Name: Vikrant Tripathi

CS331

Lab 2: Deep Learning

Problem 1

The solution to this problem implements the following classes

- Neural Network class
- Layer class
- Activation class
- Loss class

1. Neural Network class

Objects in this class and their explanations are

- listLayers: List to store layers
- lossObj: Loss function to be called at end
- target: Output Y values
- input: Input X values
- epochs: Number of epochs

Methods in this class and their explanations are

- forwardpassNN: Iterates over all layers in the network and calls forward pass of each layer to get final output.
- train: Trains the neural network using input and target, by running back propagation and steepest descent methods stored in layers along with the forward pass layer implemented.
- test: Tests the neural network by using the implemented forward pass method and comparing it with targets for test data set.

2. Layer class

Objects in this class and their explanations are

- **numNeurons**: Number of neurons in this layer.
- **numNeuronsPrev**: Number of neurons in the previous layers.
- **W**: Weight matrix associated with this layer.
- **B**: Bias matrix associated with this layer.
- **activation**: Activation class object associated with this layer.
- **alpha**: Step size for steepest descent method.
- **input**: Input to this layer.
- **output**: Output of this layer.

Methods in this class and their explanations are

- **update**: Takes inputs gradients of W and B and updates according to the steepest descent method.
- **forwardPasslinear**: Evaluates the value from layer by calculating $WX + B$.
- **forwardPass**: Calculates the activation of the value received from the layer.
- **backwardPasslinear**: Implements backward propagation for linear step using chain rule $\frac{\partial L}{\partial X} = W^T \frac{\partial L}{\partial O}$ and $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial O} X^T$ where O is the output.
- **backwardPass**: Implements the backwardPass associated with the activation function by calling the method inside Activation layer.

3. Activation class

Objects in this class and their explanations are

- **act**: Activation stored as a string.

Methods in this class and their explanations are

- **backwardPass**: Takes inputs X and $\frac{\partial L}{\partial O}$ and calls the backward pass method associated with the current activation on it.
- **forwardPass**: Evaluates the activation value from layer by calculating calling the activation method associated with the current activation on it.
- **forwardPassTanH**: Takes input number x, returns $\frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$
- **forwardPassLinear**: Takes input number x, returns x itself.
- **forwardPassSigmoid**: Takes input number x, returns $\frac{1}{1+\exp(-x)}$ itself.
- **forwardPassSoftmax**: Takes input vector X, returns $\frac{\exp(x)}{\sum_{i=1}^{n} \exp(x_i)}$

- **backwardPassTanH**: Takes input number x and $\frac{\partial L}{\partial O}$, returns $\frac{\partial L}{\partial O}(1-\tanh^2(x))$
- **backwardPassLinear**: Takes input number x and $\frac{\partial L}{\partial O}$, returns $\frac{\partial L}{\partial O}$
- **backwardPassSigmoid**: Takes input number x and $\frac{\partial L}{\partial O}$, returns $\frac{\partial L}{\partial O}(\sigma(x))(1-\sigma(x))$
- **backwardPassSoftmax**: Combines the backward pass of Softmax and Cross entropy and returns $Q-Y$ where Q is the output of Softmax forward pass on the input received and Y is the actual labels.

4. Loss class

Objects in this class and their explanations are

- **losstype**: Type of loss stored as a string.

Methods in this class and their explanations are

- **backwardPass**: Takes inputs X and Y and calls the backward pass method associated with the current activation on it.
- **forwardPass**: Evaluates the activation value from layer by calculating calling the activation method associated with the current activation on it.
- **forwardPassTanH**: Takes input number x, returns $\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$
- **forwardPassLinear**: Takes input number x, returns x itself.
- **forwardPassSigmoid**: Takes input number x, returns $\frac{1}{1+\exp(-x)}$ itself.
- **forwardPassSoftmax**: Takes input vector X, returns $\frac{\exp(X)}{\sum_{i=1}^{n}\exp(X_i)}$
- **backwardPassTanH**: Takes input number x and $\frac{\partial L}{\partial O}$, returns $\frac{\partial L}{\partial O}(1-\tanh^2(x))$
- **backwardPassLinear**: Takes input number x and $\frac{\partial L}{\partial O}$, returns $\frac{\partial L}{\partial O}$
- **backwardPassSigmoid**: Takes input number x and $\frac{\partial L}{\partial O}$, returns $\frac{\partial L}{\partial O}(\sigma(x))(1-\sigma(x))$
- **backwardPassSoftmax**: Combines the backward pass of Softmax and Cross entropy and returns $Q-Y$ where Q is the output of Softmax forward pass on the input received and Y is the actual labels.

