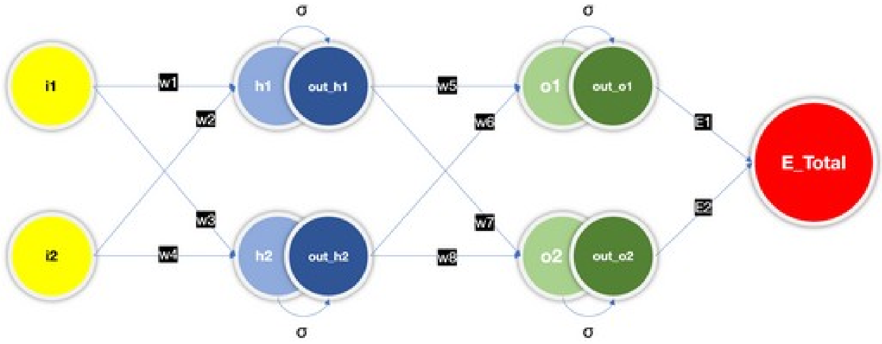
## PART 1

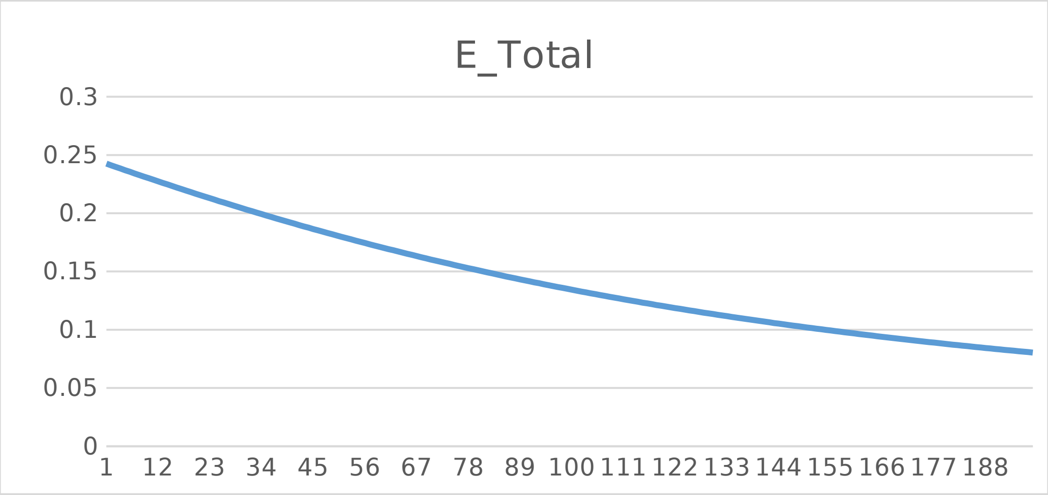
## Network:



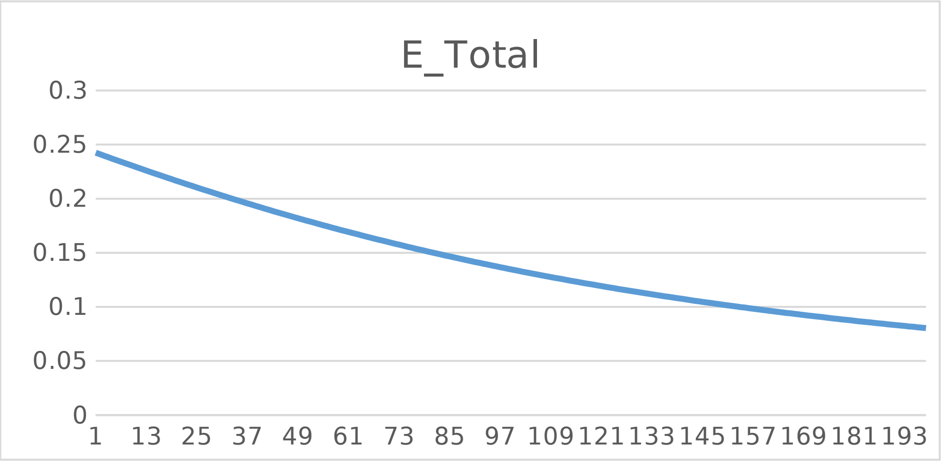
## Major Steps

- Forward pass – Wieights are evaluated to find predicted output.
- Backpropagation of error – Using chain rule, error contribution of each weight is calculated.
- Weight update – Based on previous weight value, learning rate and error contribution backpropagated, weights are updated
- Higher the learning rate – More aggressive the weight updates are

