HOMEWORK ASSIGNMENT-2
Neural Networks & CNN

1.Backpropagation

1.1 Algorithm for training Neural Networks was implemented

1.2 Activation functions with their gradi-ent calculation too were also implemented in the network (ReLU, Sigmoid, Linear, Tanh, Softmax.)

1.3 Weight initialization techniques for the hidden layers was also implemented (Zero, Random, Normal)

1.4 Several functions with zero bias for each neuron and cross-entropy loss as the loss function was implemented.

1.5 FASHION MNIST dataset was used for training and testing the neural network model `MyNeuralNetwork.`

Following architecture was used to train the model-

Architecture: #input, 256, 128, 64, #output

•Learning Rate: 0.1

• Number of Epochs: 100

• Batch Size: 32

• Using normal weight initialization

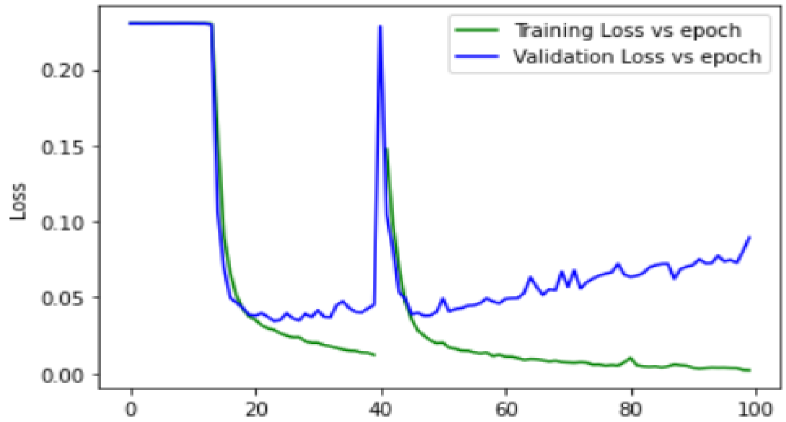
• Training using stochastic gradient descent and random shuffle on each iteration of SGD

• Dataset was split into 80% train, 10% test, and 10% Validation.

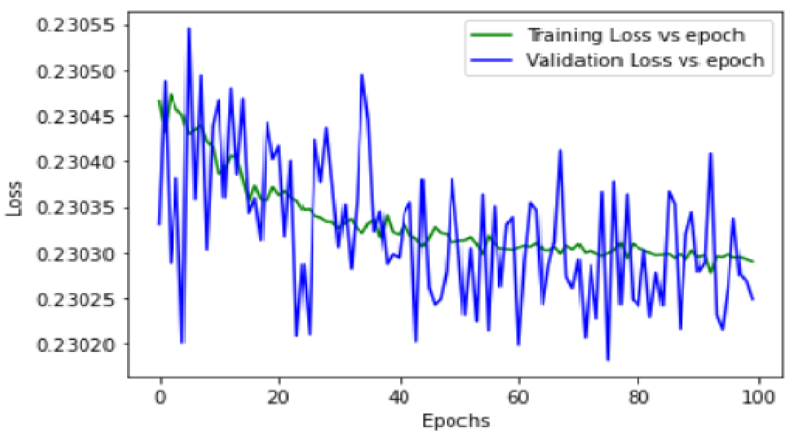
Weights and Biases were saved for 50 and 100 iteration using pickle with following file names-

-100itr_RelU_Wights_Bias.sav
-100itr_Sigmoid_Wights_Bias.sav
-100itr_Linear_Wights_Bias.sav
-100itr_TanH_Wights_Bias.sav
-50itr_ReLU_Wights_Bias.sav
-50itr_Sigmoid_Wights_Bias.sav
-50itr_Linear_Wights_Bias.sav
-50itr_Tanh_Wights_Bias.sav