Problem 2

We have implemented the following networks as required

1. Network 1

Description: Just one output neural with linear activation and least mean square loss. After running this network, we got the output and comparison graph as shown below:

```
LOSS OUTPUT FOR Q2(a)
Loss is at epoch 100 = 53.22240946503577
Loss is at epoch 200 = 68.52270117667345
Loss is at epoch 300 = 8.702315956394095
Loss is at epoch 400 = 2.1240411735952973e-05
Loss is at epoch 500 = 6.662390295997569
Loss is at epoch 600 = 23.03857269342182
Loss is at epoch 700 = 64.33250928645933
Loss is at epoch 800 = 34.50975470863134
Loss is at epoch 900 = 27.629213844678794
Loss is at epoch 1000 = 48.995462253380126
Loss is at epoch 1100 = 0.1398983657848332
Loss is at epoch 1200 = 5.135328944574756
Loss is at epoch 1300 = 23.863224876416993
Loss is at epoch 1400 = 141.17401532079367
Loss is at epoch 1500 = 3.336425137254571
Loss is at epoch 1600 = 29.27405498769253
Loss is at epoch 1700 = 43.28755576798777
Loss is at epoch 1800 = 17.251767708733723
Loss is at epoch 1900 = 46.43420047780213
Loss is at epoch 2000 = 1.2881792580899107
Loss is at epoch 2100 = 91.39559242600527
Loss is at epoch 2200 = 2.6424032699963376
Loss is at epoch 2300 = 41.89490298663059
Loss is at epoch 2400 = 64.32612805781584
...
Loss is at epoch 9900 = 87.25713835155703
Loss is at epoch 10000 = 286.98328367992326
Mean Squared Loss Error (Train Data)  : 61.59703
Mean Squared Loss Error (Test Data)  : 60.44135
```

