# ECE 763: Computer Vision

## Project 2

Rahul Mishra

[200267922]

**Objectives:** Face image classification using a simple neural network to get familiar with steps of training neural networks.

**Dataset Used:** Wider Face Dataset which consists of 32,203 RGB images and 393,703 faces.

Link: http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/

We extracted the faces using the annotations of top left (x,y) and hight and width of the bounding box and then resized each image to RGB 60x60.

Train Faces: 10000

Train Non-Faces: 10000

Test Face: 1000

Test Face: 1000

**Step 1: Preprocessing the Dataset**

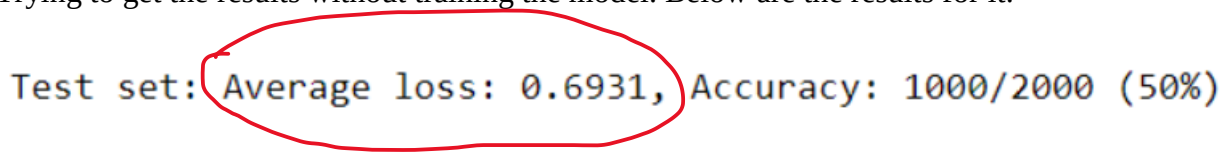
| | Train | Validation | Test |
|---|---|---|---|
| Face | 8000 | 2000 | 1000 |
| Non-Face | 8000 | 2000 | 1000 |

We calculate the mean and standard deviation of al the training images for each channel and normalize the whole dataset using these values.

We observe that training results without normalizing the dataset are bad and only improve when we normalize it.

**Step 2: Loss without training**

Trying to get the results without training the model. Below are the results for it:



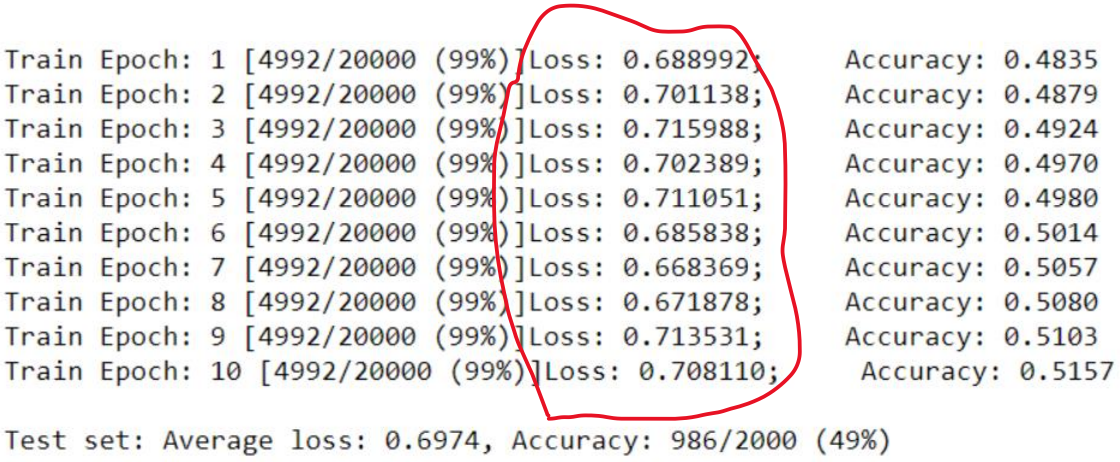

This is expected as the model is random and give the prediction randomly. This should be 50%.

Loss = 2.303log(2) = 0.6932

**Step 3: Too small learning rate**

Now, we train our model using SGD optimizer using

We choose learning rate = 1e-6. This learning late leads to stagnant learning of the model. Loss of the model is almost static and not learning at all.