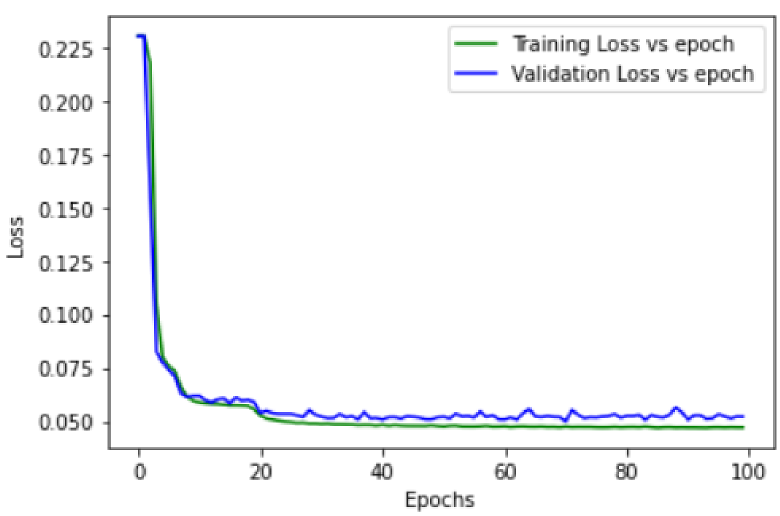
Training and Validation loss vs epoch for ReLU activation function after 100 iterations



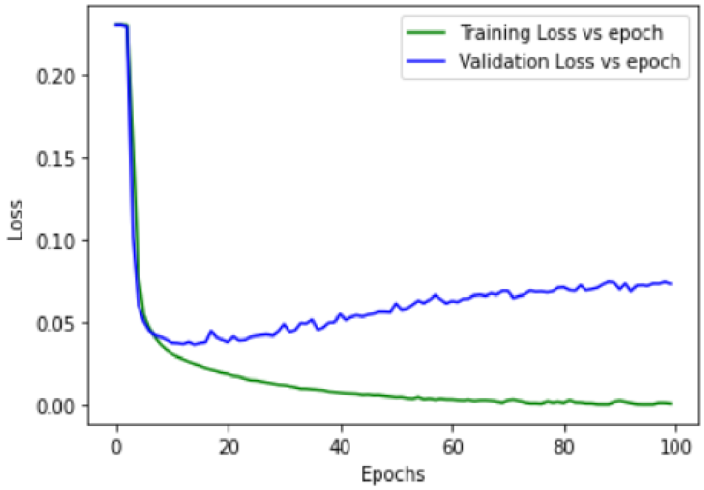
Training and Validation loss vs epoch for Sigmoid activation function after 100 iterations



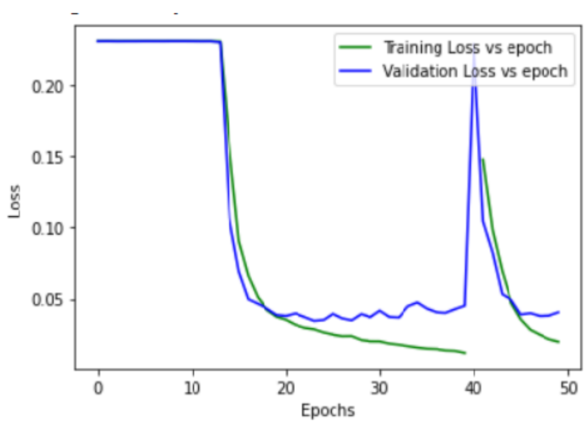
Training and Validation loss vs epoch for Linear activation function after 100 iterations



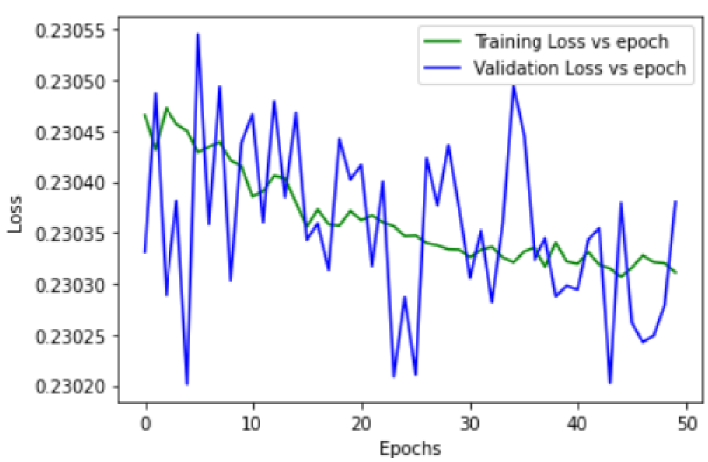
Training and Validation loss vs epoch for Tanh activation function after 100 iterations