Figure 1: Output

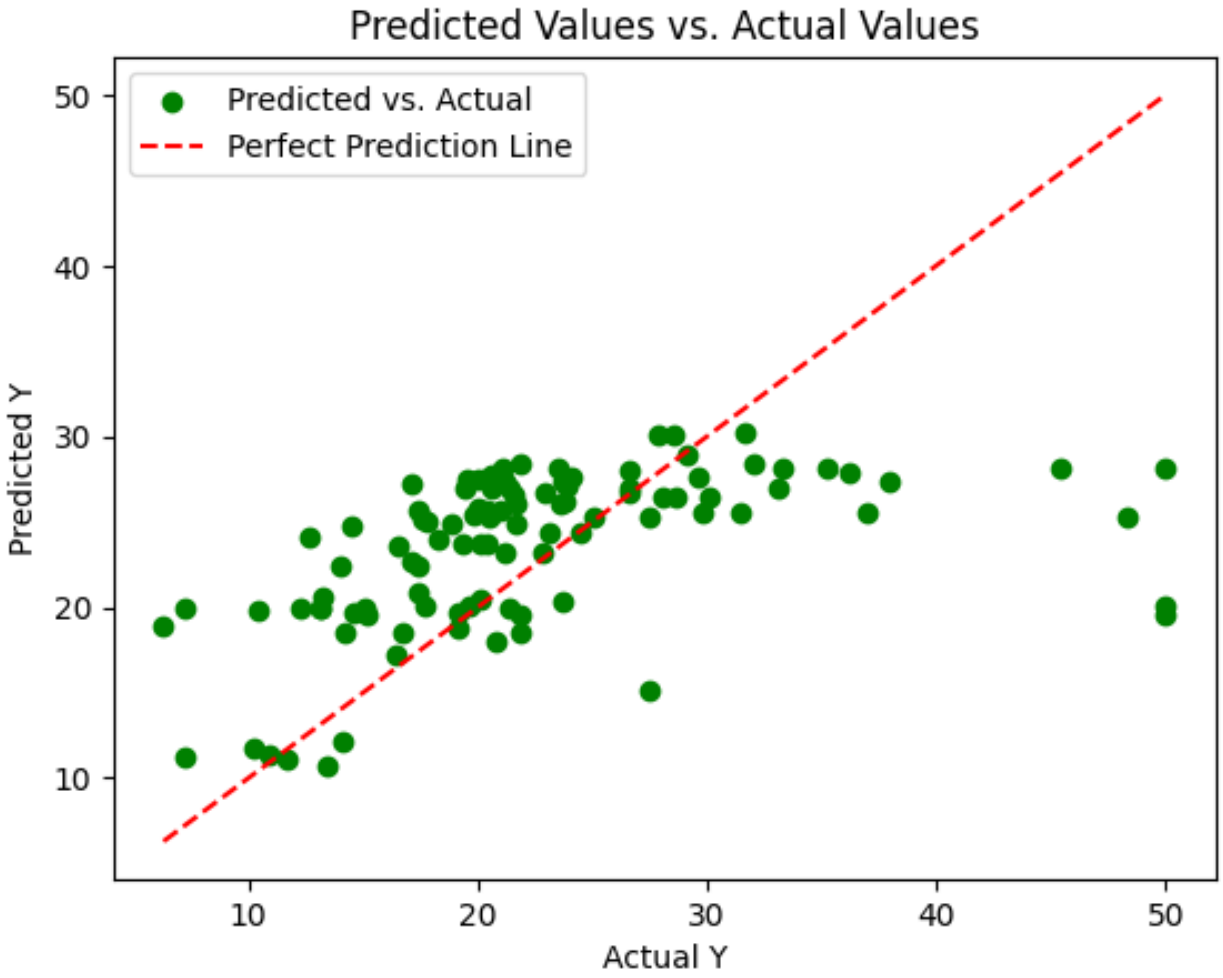

Figure 2: Comparison graph

2. Network 2

Description: Layer 1 with 13 output neurons with sigmoid activation. Layer 2 with one output neuron and linear activation. use mean squared loss.
Since, this network does not correspond to regression, we get non-sense output however we are able to run this network due to our dynamic code.
After running this network, we got the output and comparison graph as shown below:

```
LOSS OUTPUT FOR Q2(b)
Loss is at epoch 100 = 167.7117917536676
Loss is at epoch 200 = 80.18931334109298
Loss is at epoch 300 = 389.7588704451546
Loss is at epoch 400 = 84.3926807959338
Loss is at epoch 500 = 54.158316878022255
Loss is at epoch 600 = 20.563974741700186
Loss is at epoch 700 = 1.059585369985089
Loss is at epoch 800 = 0.27883027004433786
Loss is at epoch 900 = 11.578067451911684
Loss is at epoch 1000 = 125.02470990306436
Loss is at epoch 1100 = 4.414318975629955
Loss is at epoch 1200 = 4.464499966181485
Loss is at epoch 1300 = 150.84145389764487
Loss is at epoch 1400 = 37.85198081172116
Loss is at epoch 1500 = 31.556587117730235
Loss is at epoch 1600 = 19.281808271185
Loss is at epoch 1700 = 1.4196392291317055
Loss is at epoch 1800 = 1198.9678455892076
Loss is at epoch 1900 = 46.964348598629016
Loss is at epoch 2000 = 42.0198898998917
Loss is at epoch 2100 = 7.477779232572035
Loss is at epoch 2200 = 89.66947076965384
Loss is at epoch 2300 = 19.27688772780566
Loss is at epoch 2400 = 143.40507378232175
...
Loss is at epoch 9900 = 72.68485552969784
Loss is at epoch 10000 = 0.1925636066310946
Mean Squared Loss Error (Train Data)  : 67.54332
Mean Squared Loss Error (Test Data)  : 62.57934
```

