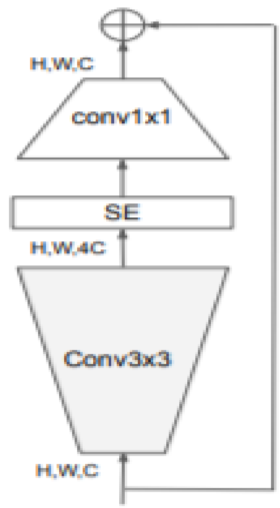
Now, different models were created to and similar augmentation and learning rate was used in training. Multiple models were created with different architectures, small tweaks in established models etc. We were restricted to limit **Model Parameters<= 3 million**. In this report, only the best models/ideas have been mentioned. Complete code of all the experiments can be found in **Part C** section of **Assignment2.2.ipynb**.

Best Models

1. Fused Convolution Blocks (Reverse Bottleneck) and Residual Connection Fused blocks were implemented by taking inspiration from https://arxiv.org/pdf/2104.00298v3.pdf, https://arxiv.org/pdf/1610.02915v4.pdf

Bottleneck structure and fused block structures haven given below





Fused-MBConv

Modified Fused Block: Due to Limited parameter constraint: fused block was shortened to two layers:

3 X 3 Conv(Input Channels=C, Output Channels=4C) → 1 X 1 Conv(Input Channels=4C, Output Channels=C) (Skip connection was also removed)

Training Specs: batch Size=200, Data Augmentation: Random Choice of Rotation (5 degree) and Affine Translation (0.1 in X and 0.1 in Y direction), Adam Optimizer, Lr=1e-3, Epochs=100

Similar training specifications were

Results: Public Test Accuracy: **.901**

```
class fused_block(nn.Module):
    def __init__(self,i):
        super(fused_block,self).__init__()
        self.c1=nn.Conv2d(i,4*i,3,padding='same')
        self.bn1=nn.BatchNorm2d(4*i)
        self.c2=nn.Conv2d(4*i,i,1,padding='same')
        self.bn2=nn.BatchNorm2d(i)
    def __call__(self,X):
        return F.relu(self.bn2(self.c2(F.relu(self.bn1(self.c1(X))))))
```

This block was used as basic building block of the architecture:

```
CNN(
  (c1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fb1): fused_block(
    (c1): Conv2d(64, 256, kernel_size=(3, 3), stride=(1, 1), padding=same)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (c2): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), padding=same)
    (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (p1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (c2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fb2): fused_block(
    (c1): Conv2d(128, 512, kernel_size=(3, 3), stride=(1, 1), padding=same)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (c2): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), padding=same)
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (p2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (c3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (p3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (p4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (c4): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (bn4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc0): Linear(in_features=2048, out_features=256, bias=True)
  (fc1): Linear(in_features=256, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=10, bias=True)
)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
       BatchNorm2d-2           [-1, 64, 32, 32]             128
            Conv2d-3          [-1, 256, 32, 32]         147,712
       BatchNorm2d-4          [-1, 256, 32, 32]             512
            Conv2d-5           [-1, 64, 32, 32]          16,448
       BatchNorm2d-6           [-1, 64, 32, 32]             128
         MaxPool2d-7           [-1, 64, 16, 16]               0
            Conv2d-8          [-1, 128, 16, 16]          73,856
       BatchNorm2d-9          [-1, 128, 16, 16]             256
           Conv2d-10          [-1, 512, 16, 16]         590,336
      BatchNorm2d-11          [-1, 512, 16, 16]           1,024
           Conv2d-12          [-1, 128, 16, 16]          65,664
      BatchNorm2d-13          [-1, 128, 16, 16]             256
        MaxPool2d-14            [-1, 128, 8, 8]               0
           Conv2d-15            [-1, 256, 8, 8]         295,168
      BatchNorm2d-16            [-1, 256, 8, 8]             512
        MaxPool2d-17            [-1, 256, 4, 4]               0
           Conv2d-18            [-1, 512, 4, 4]       1,180,160
      BatchNorm2d-19            [-1, 512, 4, 4]           1,024
        MaxPool2d-20            [-1, 512, 2, 2]               0
           Linear-21                  [-1, 256]         524,544
           Linear-22                  [-1, 128]          32,896
           Linear-23                   [-1, 10]           1,290
================================================================
Total params: 2,933,706
Trainable params: 2,933,706
Non-trainable params: 0
```