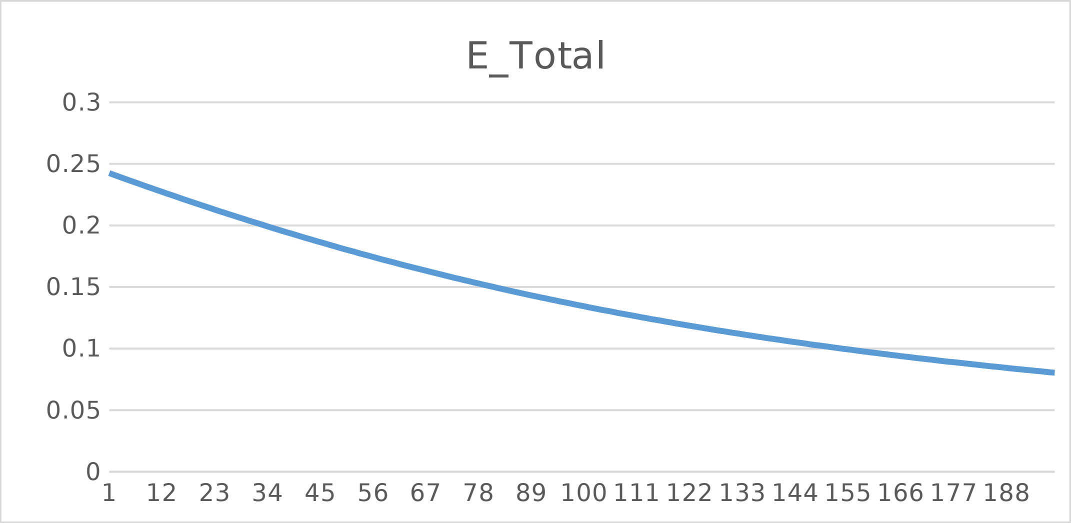
Lr = 0.1



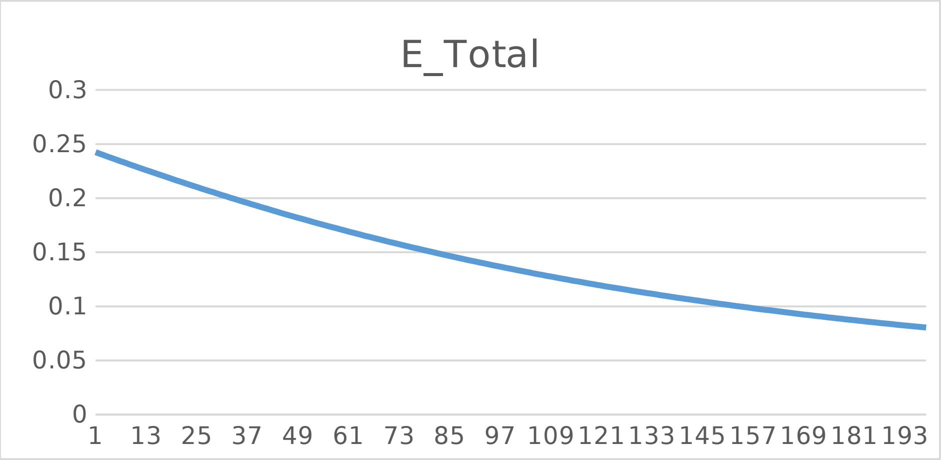
```
Lr = 0.2
```



```
lr = 0.8
```



```
lr = 1
```



## PART 2

The intention is that for MNIST achieve

99.4% validation accuracy : achieved 99%
Less than 20k Parameters : 25.3K
Less than 20 Epochs : OK

**Network**

Use of Conv layers (with and without padding),

use of BN (functional) : to make sure that the features available to the next layer is good, at - least 2 Conv layers away from output

use of max pooling : used twice, at - least 2 Conv layers away from output

use of GAP : instead of FC layers

use of 1x1 : to reduce the number of channels

```
self.conv1 = nn.Conv2d(1, 4, 3, padding=1) #input -1x28x28 Output - 4x28x28
# BN applied after this
self.conv2 = nn.Conv2d(4, 8, 3, padding=1) #input -4x28x28 Output - 8x28x28
# BN applied after this
self.pool1 = nn.MaxPool2d(2, 2) #input -8x28x28 Output - 8x14x14
# drop out applied
self.conv3 = nn.Conv2d(8, 16, 3, padding=1) #input -8x14x14 Output - 16x14x14
# BN applied after this
self.conv4 = nn.Conv2d(16, 32, 3, padding=1)#input -16x14x14 Output - 32x14x14
# BN applied after this
self.pool2 = nn.MaxPool2d(2, 2) #input -32x14x14 Output - 32x7x7
# dropout applied after this
self.conv5 = nn.Conv2d( 32, 64, 3) #input -32x7x7 Output - 64x5x5
# BN applied after this
self.conv6 = nn.Conv2d(64, 10, 1) #input -64x5x5 Output - 10x5x5
self.gap = nn.AvgPool2d(5) # input - 10x5x5 Output -10x1x1
```