Figure 3: Output

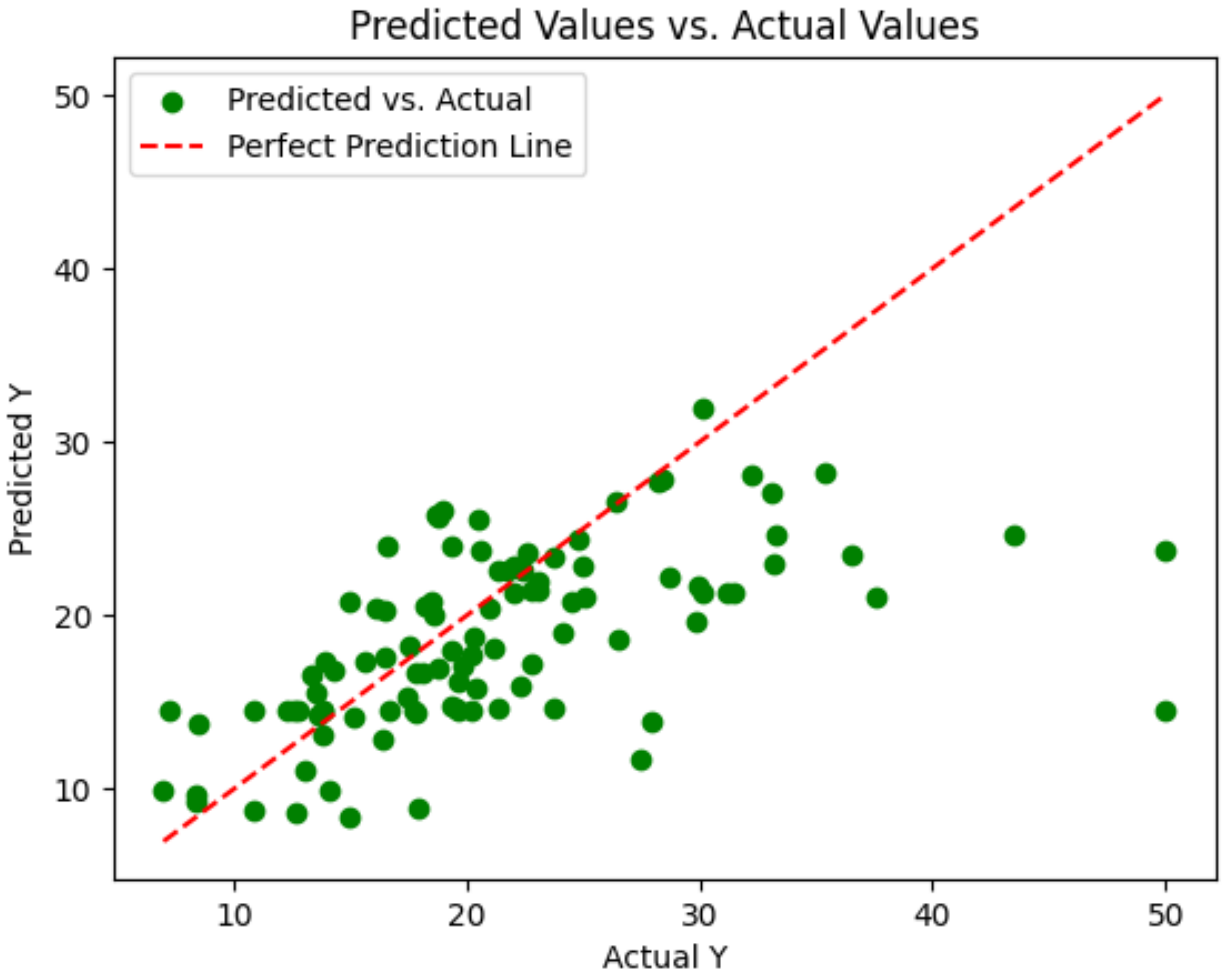

Figure 4: Comparison graph

3. Network 3

Description: Layer 1 with 13 output neurons with sigmoid activation. Layer 2 with 13 output neurons and sigmoid activation. Layer 3 with one output neuron and linear activation. use mean squared loss
Since, this network does not correspond to regression, we get non-sense output however we are able to run this network due to our dynamic code.
After running this network, we got the output and comparison graph as shown below:

```
LOSS OUTPUT FOR Q2(c)
Loss is at epoch 100 = 2.064721620552268
Loss is at epoch 200 = 45.66100044431093
Loss is at epoch 300 = 17.442872345808325
Loss is at epoch 400 = 75.73210652650958
Loss is at epoch 500 = 10.266445430364632
Loss is at epoch 600 = 0.006933207877279072
Loss is at epoch 700 = 145.80639279581868
Loss is at epoch 800 = 19.305562515023485
Loss is at epoch 900 = 80.5840734603834
Loss is at epoch 1000 = 42.42609781242966
Loss is at epoch 1100 = 0.037050600677712826
Loss is at epoch 1200 = 0.12122961100591724
Loss is at epoch 1300 = 42.628910497238856
Loss is at epoch 1400 = 194.73220967380686
Loss is at epoch 1500 = 96.99579153384089
Loss is at epoch 1600 = 0.011106823785863154
Loss is at epoch 1700 = 140.5133350813648
Loss is at epoch 1800 = 25.08961484103045
Loss is at epoch 1900 = 106.99157587516872
Loss is at epoch 2000 = 0.12359948859990283
Loss is at epoch 2100 = 427.75516124908194
Loss is at epoch 2200 = 8.335413333935808
Loss is at epoch 2300 = 7.140335888372658
Loss is at epoch 2400 = 121.36671823434453
...
Loss is at epoch 9900 = 165.46684567847134
Loss is at epoch 10000 = 122.41642709116576
Mean Squared Loss Error (Train Data)  : 129.57651
Mean Squared Loss Error (Test Data)  : 128.65742
```