2. Two layered fused block was also with residual connection

Test Accuracy: .896

```
# resnet style fused block
class fused_block(nn.Module):
    def __init__(self,i):
        super(fused_block,self).__init__()
        self.c1=nn.Conv2d(i,4*i,3,padding='same')
        self.bn1=nn.BatchNorm2d(4*i)
        self.c2=nn.Conv2d(4*i,i,1,padding='same')
        self.bn2=nn.BatchNorm2d(i)

    def __call__(self,X):
        temp=self.bn2(self.c2(F.relu(self.bn1(self.c1(X)))))
        temp+=X
        return F.relu(temp)
```

Model Parameters and layer details remains same as only skip connection is added.

3. Another architecture was made using 3 layered fused block. However, to decrease model parameters (<3M), 1*1 conv was performed in first and last layers and 3*3 conv in middle layer

```
# 3 layered residual fused block
class fused_block(nn.Module):
    def __init__(self,i):
        super(fused_block,self).__init__()
        self.c1=nn.Conv2d(i,4*i,1,padding='same')
        self.bn1=nn.BatchNorm2d(4*i)
        self.c2=nn.Conv2d(4*i,i,3,padding='same')
        self.bn2=nn.BatchNorm2d(i)
        self.c3=nn.Conv2d(i,i,1,padding='same')
        self.bn3=nn.BatchNorm2d(i)

    def __call__(self,X):
        temp=self.bn3(self.c3(F.relu(self.bn2(self.c2(F.relu(self.bn1(self.c1(X))))))))
        temp+=X
        return F.relu(temp)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
       BatchNorm2d-2           [-1, 64, 32, 32]             128
            Conv2d-3          [-1, 256, 32, 32]          16,640
       BatchNorm2d-4          [-1, 256, 32, 32]             512
            Conv2d-5           [-1, 64, 32, 32]         147,520
       BatchNorm2d-6           [-1, 64, 32, 32]             128
            Conv2d-7           [-1, 64, 32, 32]           4,160
       BatchNorm2d-8           [-1, 64, 32, 32]             128
         MaxPool2d-9           [-1, 64, 16, 16]               0
           Conv2d-10          [-1, 128, 16, 16]          73,856
      BatchNorm2d-11          [-1, 128, 16, 16]             256
           Conv2d-12          [-1, 512, 16, 16]          66,048
      BatchNorm2d-13          [-1, 512, 16, 16]           1,024
           Conv2d-14          [-1, 128, 16, 16]         589,952
      BatchNorm2d-15          [-1, 128, 16, 16]             256
           Conv2d-16          [-1, 128, 16, 16]          16,512
      BatchNorm2d-17          [-1, 128, 16, 16]             256
        MaxPool2d-18            [-1, 128, 8, 8]               0
           Conv2d-19            [-1, 256, 8, 8]         295,168
      BatchNorm2d-20            [-1, 256, 8, 8]             512
        MaxPool2d-21            [-1, 256, 4, 4]               0
           Conv2d-22            [-1, 512, 4, 4]       1,180,160
      BatchNorm2d-23            [-1, 512, 4, 4]           1,024
        MaxPool2d-24            [-1, 512, 2, 2]               0
           Linear-25                  [-1, 256]         524,544
           Linear-26                  [-1, 128]          32,896
           Linear-27                   [-1, 10]           1,290
================================================================
Total params: 2,954,762
Trainable params: 2,954,762
Non-trainable params: 0
```

This model gave accuracy of .898 on public test.