**Parameters**

```
   Layer (type)               Output Shape         Param #

       Conv2d-1      |      [-1, 4, 28, 28]        |      40
       Conv2d-2      |       [-1, 8, 28, 28]       |         296
    MaxPool2d-3      |       [-1, 8, 14, 14]       |         0
       Conv2d-4      |       [-1, 16, 14, 14]      |       1,168
       Conv2d-5      |      [-1, 32, 14, 14]       |       4,640
    MaxPool2d-6      |         [-1, 32, 7, 7]      |           0
       Conv2d-7      |         [-1, 64, 5, 5]      |      18,496
       Conv2d-8      |         [-1, 10, 5, 5]      |        650
    AvgPool2d-9      |         [-1, 10, 1, 1]      |          0

   Total params: 25,290
   Trainable params: 25,290
   Non-trainable params: 0
```

**Logs**

```
--------epoch-1------- 0%| | 0/469 [00:00<?,
?it/s]/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.p
y:61: UserWarning: Implicit dimension choice for log_softmax has
been deprecated. Change the call to include dim=X as an argument.
loss=0.3111923336982727 batch_id=468: 100%|| 469/469
[00:22<00:00, 20.63it/s]

Test set: Average loss: 0.2894, Accuracy: 9316/10000 (93%)

--------epoch-2------- loss=0.12034612149000168 batch_id=468:
100%|| 469/469 [00:22<00:00, 21.10it/s]

Test set: Average loss: 0.1545, Accuracy: 9609/10000 (96%)

--------epoch-3------- loss=0.1218857541680336 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.98it/s]

Test set: Average loss: 0.1199, Accuracy: 9693/10000 (97%)

--------epoch-4------- loss=0.25661221146583557 batch_id=468:
100%|| 469/469 [00:22<00:00, 21.07it/s]

Test set: Average loss: 0.1084, Accuracy: 9697/10000 (97%)

--------epoch-5------- loss=0.12994997203350067 batch_id=468:
```

```
100%|| 469/469 [00:22<00:00, 21.00it/s]

Test set: Average loss: 0.0861, Accuracy: 9784/10000 (98%)

--------epoch-6------- loss=0.14073851704597473 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.95it/s]

Test set: Average loss: 0.0827, Accuracy: 9769/10000 (98%)

--------epoch-7------- loss=0.06521926820278168 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.86it/s]

Test set: Average loss: 0.0759, Accuracy: 9789/10000 (98%)

--------epoch-8------- loss=0.09442844986915588 batch_id=468:
100%|| 469/469 [00:22<00:00, 21.10it/s]

Test set: Average loss: 0.0725, Accuracy: 9786/10000 (98%)

--------epoch-9------- loss=0.07865099608898163 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.87it/s]

Test set: Average loss: 0.0663, Accuracy: 9824/10000 (98%)

--------epoch-10------- loss=0.035089436918497086 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.79it/s]

Test set: Average loss: 0.0648, Accuracy: 9819/10000 (98%)

--------epoch-11------- loss=0.10404521226882935 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.96it/s]

Test set: Average loss: 0.0602, Accuracy: 9835/10000 (98%)

--------epoch-12------- loss=0.06454789638519287 batch_id=468:
100%|| 469/469 [00:22<00:00, 21.02it/s]

Test set: Average loss: 0.0586, Accuracy: 9826/10000 (98%)

--------epoch-13------- loss=0.05638539418578148 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.87it/s]

Test set: Average loss: 0.0588, Accuracy: 9829/10000 (98%)

--------epoch-14------- loss=0.03229639306664467 batch_id=468:
```

```
100%|| 469/469 [00:22<00:00, 20.78it/s]

Test set: Average loss: 0.0535, Accuracy: 9855/10000 (99%)

--------epoch-15------- loss=0.08589702099561691 batch_id=468:
100%|| 469/469 [00:22<00:00, 21.01it/s]

Test set: Average loss: 0.0531, Accuracy: 9843/10000 (98%)

--------epoch-16------- loss=0.01948048174381256 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.84it/s]

Test set: Average loss: 0.0505, Accuracy: 9861/10000 (99%)

--------epoch-17------- loss=0.03663216158747673 batch_id=468:
100%|| 469/469 [00:22<00:00, 21.16it/s]

Test set: Average loss: 0.0569, Accuracy: 9833/10000 (98%)

--------epoch-18------- loss=0.03001176007091999 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.72it/s]

Test set: Average loss: 0.0519, Accuracy: 9851/10000 (99%)

--------epoch-19------- loss=0.023839695379137993 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.76it/s]

Test set: Average loss: 0.0478, Accuracy: 9864/10000 (99%)

--------epoch-20------- loss=0.06415826082229614 batch_id=468:
100%|| 469/469 [00:22<00:00, 20.76it/s]

Test set: Average loss: 0.0513, Accuracy: 9856/10000 (99%)
```