Figure 5: Output

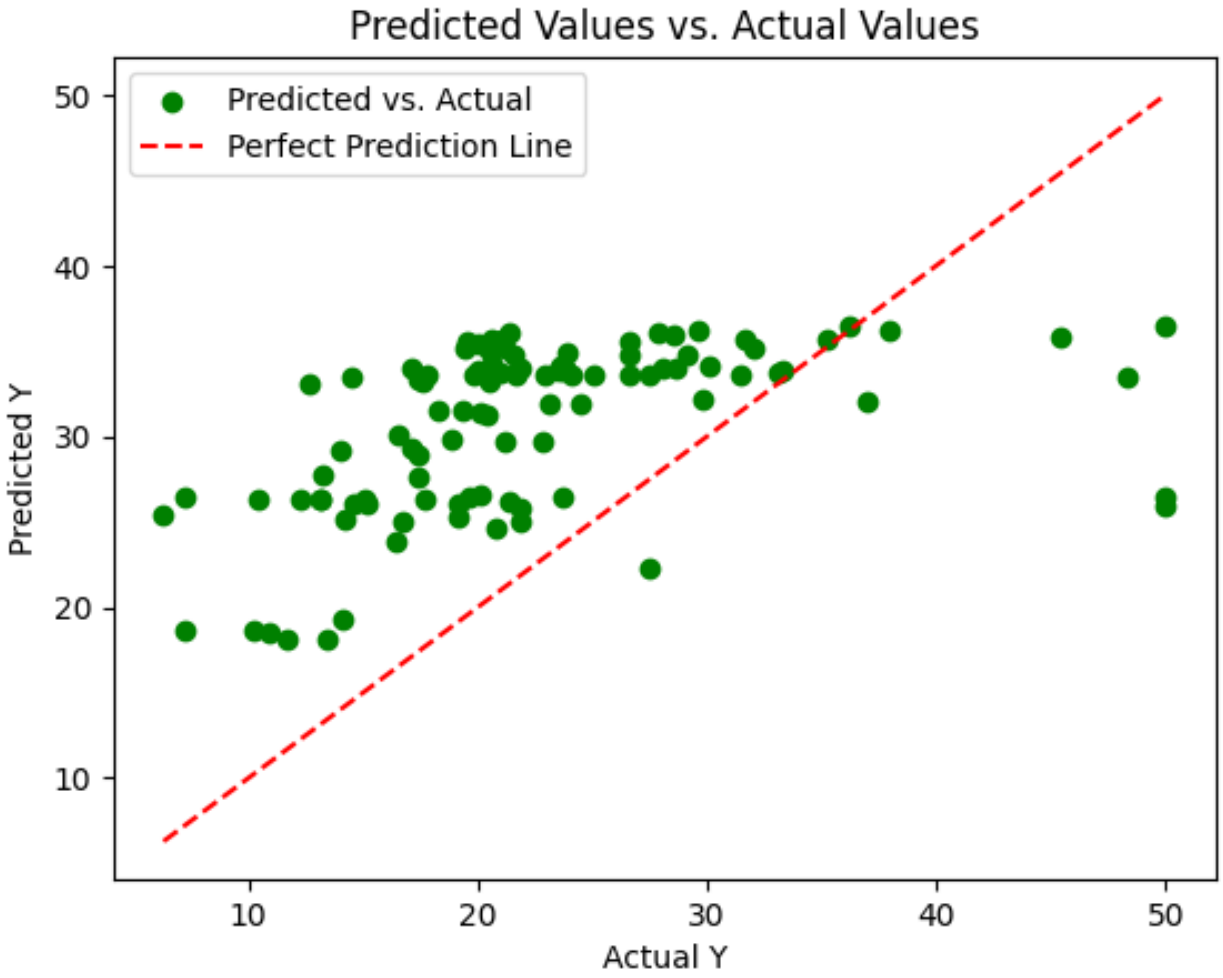

Figure 6: Comparison graph

Problem 3

We have implemented the following networks as required

1. Network 1

Description: Just one output neural with linear activation and least mean square loss. After running this network, we got the output and comparison graph as shown below:

```
LOSS OUTPUT FOR Q3(a)
Loss is at epoch 100 = 0.8056838867376064
Loss is at epoch 200 = 0.2712739953518245
Loss is at epoch 300 = 0.1153018029765729 87
Loss is at epoch 400 = 0.09382083427727414
Loss is at epoch 500 = 0.09144588937509941
Loss is at epoch 600 = 0.09179409484440275
Loss is at epoch 700 = 0.09237391047065714
Loss is at epoch 800 = 0.09176137117312803
Loss is at epoch 900 = 0.0918965042571025
Loss is at epoch 1000 = 0.08693824155879866
Loss is at epoch 1100 = 0.08611171760697858
Loss is at epoch 1200 = 0.08952672222406814
Loss is at epoch 1300 = 0.09459202080824239
Loss is at epoch 1400 = 0.09459456823003999
Loss is at epoch 1500 = 0.08762456543859112
Loss is at epoch 1600 = 0.08762579936629673
Loss is at epoch 1700 = 0.09008256929683485
Loss is at epoch 1800 = 0.08795524332473201
Loss is at epoch 1900 = 0.09325424486816039
Loss is at epoch 2000 = 0.0907797737031035
Loss is at epoch 2100 = 0.08757799481161192
Loss is at epoch 2200 = 0.09304283412483727
Loss is at epoch 2300 = 0.08724383887348323
Loss is at epoch 2400 = 0.08978765137925832
...
Loss is at epoch 9900 = 0.022720212485805014
Loss is at epoch 10000 = 0.006200274370297936
Classification Accuracy (Training Data ): 1194/1437 = 83.08977035490605 %
Classification Accuracy (Testing Data ): 293/360 = 81.38888888888889 %
```