Other Experiments Done before reaching final model:

1. Several more Fully connected layers were added in the down sampling part (Fully Connected Network) of model used in part B.
   Test Accuracy: 0.818

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 30, 30]	896
BatchNorm2d-2	[-1, 32, 30, 30]	64
MaxPool2d-3	[-1, 32, 15, 15]	0
Conv2d-4	[-1, 64, 13, 13]	18,496
BatchNorm2d-5	[-1, 64, 13, 13]	128
MaxPool2d-6	[-1, 64, 6, 6]	0
Conv2d-7	[-1, 512, 4, 4]	295,424
BatchNorm2d-8	[-1, 512, 4, 4]	1,024
MaxPool2d-9	[-1, 512, 2, 2]	0
Conv2d-10	[-1, 1024, 1, 1]	2,098,176
Linear-11	[-1, 256]	262,400
Dropout-12	[-1, 256]	0
Linear-13	[-1, 128]	32,896
Linear-14	[-1, 64]	8,256
Linear-15	[-1, 10]	650

Total params: 2,718,410
Trainable params: 2,718,410
Non-trainable params: 0

Taking inspiration from VGG models, smaller and custom variants of the same were implemented.

2.

Test Accuracy: 0.8635

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 32, 32]	896
BatchNorm2d-2	[-1, 32, 32, 32]	64
Conv2d-3	[-1, 32, 32, 32]	9,248
BatchNorm2d-4	[-1, 32, 32, 32]	64
MaxPool2d-5	[-1, 32, 16, 16]	0
Conv2d-6	[-1, 64, 16, 16]	18,496
BatchNorm2d-7	[-1, 64, 16, 16]	128
Conv2d-8	[-1, 64, 16, 16]	36,928
BatchNorm2d-9	[-1, 64, 16, 16]	128
MaxPool2d-10	[-1, 64, 8, 8]	0
Conv2d-11	[-1, 128, 8, 8]	73,856
BatchNorm2d-12	[-1, 128, 8, 8]	256
Conv2d-13	[-1, 128, 8, 8]	147,584
BatchNorm2d-14	[-1, 128, 8, 8]	256
MaxPool2d-15	[-1, 128, 4, 4]	0
Conv2d-16	[-1, 256, 4, 4]	295,168
BatchNorm2d-17	[-1, 256, 4, 4]	512
Conv2d-18	[-1, 256, 4, 4]	590,080
BatchNorm2d-19	[-1, 256, 4, 4]	512
MaxPool2d-20	[-1, 256, 2, 2]	0
Linear-21	[-1, 512]	524,800
Dropout-22	[-1, 512]	0
Linear-23	[-1, 256]	131,328
Linear-24	[-1, 128]	32,896
Linear-25	[-1, 64]	8,256
Linear-26	[-1, 10]	650