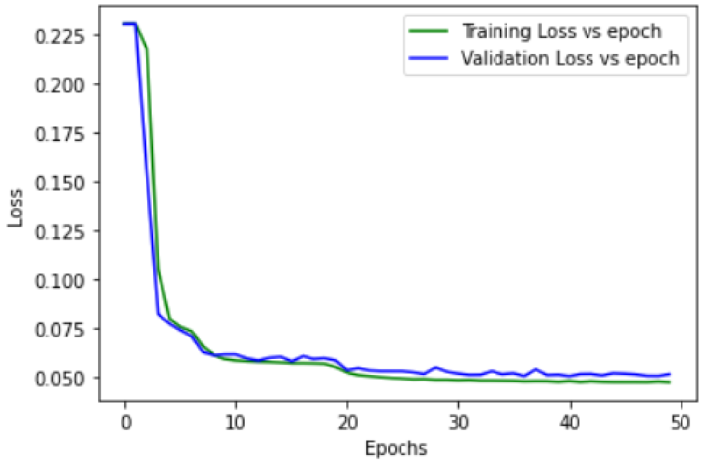
Training and Validation loss vs epoch for ReLU activation function after 50 iterations



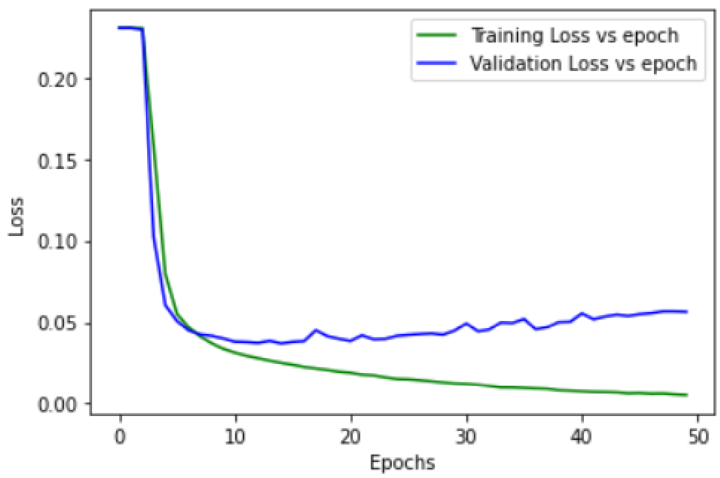
Training and Validation loss vs epoch for Sigmoid activation function after 50 iterations



Training and Validation loss vs epoch for Linear activation function after 50 iterations



Training and Validation loss vs epoch for Linear activation function after 50 iterations



Testing saved model on test set and comparing accuracies on 50 and 100 iterations

1. ReLU with 100 iterations

```
Training Accuracy: 0.9870164609053498
Testing Accuracy: 0.8865
```

2. ReLU with 50 iterations

```
Training Accuracy: 0.935164609053498
Testing Accuracy: 0.8833333333333333
```

3. Sigmoid with 100 iterations

```
Sigmoid activation function
Training Accuracy: 0.09958847736625515
Testing Accuracy: 0.10116666666666667
```

4. Sigmoid with 50 iterations

```
Training Accuracy: 0.09983539094650205
Testing Accuracy: 0.1085
```

5. Linear with 100 iterations

```
Training Accuracy: 0.8377777777777777
Testing Accuracy: 0.8171666666666667
```

6. Linear with 50 iterations

```
Training Accuracy: 0.8383333333333334
Testing Accuracy: 0.8218333333333333
```

7. Tanh with 100 iterations

```
Tanh activation function
Training Accuracy: 0.9985802469135803
Testing Accuracy: 0.8821666666666667
```

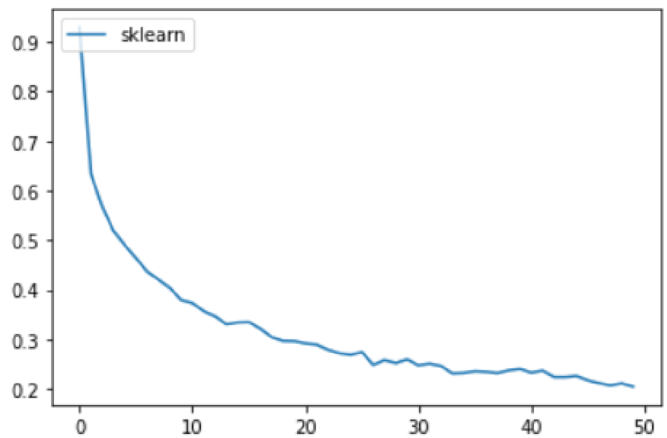
8. Tanh with 50 iterations

```
ReLU activation function
Training Accuracy: 0.9863168724279835
Testing Accuracy: 0.881
```