Figure 7: Output

2. Network 2

Description: Layer 1 with 13 output neurons with sigmoid activation. Layer 2 with one output neuron and linear activation. use mean squared loss.
Since, this network does not correspond to regression, we get non-sense output however we are able to run this network due to our dynamic code.
After running this network, we got the output as shown below:

```
LOSS OUTPUT FOR Q3(b)
Loss is at epoch 100 = 3.190508271341567
Loss is at epoch 200 = 2.881713170640052
Loss is at epoch 300 = 2.0544193928245567
Loss is at epoch 400 = 3.4934995084619533
Loss is at epoch 500 = 3.0682536005299363
Loss is at epoch 600 = 2.0817771047361044
Loss is at epoch 700 = 2.80745719356828 44
Loss is at epoch 800 = 2.6764347439228953
Loss is at epoch 900 = 2.360877584329999
Loss is at epoch 1000 = 1.7868591074316573
Loss is at epoch 1100 = 1.8553002845694198
Loss is at epoch 1200 = 1.8059851953031945
Loss is at epoch 1300 = 1.8780806547355176
Loss is at epoch 1400 = 2.2398990732047506
Loss is at epoch 1500 = 2.1114905466709213
Loss is at epoch 1600 = 1.3242939931047397
Loss is at epoch 1700 = 1.104298009309978
Loss is at epoch 1800 = 1.2106495182666293
Loss is at epoch 1900 = 1.8331463493236269
Loss is at epoch 2000 = 1.2044574590591648
Loss is at epoch 2100 = 1.7069599697095108
Loss is at epoch 2200 = 1.9369717616647322
Loss is at epoch 2300 = 1.290875869676732
Loss is at epoch 2400 = 1.127245813384751
...
Loss is at epoch 9900 = 0.009939291396129077
Loss is at epoch 10000 = 0.0033581551138109826
Classification Accuracy (Training Data ): 1343/1437 = 93.45859429366736 %
Classification Accuracy (Testing Data ): 329/360 = 91.38888888888889 %
```

Figure 8: Output

Problem 4 and Problem 5

Our implementation for problem 4 takes into account multiple input channels as required in problem 5. We have defined 3 matrix operations

- **rotate_180**: Rotates a matrix by 180°.
- **convolve2d**: Takes 2 matrices and Applies convolution operation $f * g$. Here $*$ is the convolution used in ML. Other parameters include stride and padding.
- **Flatten**: Flattens a matrix into a 1d vector.

Using these tools we implement the forward pass and backward pass of CNN.

Forward Pass

Forward pass takes arguments

- **inputs**: An $n \times j \times k$ matrix containing n channels of $j \times k$ inputs.
- **filters**: An $m \times n \times f \times f$ matrix where m is the number of outputs, n is the number of inputs, and each 2d vector stored in this matrix is a $f \times f$ filter where we took $f = 3$ as default.
- **stride**: Strides s for convolution.
- **pad**: Padding size p

and returns an output matrix O of size $m \times o \times q$ where $o = \frac{j+2p-f}{s} + 1$ and $q = \frac{k+2p-f}{s} + 1$. Where each element is a 2d matrix of size $o \times q$ received by using convolve2d on filters and inputs and adding the values.

Backward Pass

Backward pass takes arguments

- **inputs**: An $n \times j \times k$ matrix I containing n channels of $j \times k$ inputs.
- **filters**: An $m \times n \times f \times f$ matrix where m is the number of outputs, n is the number of inputs, and each 2d vector stored in this matrix is a $f \times f$ filter where we took $f = 3$ as default.
- **stride**: Strides s for convolution.
- **pad**: Padding size p
- $\frac{\partial L}{\partial O}$: Derivative of loss with respect to the output of the convolution layer.

and returns an output matrix $\frac{\partial L}{\partial I}$ of size $m \times o \times q$ where $o = \frac{j+2p-f}{s} + 1$ and $q = \frac{k+2p-f}{s} + 1$. Where each element is a 2d matrix of size $o \times q$ received by using convolve2d on filters and inputs and adding the values.

Problem 6

We designed a Convolutional Neural Network (CNN) for the MNIST dataset with the following layers:

- **Layer 1:** Convolutional layer with 16 output channels, followed by flattening and a hyperbolic tangent (tanh) activation function.
- **Layer 2:** Fully connected layer with 10 output neurons and a linear activation function.

The network is trained using softmax cross-entropy loss to optimize its performance on digit classification.