Total params: 1,872,106
Trainable params: 1,872,106
Non-trainable params: 0

3. Test Accuracy: 0.863

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             896
       BatchNorm2d-2           [-1, 32, 32, 32]              64
            Conv2d-3           [-1, 32, 32, 32]           9,248
       BatchNorm2d-4           [-1, 32, 32, 32]              64
         MaxPool2d-5           [-1, 32, 16, 16]               0
            Conv2d-6           [-1, 64, 16, 16]          18,496
       BatchNorm2d-7           [-1, 64, 16, 16]             128
            Conv2d-8           [-1, 64, 16, 16]          36,928
       BatchNorm2d-9           [-1, 64, 16, 16]             128
        MaxPool2d-10             [-1, 64, 8, 8]               0
           Conv2d-11            [-1, 128, 8, 8]          73,856
      BatchNorm2d-12            [-1, 128, 8, 8]             256
           Conv2d-13            [-1, 128, 8, 8]         147,584
      BatchNorm2d-14            [-1, 128, 8, 8]             256
        MaxPool2d-15            [-1, 128, 4, 4]               0
           Conv2d-16            [-1, 256, 4, 4]         295,168
      BatchNorm2d-17            [-1, 256, 4, 4]             512
           Conv2d-18            [-1, 256, 4, 4]         590,080
      BatchNorm2d-19            [-1, 256, 4, 4]             512
           Linear-20                  [-1, 256]       1,048,832
          Dropout-21                  [-1, 256]               0
           Linear-22                  [-1, 128]          32,896
           Linear-23                   [-1, 64]           8,256
           Linear-24                   [-1, 10]             650
================================================================
Total params: 2,264,810
Trainable params: 2,264,810
Non-trainable params: 0
```

4. Test accuracy: 0.872

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             896
       BatchNorm2d-2           [-1, 32, 32, 32]              64
            Conv2d-3           [-1, 32, 32, 32]           9,248
       BatchNorm2d-4           [-1, 32, 32, 32]              64
            Conv2d-5           [-1, 32, 32, 32]           9,248
       BatchNorm2d-6           [-1, 32, 32, 32]              64
         MaxPool2d-7           [-1, 32, 16, 16]               0
            Conv2d-8           [-1, 64, 16, 16]          18,496
       BatchNorm2d-9           [-1, 64, 16, 16]             128
           Conv2d-10           [-1, 64, 16, 16]          36,928
      BatchNorm2d-11           [-1, 64, 16, 16]             128
           Conv2d-12           [-1, 64, 16, 16]          36,928
      BatchNorm2d-13           [-1, 64, 16, 16]             128
        MaxPool2d-14             [-1, 64, 8, 8]               0
           Conv2d-15            [-1, 128, 8, 8]          73,856
      BatchNorm2d-16            [-1, 128, 8, 8]             256
           Conv2d-17            [-1, 128, 8, 8]         147,584
      BatchNorm2d-18            [-1, 128, 8, 8]             256
           Conv2d-19            [-1, 128, 8, 8]         147,584
      BatchNorm2d-20            [-1, 128, 8, 8]             256
        MaxPool2d-21            [-1, 128, 4, 4]               0
           Conv2d-22            [-1, 256, 4, 4]         295,168
      BatchNorm2d-23            [-1, 256, 4, 4]             512
           Conv2d-24            [-1, 256, 4, 4]         590,080
      BatchNorm2d-25            [-1, 256, 4, 4]             512
           Linear-26                  [-1, 256]       1,048,832
          Dropout-27                  [-1, 256]               0
           Linear-28                  [-1, 128]          32,896
           Linear-29                   [-1, 64]           8,256
           Linear-30                   [-1, 10]             650
================================================================
Total params: 2,459,018
Trainable params: 2,459,018
Non-trainable params: 0
```

5. Test Accuracy: 0.861

```
       Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             896
       BatchNorm2d-2           [-1, 32, 32, 32]              64
            Conv2d-3           [-1, 32, 32, 32]           9,248
       BatchNorm2d-4           [-1, 32, 32, 32]              64
         MaxPool2d-5           [-1, 32, 16, 16]               0
            Conv2d-6           [-1, 64, 16, 16]          18,496
       BatchNorm2d-7           [-1, 64, 16, 16]             128
            Conv2d-8           [-1, 64, 16, 16]          36,928
       BatchNorm2d-9           [-1, 64, 16, 16]             128
        MaxPool2d-10             [-1, 64, 8, 8]               0
           Conv2d-11            [-1, 128, 8, 8]          73,856
      BatchNorm2d-12            [-1, 128, 8, 8]             256
           Conv2d-13            [-1, 128, 8, 8]         147,584
      BatchNorm2d-14            [-1, 128, 8, 8]             256
        MaxPool2d-15            [-1, 128, 4, 4]               0
           Conv2d-16            [-1, 256, 4, 4]         295,168
      BatchNorm2d-17            [-1, 256, 4, 4]             512
           Conv2d-18            [-1, 256, 4, 4]         590,080
      BatchNorm2d-19            [-1, 256, 4, 4]             512
           Conv2d-20            [-1, 128, 4, 4]         295,040
      BatchNorm2d-21            [-1, 128, 4, 4]             256
           Conv2d-22            [-1, 128, 4, 4]         147,584
      BatchNorm2d-23            [-1, 128, 4, 4]             256
           Conv2d-24             [-1, 64, 4, 4]          73,792
      BatchNorm2d-25             [-1, 64, 4, 4]             128
           Conv2d-26             [-1, 64, 4, 4]          36,928
      BatchNorm2d-27             [-1, 64, 4, 4]             128
           Linear-28                   [-1, 512]         524,800
          Dropout-29                   [-1, 512]               0
           Linear-30                   [-1, 256]         131,328
           Linear-31                   [-1, 128]          32,896
           Linear-32                    [-1, 64]           8,256
           Linear-33                    [-1, 10]             650
================================================================
Total params: 2,426,218
```