6.MLP Classifier

Tanh with 50 iterations

Loss vs epochs

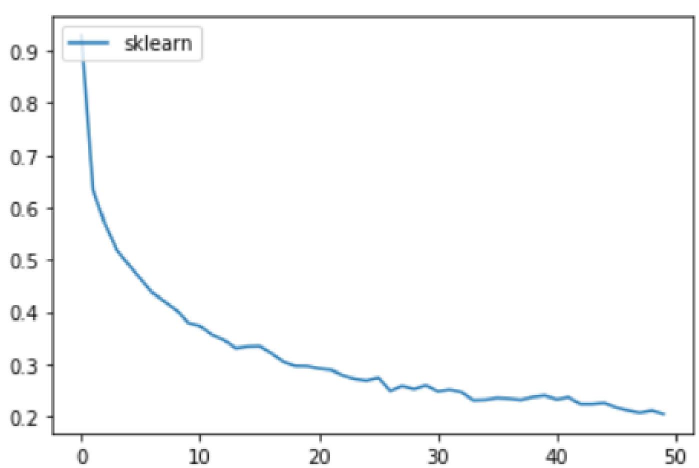


Testing accuracy

0.8731666666666666

Linear with 50 iterations

Loss vs epochs

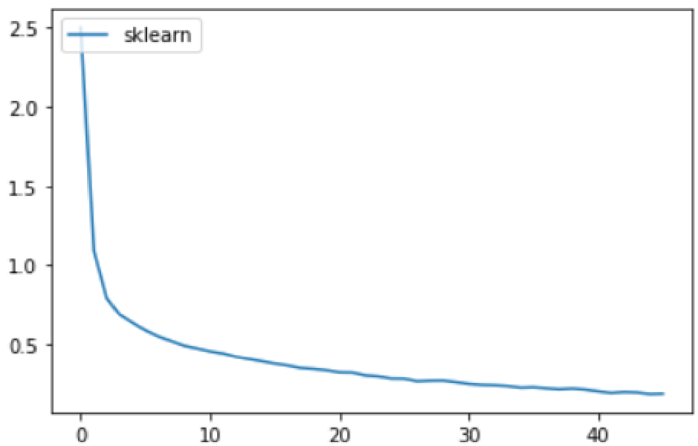


Testing accuracy

```
0.0
```

ReLU with 50 iterations

Loss vs epoch

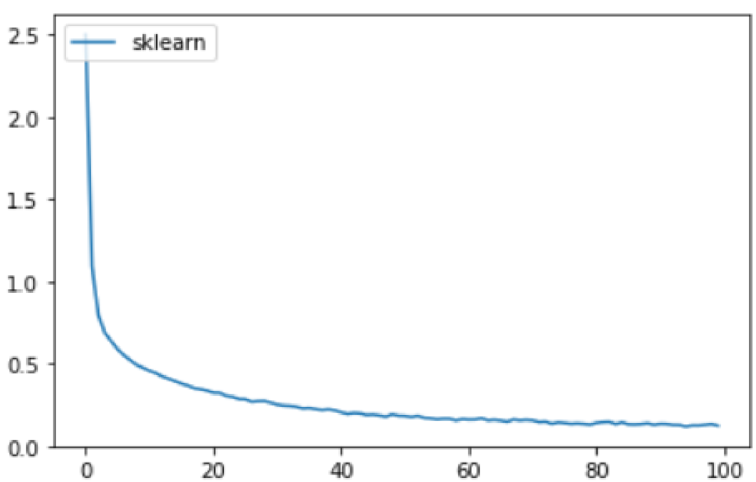


Test Accuracy

```
0.0
```

Sigmoid with 100 iterations

Loss vs epoch

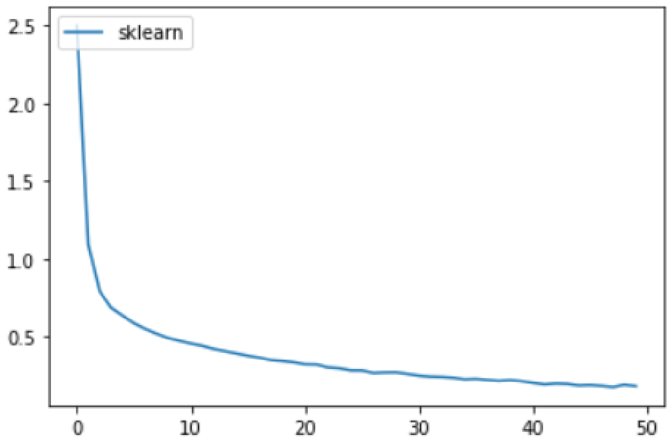


Test accuracy

0.874

Sigmoid with 50 iterations

Loss vs Epochs

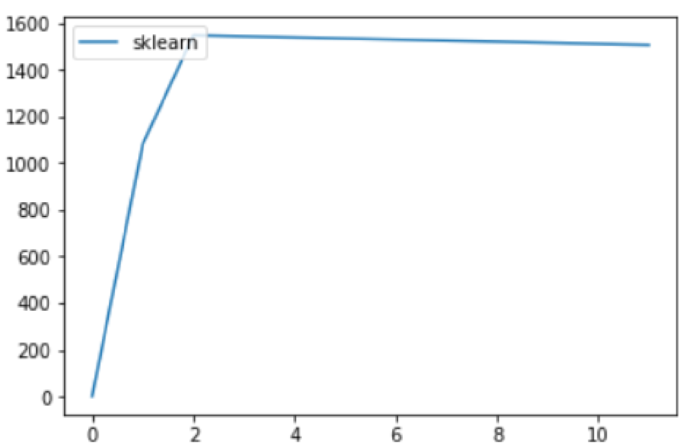


Test accuracy

0.878

ReLU with 50 iterations

Loss vs Epochs