The structure for this Neural Network is implemented inside of the Neural Network Class along with the forward and backward passes for the CNN.

```
Loss is at epoch 100 = 7.829611115635083
Loss is at epoch 200 = 4.407050912370296
Loss is at epoch 300 = 6.969959125635558
Loss is at epoch 400 = 1.4215038453924642
Loss is at epoch 500 = 7.631076385787184
Loss is at epoch 600 = 4.587388425996418
Loss is at epoch 700 = 0.0018879587659568365
Loss is at epoch 800 = 0.37107198401017405
Loss is at epoch 900 = 16.39186673811539
Loss is at epoch 1000 = 6.029140469089237
Loss is at epoch 1100 = 7.011865313474318
Loss is at epoch 1200 = 6.960887181547353
Loss is at epoch 1300 = 0.04045317544468337
Loss is at epoch 1400 = 7.793402940903217
Loss is at epoch 1500 = 5.55828206907313
Loss is at epoch 1600 = 0.7703205238804745
Loss is at epoch 1700 = 0.022108073324481052
Loss is at epoch 1800 = 0.39828430869157316
Loss is at epoch 1900 = 0.12080966766892945
Loss is at epoch 2000 = 0.1201106375881698
Loss is at epoch 2100 = 0.4738066276930871
Loss is at epoch 2200 = 0.008753902612843182
Loss is at epoch 2300 = 0.0003744248462368348
Loss is at epoch 2400 = 0.007297783149179532
Loss is at epoch 2500 = 0.00448791013502818
...
Loss is at epoch 9900 = 6.982992848462439e-07
Loss is at epoch 10000 = 0.0005908581037623484
Classification Accuracy (Training Data ):1361/1437 = 94.71120389700765 %
Classification Accuracy (Testing Data ):328/360 = 91.11111111111111 %
```