**Step 4: Too high learning rate**

On the other hand, if we make learning rate too high, we observe that the learning is not converging, and loss is random. Due to the way PyTorch is written it's not showing us "NAN" but still we know that the model is not learning as accuracy is not changing.

```
(base) C:\Users\sport\Desktop\Project2>python main.py --lr 1000000
Train Samples = 16000
Validation Samples = 4000
Train Epoch: 1 [15872/16000 (99%)]Loss: 0.649199;     Accuracy: 0.626
Validation set: Average loss: 0.6868, Accuracy: 2506/4000 (63%)
Train Epoch: 2 [15872/16000 (99%)]Loss: 0.711699;     Accuracy: 0.626
Validation set: Average loss: 0.6903, Accuracy: 2492/4000 (62%)
Train Epoch: 3 [15872/16000 (99%)]Loss: 0.563262;     Accuracy: 0.625
Validation set: Average loss: 0.6895, Accuracy: 2495/4000 (62%)
Train Epoch: 4 [15872/16000 (99%)]Loss: 0.664824;     Accuracy: 0.626
Validation set: Average loss: 0.6885, Accuracy: 2499/4000 (62%)
Train Epoch: 5 [15872/16000 (99%)]Loss: 0.680449;     Accuracy: 0.627
Validation set: Average loss: 0.6860, Accuracy: 2509/4000 (63%)
Train Epoch: 6 [15872/16000 (99%)]Loss: 0.649199;     Accuracy: 0.627
Validation set: Average loss: 0.6908, Accuracy: 2490/4000 (62%)
Train Epoch: 7 [15872/16000 (99%)]Loss: 0.727324;     Accuracy: 0.627
Validation set: Average loss: 0.6885, Accuracy: 2499/4000 (62%)
Train Epoch: 8 [15872/16000 (99%)]Loss: 0.672637;     Accuracy: 0.627
Validation set: Average loss: 0.6863, Accuracy: 2508/4000 (63%)
Train Epoch: 9 [15872/16000 (99%)]Loss: 0.664824;     Accuracy: 0.625
Validation set: Average loss: 0.6903, Accuracy: 2492/4000 (62%)
Train Epoch: 10 [15872/16000 (99%)]Loss: 0.649199;     Accuracy: 0.627
Validation set: Average loss: 0.6908, Accuracy: 2490/4000 (62%)
Test set: Average loss: 0.6763, Accuracy: 1274/2000 (64%)
```

**Step 5: Zero Regularization and Overfitting**

We try to overfit the model on small dataset of only 20 images. We train our model for 200 epochs. Indeed, we see that our model is capable of overfitting the small dataset.

```
Train Epoch: 1 [18/20 (90%)]Loss: 0.691405;     Accuracy: 0.300
Train Epoch: 2 [18/20 (90%)]Loss: 0.701769;     Accuracy: 0.500
Train Epoch: 3 [18/20 (90%)]Loss: 0.689360;     Accuracy: 0.650
Train Epoch: 4 [18/20 (90%)]Loss: 0.691879;     Accuracy: 0.650
Train Epoch: 5 [18/20 (90%)]Loss: 0.692330;     Accuracy: 0.700
Train Epoch: 6 [18/20 (90%)]Loss: 0.673783;     Accuracy: 0.700
Train Epoch: 7 [18/20 (90%)]Loss: 0.672685;     Accuracy: 0.600
Train Epoch: 8 [18/20 (90%)]Loss: 0.691854;     Accuracy: 0.650
Train Epoch: 9 [18/20 (90%)]Loss: 0.684321;     Accuracy: 0.700
Train Epoch: 10 [18/20 (90%)]Loss: 0.697688;     Accuracy: 0.700
Train Epoch: 11 [18/20 (90%)]Loss: 0.688939;     Accuracy: 0.750
Train Epoch: 12 [18/20 (90%)]Loss: 0.695417;     Accuracy: 0.750
Train Epoch: 13 [18/20 (90%)]Loss: 0.696503;     Accuracy: 0.700
Train Epoch: 14 [18/20 (90%)]Loss: 0.644881;     Accuracy: 0.700
Train Epoch: 15 [18/20 (90%)]Loss: 0.644515;     Accuracy: 0.750
```

```
Train Epoch: 184 [18/20 (90%)]Loss: 0.313276;     Accuracy: 1.000
Train Epoch: 185 [18/20 (90%)]Loss: 0.314653;     Accuracy: 1.000
Train Epoch: 186 [18/20 (90%)]Loss: 0.313954;     Accuracy: 1.000
Train Epoch: 187 [18/20 (90%)]Loss: 0.313762;     Accuracy: 1.000
Train Epoch: 188 [18/20 (90%)]Loss: 0.313529;     Accuracy: 1.000
Train Epoch: 189 [18/20 (90%)]Loss: 0.313380;     Accuracy: 1.000
Train Epoch: 190 [18/20 (90%)]Loss: 0.313470;     Accuracy: 1.000
Train Epoch: 191 [18/20 (90%)]Loss: 0.313393;     Accuracy: 1.000
Train Epoch: 192 [18/20 (90%)]Loss: 0.313441;     Accuracy: 1.000
Train Epoch: 193 [18/20 (90%)]Loss: 0.313264;     Accuracy: 1.000
Train Epoch: 194 [18/20 (90%)]Loss: 0.313683;     Accuracy: 1.000
Train Epoch: 195 [18/20 (90%)]Loss: 0.313262;     Accuracy: 1.000
Train Epoch: 196 [18/20 (90%)]Loss: 0.313474;     Accuracy: 1.000
Train Epoch: 197 [18/20 (90%)]Loss: 0.313931;     Accuracy: 1.000
Train Epoch: 198 [18/20 (90%)]Loss: 0.313263;     Accuracy: 1.000
Train Epoch: 199 [18/20 (90%)]Loss: 0.313262;     Accuracy: 1.000
Train Epoch: 200 [18/20 (90%)]Loss: 0.313423;     Accuracy: 1.000

Test set: Average loss: 0.6837, Accuracy: 1224/2000 (61%)
```