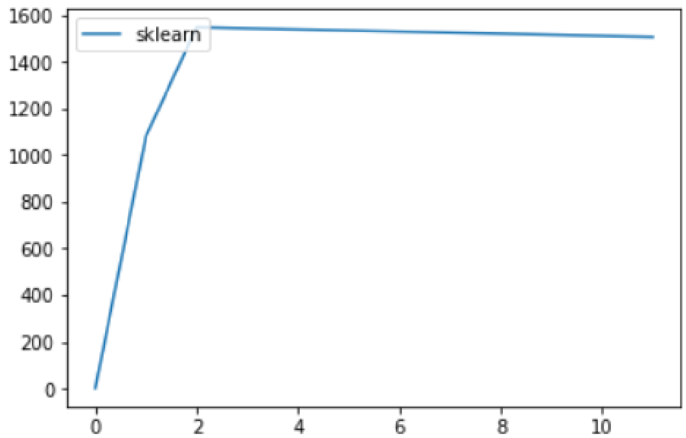


Test accuracy

0.0

Linear with 100 iterations

Loss vs Epochs

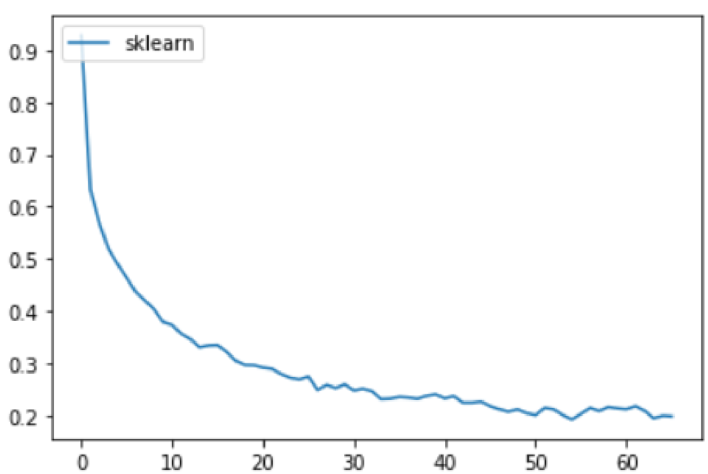


Test accuracy

```
0.0
```

Tanh with 100 iterations

Loss vs Epochs



Test accuracy

```
0.8743333333333333
```

The best model in tems of testing accuracy is model with Sigmoid activation function and having 50 iteration

Q2. Convolutional Neural Networks

VGG16 model was used in FASHION MNIST Dataset, dataset was divided into 80% train and 20% test, the weights of lower convolutiinal layer was freezed. The Upper dense layers of networks were replaced and the model was trained using custom added layers.