6. Test Accuracy: 0.864

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             896
       BatchNorm2d-2           [-1, 32, 32, 32]              64
            Conv2d-3           [-1, 32, 32, 32]           9,248
       BatchNorm2d-4           [-1, 32, 32, 32]              64
            Conv2d-5           [-1, 32, 32, 32]           9,248
       BatchNorm2d-6           [-1, 32, 32, 32]              64
         MaxPool2d-7           [-1, 32, 16, 16]               0
            Conv2d-8           [-1, 64, 16, 16]          18,496
       BatchNorm2d-9           [-1, 64, 16, 16]             128
           Conv2d-10           [-1, 64, 16, 16]          36,928
      BatchNorm2d-11           [-1, 64, 16, 16]             128
           Conv2d-12           [-1, 64, 16, 16]          36,928
      BatchNorm2d-13           [-1, 64, 16, 16]             128
        MaxPool2d-14             [-1, 64, 8, 8]               0
           Conv2d-15            [-1, 128, 8, 8]          73,856
      BatchNorm2d-16            [-1, 128, 8, 8]             256
           Conv2d-17            [-1, 128, 8, 8]         147,584
      BatchNorm2d-18            [-1, 128, 8, 8]             256
           Conv2d-19            [-1, 128, 8, 8]         147,584
      BatchNorm2d-20            [-1, 128, 8, 8]             256
        MaxPool2d-21            [-1, 128, 4, 4]               0
           Conv2d-22            [-1, 256, 4, 4]         295,168
      BatchNorm2d-23            [-1, 256, 4, 4]             512
           Conv2d-24            [-1, 256, 4, 4]         590,080
      BatchNorm2d-25            [-1, 256, 4, 4]             512
           Linear-26                  [-1, 256]       1,048,832
          Dropout-27                  [-1, 256]               0
           Linear-28                  [-1, 128]          32,896
           Linear-29                   [-1, 64]           8,256
           Linear-30                   [-1, 10]             650
================================================================
Total params: 2,459,018
Trainable params: 2,459,018
```

7. Test Accuracy:0.857 (Increased number of conv layers in VGG blocks by 1.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 32, 32]	896
BatchNorm2d-2	[-1, 32, 32, 32]	64
Conv2d-3	[-1, 32, 32, 32]	9,248
BatchNorm2d-4	[-1, 32, 32, 32]	64
Conv2d-5	[-1, 32, 32, 32]	9,248
BatchNorm2d-6	[-1, 32, 32, 32]	64
Conv2d-7	[-1, 32, 32, 32]	9,248
BatchNorm2d-8	[-1, 32, 32, 32]	64
MaxPool2d-9	[-1, 32, 16, 16]	0
Conv2d-10	[-1, 64, 16, 16]	18,496
BatchNorm2d-11	[-1, 64, 16, 16]	128
Conv2d-12	[-1, 64, 16, 16]	36,928
BatchNorm2d-13	[-1, 64, 16, 16]	128
Conv2d-14	[-1, 64, 16, 16]	36,928
BatchNorm2d-15	[-1, 64, 16, 16]	128
Conv2d-16	[-1, 64, 16, 16]	36,928
BatchNorm2d-17	[-1, 64, 16, 16]	128
MaxPool2d-18	[-1, 64, 8, 8]	0
Conv2d-19	[-1, 128, 8, 8]	73,856
BatchNorm2d-20	[-1, 128, 8, 8]	256
Conv2d-21	[-1, 128, 8, 8]	147,584
BatchNorm2d-22	[-1, 128, 8, 8]	256
Conv2d-23	[-1, 128, 8, 8]	147,584
BatchNorm2d-24	[-1, 128, 8, 8]	256
Conv2d-25	[-1, 128, 8, 8]	147,584
BatchNorm2d-26	[-1, 128, 8, 8]	256
MaxPool2d-27	[-1, 128, 4, 4]	0
Conv2d-28	[-1, 256, 4, 4]	295,168
BatchNorm2d-29	[-1, 256, 4, 4]	512
Conv2d-30	[-1, 256, 4, 4]	590,080
BatchNorm2d-31	[-1, 256, 4, 4]	512
Linear-32	[-1, 256]	1,048,832
Dropout-33	[-1, 256]	0
Linear-34	[-1, 128]	32,896
Linear-35	[-1, 64]	8,256
Linear-36	[-1, 10]	650

Total params: 2,653,226
Trainable params: 2,653,226
Non-trainable params: 0

Taking inspiration from the Inception blocks used in modern CNNs, the following networks were implemented. Code for Inception Block:

Inception module with dimensionality reductions: https://arxiv.org/pdf/1409.4842v1.pdf

Used 1x1 max pooling in inception block instead of 3x3 due to smaller image size (32x32)