**6. Final Results:**

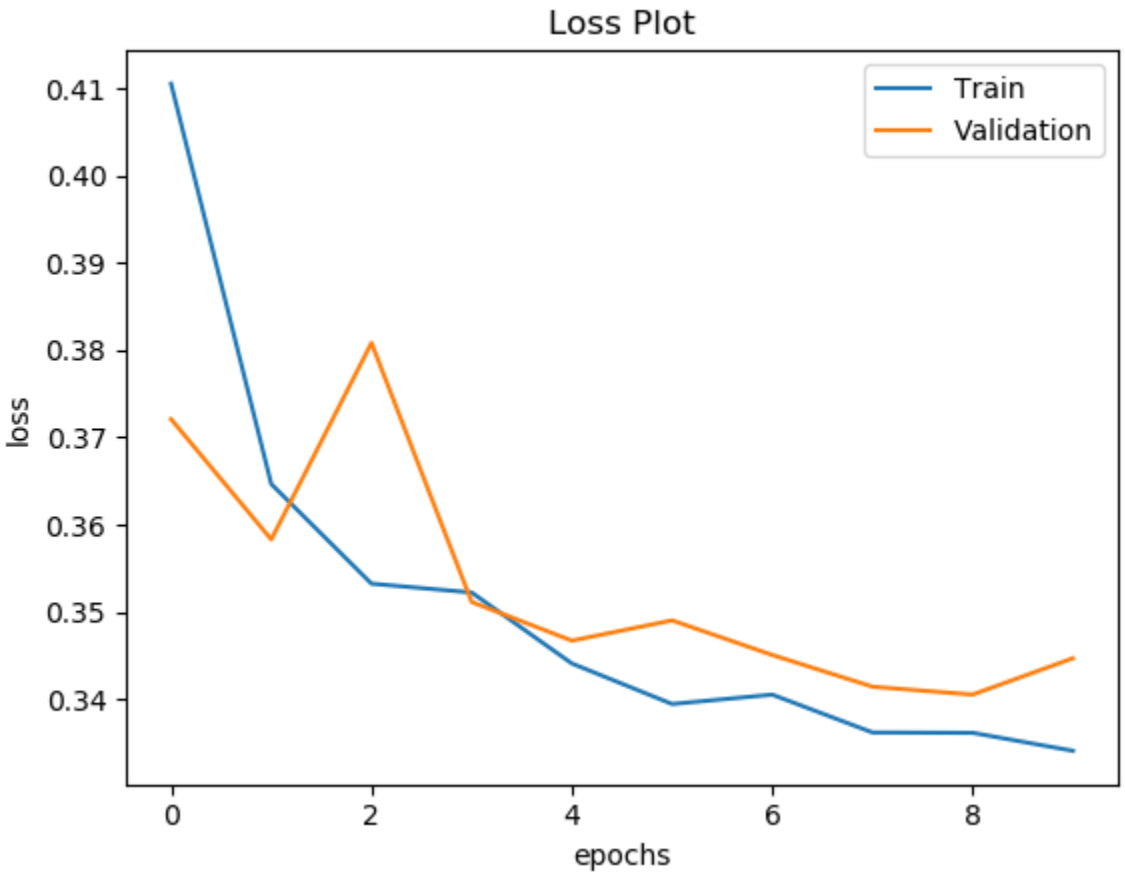
Finally, after choosing learning rate of 0.001 and with the following convolution layer:

```
ConvNet(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (fc1): Linear(in_features=12544, out_features=2048, bias=True)
  (fc2): Linear(in_features=2048, out_features=2, bias=True)
  (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (bn3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (bn4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (bn5): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
```

```python
class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        self.conv1 = nn.Conv2d(3,32,3,1, padding = 1)
        self.conv2 = nn.Conv2d(32,64,3,1, padding=1)
        self.conv3 = nn.Conv2d(64,128,3,1,padding=1)
        self.conv4 = nn.Conv2d(128,256,3,1,padding=1)
        self.fc1 = nn.Linear(7*7*256,4*4*128)
        self.fc2 = nn.Linear(4*4*128,2)
        self.bn1 = nn.BatchNorm2d(32)
        self.bn2 = nn.BatchNorm2d(64)
        self.bn3 = nn.BatchNorm2d(128)
        self.bn4 = nn.BatchNorm2d(256)
        self.bn5 = nn.BatchNorm1d(4*4*128)
    def forward(self, x):
        x = F.relu(self.bn1(self.conv1(x)))
        x = F.relu(self.bn2(self.conv2(x)))
        x = F.max_pool2d(x,2,2)
        x = F.relu(self.bn3(self.conv3(x)))
        x = F.max_pool2d(x,2,2)
        x = F.relu(self.bn4(self.conv4(x)))
        x = F.max_pool2d(x,2,2)
        x = x.view(-1,7*7*256)
        x = F.relu(self.bn5(self.fc1(x)))
        x = self.fc2(x)
        return F.softmax(x, dim=1)
```