Figure 9: Accuracy

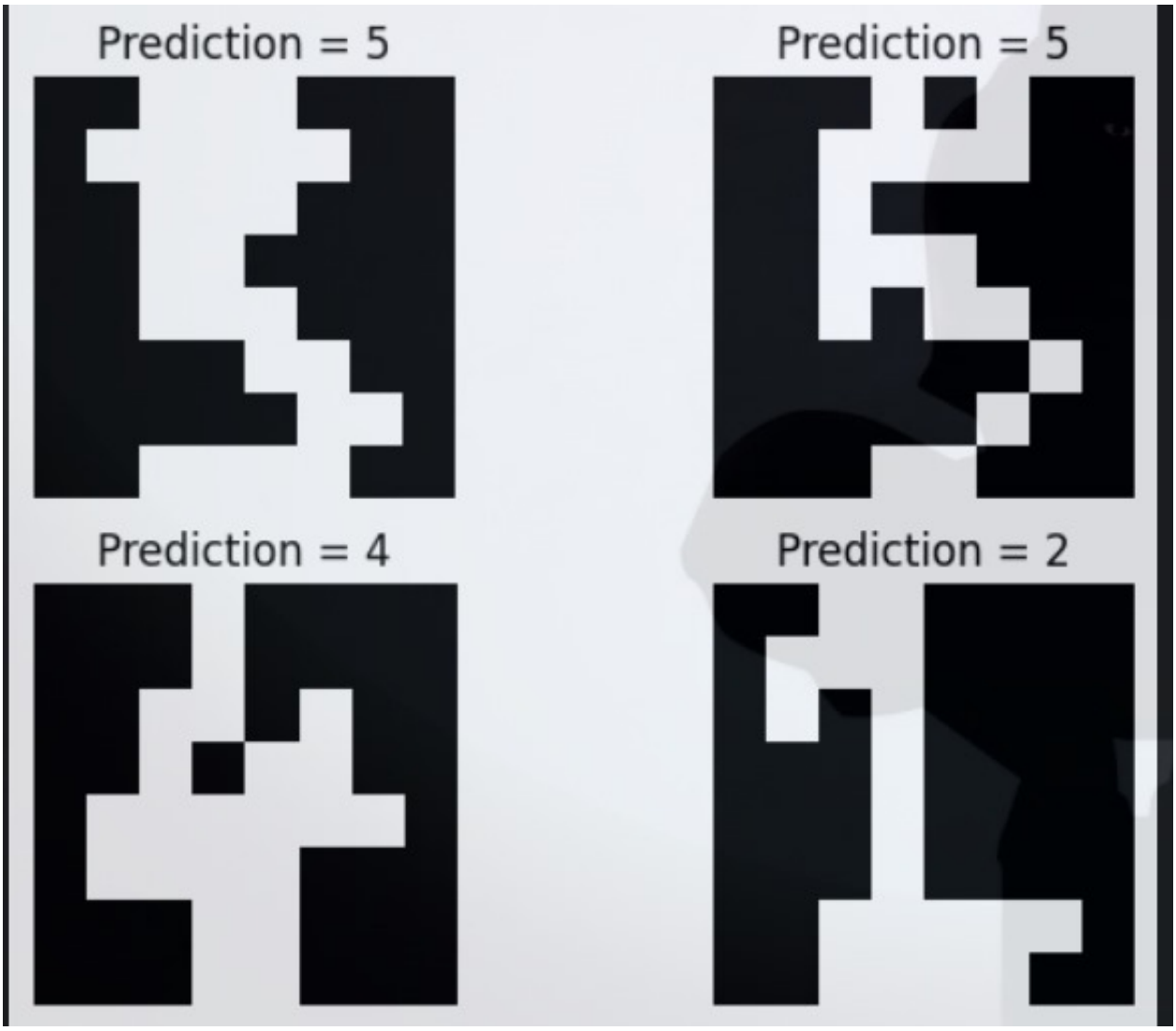

Figure 10: Visualizaing the CNN Output

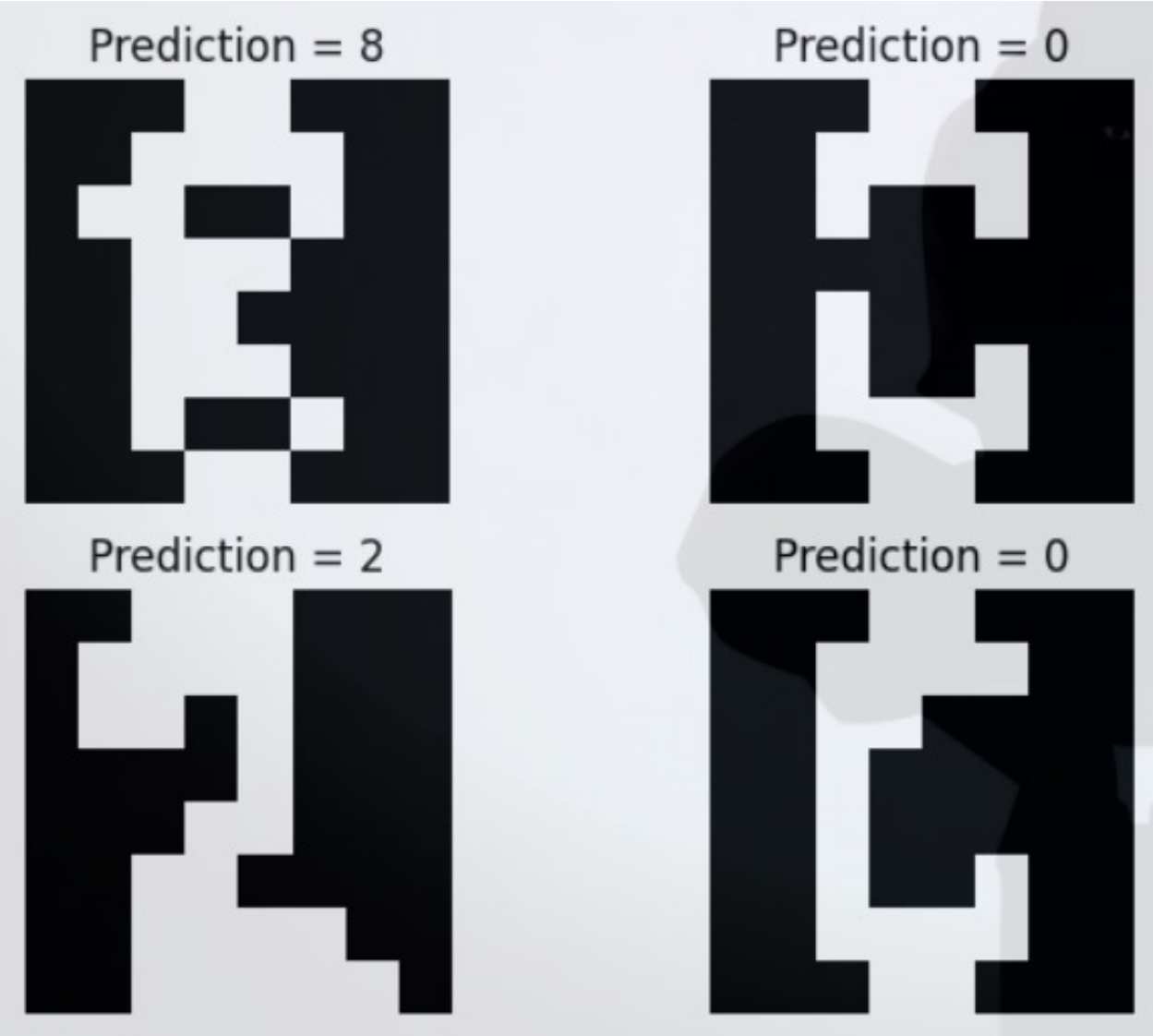

Figure 11: Visualizaing the CNN Output