```
class inception(nn.Module):
    def __init__(self,i):
        super(inception, self).__init__()
        self.br1_c1 = nn.Conv2d(i,64,1)
        self.bn1=nn.BatchNorm2d(64)
        self.br2_c1 = nn.Conv2d(i,96,1)
        self.bn2=nn.BatchNorm2d(96)
        self.br2_c2 = nn.Conv2d(96,128,3,padding=1)
        self.bn3=nn.BatchNorm2d(128)
        self.br3_c1 = nn.Conv2d(i,16,1)
        self.bn4=nn.BatchNorm2d(16)
        self.br3_c2 = nn.Conv2d(16,32,5,padding=2)
        self.bn5=nn.BatchNorm2d(32)
        self.br4_p1 = nn.MaxPool2d(i,1, padding=1)
        self.br4_c1 = nn.Conv2d(i,32,1)
        self.bn6=nn.BatchNorm2d(32)
    def forward(self, x):
        br1=F.relu(self.bn1(self.br1_c1(x)))
        br2=F.relu(self.bn3(self.br2_c2(F.relu(self.bn2(self.br2_c1(x))))))
        br3=F.relu(self.bn5(self.br3_c2(F.relu(self.bn4(self.br3_c1(x))))))
        br4= F.relu(self.bn6(self.br4_c1(self.br4_p1(x))))
        c=(br1, br2, br3, br4)
        return torch.cat(c, dim=1)
```

8. Test Accuracy:0.849

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]             256
       BatchNorm2d-2           [-1, 64, 32, 32]             128
            Conv2d-3           [-1, 96, 32, 32]             384
       BatchNorm2d-4           [-1, 96, 32, 32]             192
            Conv2d-5          [-1, 128, 32, 32]         110,720
       BatchNorm2d-6          [-1, 128, 32, 32]             256
            Conv2d-7           [-1, 16, 32, 32]              64
       BatchNorm2d-8           [-1, 16, 32, 32]              32
            Conv2d-9           [-1, 32, 32, 32]          12,832
      BatchNorm2d-10           [-1, 32, 32, 32]              64
        MaxPool2d-11            [-1, 3, 32, 32]               0
           Conv2d-12           [-1, 32, 32, 32]             128
      BatchNorm2d-13           [-1, 32, 32, 32]              64
        MaxPool2d-14          [-1, 256, 16, 16]               0
           Conv2d-15          [-1, 256, 16, 16]         590,080
      BatchNorm2d-16          [-1, 256, 16, 16]             512
           Conv2d-17          [-1, 256, 16, 16]         590,080
      BatchNorm2d-18          [-1, 256, 16, 16]             512
        MaxPool2d-19            [-1, 256, 8, 8]               0
           Conv2d-20            [-1, 512, 8, 8]       1,180,160
      BatchNorm2d-21            [-1, 512, 8, 8]           1,024
        MaxPool2d-22            [-1, 512, 2, 2]               0
           Linear-23                  [-1, 128]         262,272
           Linear-24                   [-1, 10]           1,290
================================================================
Total params: 2,751,050
Trainable params: 2,751,050
Non-trainable params: 0
----------------------------------------------------------------
```

9. Test Accuracy:.854. Competitive style inception was implemented using an architecture similar in **https://arxiv.org/pdf/1511.05635v1.pdf**