Class wise accuracy.

```
 95%|█████████▍| 178/188 [05:09<00:17,  1.73s/it]1.8336505889892578
Eval tensor(0.3125, device='cuda:0')


 95%|█████████▌| 179/188 [05:10<00:15,  1.75s/it]1.8374768495559692
Eval tensor(0.3281, device='cuda:0')


 96%|█████████▌| 180/188 [05:12<00:14,  1.75s/it]1.5610023736953735
Eval tensor(0.4844, device='cuda:0')


 96%|█████████▋| 181/188 [05:14<00:12,  1.73s/it]1.2519526481628418
Eval tensor(0.5938, device='cuda:0')


 97%|█████████▋| 182/188 [05:16<00:10,  1.72s/it]0.9498541951179504
Eval tensor(0.5938, device='cuda:0')


 97%|█████████▋| 183/188 [05:17<00:08,  1.74s/it]1.0633518695831299
Eval tensor(0.6094, device='cuda:0')
```

THEORY QUESTIONS

1. The KL divergence between two discrete probability distributions P and Q is given by $KL(P|Q) = \sum_z P(z) \log \frac{P(z)}{Q(z)}$.
Show that the cross-entropy loss used for multi-class classification is the same as the KL divergence under certain assumptions of the posterior distribution $P(y|x)$, where y are the class labels, while x are the data samples. What are these assumptions?

→ KL divergence and Cross entropy are used to measure the distance between two probability distribution

KL divergence between two discrete probability distribution P and Q is given by

$$KL(P|Q) = \sum_z P(z) \log \frac{P(z)}{Q(z)}$$

And Cross-entropy is given by

$$H(P,Q) = -\sum_z P(z) \log Q(z)$$

Cross-entropy loss used for multi-class classification is same as the KL divergence

Entropy of a distribution is given by

$$S(a) = -\sum_i p(a_i) \log P(a_i)$$

$p(a_i)$ is the probability of different states of the system (a_i). And $S(a)$ is the ~~an~~ quantity of information for removing the Uncertainty.

Under the assumption that this entropy is constant.

~~A~~ According to KL Divergence

$$D_{KL}(P|Q) = \sum_i P(a_i) \log P(a_i) - P_P(a_i) \log P_Q(a_i)$$

Right side is the entropy distribution of ~~P~~ P and left side is the entropy distribution of Q

for relating cross-entropy to entropy and KL divergence, and formalizing cross entropy in terms of distribution of P and Q as

$$H(P,Q) = -\sum_i P_A(a_i) \log P_Q(a_i)$$

$$\therefore H(P,Q) = D_{KL}(P|Q) + S_A.$$

If S_A in constant, then ~~minimizing~~ reducing or minimizing $H(P,Q)$ is equivalent to minimize $D_{KL}(P||Q)$.

Therefore, <u>entropy is to be constant</u> for minimizing cross entropy and KL divergence which is same under this particular assumption.

2. Network 1 :

- Input : Feature map of size 28 X 28 X 192
- Convolution operation : Filter size 5X5, same padding, 32 filters.

Network 2 :

- Input : Feature map of size 28 X 28 X 192
- Convolution operation : Filter size 1 X 1, same padding, 16 filters.
- Convolution operation : Filter size 5X5, same padding 32 filters.

→ In Network 1.

Input : Feature map 28 X 28 X 192

Padding = 32 filter

filter size : 5 X 5

5 X 5 X 32

Total Parameter = (5 X 5 X 192) + 1) X 32

= 55,328

- Output feature map.

$$= \frac{28 - 5 + 2}{1} + 1$$

Output feature map size

$26 \times 26 \times 32$.

→ In Network 2

$28 \times 28 \times 19$ feature map size

$1 \times 1 \times 16$ → Convolution operation

$$\frac{28 - 1 + 2}{1} + 1 = 30 \times 30 \times 16$$

$\downarrow$

$5 \times 5 \times 32$

Total Parameter

$= 3088 = ((1 \times 1 \times 192) + 1 \times 16)$ Convolution operation

$$= \frac{(30 - 5 + 2) + 1}{1} + 1 = 28 \times 28 \times 32$$

Output feature Map size $= 28 \times 28 \times 32$

Total Params in output layer $= ((5 \times 5 \times 16)+1) \times 32$
$= 12,832$

Overall parameter in Network 2 $= 3088 + 12,832$
$= 15,920$

So, comparing Network 1 with 2.

Network 1 performs more Number of computation than 2.

Case 1 >> Case 2
(55,328) (15,920)