Loss function used categorical cross-entropy.

Below are the results obtained while training this model.



```
(base) C:\Users\sport\Desktop\Project2>python main.py
Train Samples = 16000
Validation Samples = 4000
Train Epoch: 1 [15872/16000 (99%)]Loss: 0.376178;     Accuracy: 0.898
Validation set: Average loss: 0.3721, Accuracy: 3759/4000 (94%)
Train Epoch: 2 [15872/16000 (99%)]Loss: 0.340717;     Accuracy: 0.947
Validation set: Average loss: 0.3583, Accuracy: 3815/4000 (95%)
Train Epoch: 3 [15872/16000 (99%)]Loss: 0.377945;     Accuracy: 0.958
Validation set: Average loss: 0.3809, Accuracy: 3719/4000 (93%)
Train Epoch: 4 [15872/16000 (99%)]Loss: 0.365489;     Accuracy: 0.960
Validation set: Average loss: 0.3511, Accuracy: 3844/4000 (96%)
Train Epoch: 5 [15872/16000 (99%)]Loss: 0.363311;     Accuracy: 0.968
Validation set: Average loss: 0.3467, Accuracy: 3864/4000 (97%)
Train Epoch: 6 [15872/16000 (99%)]Loss: 0.347364;     Accuracy: 0.973
Validation set: Average loss: 0.3490, Accuracy: 3849/4000 (96%)
Train Epoch: 7 [15872/16000 (99%)]Loss: 0.346014;     Accuracy: 0.972
Validation set: Average loss: 0.3450, Accuracy: 3870/4000 (97%)
Train Epoch: 8 [15872/16000 (99%)]Loss: 0.326072;     Accuracy: 0.977
Validation set: Average loss: 0.3414, Accuracy: 3880/4000 (97%)
Train Epoch: 9 [15872/16000 (99%)]Loss: 0.345606;     Accuracy: 0.977
Validation set: Average loss: 0.3405, Accuracy: 3885/4000 (97%)
Train Epoch: 10 [15872/16000 (99%)]Loss: 0.337726;     Accuracy: 0.979
Validation set: Average loss: 0.3447, Accuracy: 3870/4000 (97%)
Test set: Average loss: 0.3427, Accuracy: 1939/2000 (97%)
```

Final accuracy obtained on the test dataset is 97%.