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	256
BatchNorm2d-2	[-1, 64, 32, 32]	128
Conv2d-3	[-1, 64, 32, 32]	1,792
BatchNorm2d-4	[-1, 64, 32, 32]	128
Conv2d-5	[-1, 64, 32, 32]	4,864
BatchNorm2d-6	[-1, 64, 32, 32]	128
Conv2d-7	[-1, 64, 32, 32]	9,472
BatchNorm2d-8	[-1, 64, 32, 32]	128
inception-9	[-1, 64, 32, 32]	0
MaxPool2d-10	[-1, 64, 16, 16]	0
Conv2d-11	[-1, 128, 16, 16]	8,320
BatchNorm2d-12	[-1, 128, 16, 16]	256
Conv2d-13	[-1, 128, 16, 16]	73,856
BatchNorm2d-14	[-1, 128, 16, 16]	256
Conv2d-15	[-1, 128, 16, 16]	204,928
BatchNorm2d-16	[-1, 128, 16, 16]	256
Conv2d-17	[-1, 128, 16, 16]	401,536
BatchNorm2d-18	[-1, 128, 16, 16]	256
inception-19	[-1, 128, 16, 16]	0
MaxPool2d-20	[-1, 128, 8, 8]	0
Conv2d-21	[-1, 256, 8, 8]	295,168
BatchNorm2d-22	[-1, 256, 8, 8]	512
Conv2d-23	[-1, 256, 8, 8]	590,080
BatchNorm2d-24	[-1, 256, 8, 8]	512
MaxPool2d-25	[-1, 256, 4, 4]	0
Conv2d-26	[-1, 512, 4, 4]	1,180,160
BatchNorm2d-27	[-1, 512, 4, 4]	1,024
MaxPool2d-28	[-1, 512, 2, 2]	0
Linear-29	[-1, 64]	131,136
Linear-30	[-1, 10]	650

Total params: 2,905,802
Trainable params: 2,905,802
Non-trainable params: 0

Taking inspiration from resnet structures, smaller variants of the same were implemented

10. Test Accuracy: 0.8807

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             864
       BatchNorm2d-2           [-1, 32, 32, 32]              64
            Conv2d-3           [-1, 32, 32, 32]           9,216
       BatchNorm2d-4           [-1, 32, 32, 32]              64
            Conv2d-5           [-1, 32, 32, 32]           9,216
       BatchNorm2d-6           [-1, 32, 32, 32]              64
         MaxPool2d-7           [-1, 32, 16, 16]               0
            Conv2d-8           [-1, 64, 16, 16]          18,432
       BatchNorm2d-9           [-1, 64, 16, 16]             128
           Conv2d-10           [-1, 64, 16, 16]          36,864
      BatchNorm2d-11           [-1, 64, 16, 16]             128
           Conv2d-12           [-1, 64, 16, 16]          36,864
      BatchNorm2d-13           [-1, 64, 16, 16]             128
        MaxPool2d-14             [-1, 64, 8, 8]               0
           Conv2d-15            [-1, 128, 8, 8]          73,728
      BatchNorm2d-16            [-1, 128, 8, 8]             256
           Conv2d-17            [-1, 128, 8, 8]         147,456
      BatchNorm2d-18            [-1, 128, 8, 8]             256
           Conv2d-19            [-1, 128, 8, 8]         147,456
      BatchNorm2d-20            [-1, 128, 8, 8]             256
        MaxPool2d-21            [-1, 128, 4, 4]               0
           Conv2d-22            [-1, 256, 4, 4]         295,168
      BatchNorm2d-23            [-1, 256, 4, 4]             512
           Conv2d-24            [-1, 256, 4, 4]         590,080
      BatchNorm2d-25            [-1, 256, 4, 4]             512
           Linear-26                  [-1, 256]       1,048,832
          Dropout-27                  [-1, 256]               0
           Linear-28                  [-1, 128]          32,896
           Linear-29                   [-1, 64]           8,256
           Linear-30                   [-1, 10]             650
================================================================
Total params: 2,458,346
Trainable params: 2,458,346
Non-trainable params: 0
----------------------------------------------------------------
```