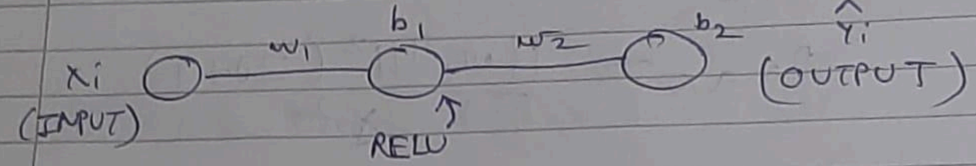
SECTION A

Q1.(a)

Q1.(a) $L = (\hat{y}_i - y_i)^2 \rightarrow$ MSE loss

$\hookrightarrow$ $\hat{y}_i$ being the predicted label for an input x_i & y_i being actual label



After each iteration, we need to find out the weight updation rule.

$$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2} \quad \text{(Gradient Descent)}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_2}$$

$\hat{y}_i = w_2 v_1 + b_2$ where v_1 is the output of RELU in the middle layer & let z_1 be the pre-activate input to that RELU function.

$$= 2(\hat{y}_i - y_i) v_1$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial b_2} = 2(\hat{y}_i - y_i)(1)$$

$$= 2(\hat{y}_i - y_i)$$

$$b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

$\Rightarrow$ Now, for middle layer. $\rightarrow z_1 = w_1 x_i + b_1$

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1} \qquad b_1 = b_1 - \eta \frac{\partial L}{\partial b_1}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_1} \frac{\partial z_1}{\partial w_1} = \frac{\partial L}{\partial v_1} \frac{\partial v_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$$

$$= \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial v_1} \frac{\partial v_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$$

$$= 2(\hat{y}_i - y_i) \frac{\partial (w_2 v_1 + b_2)}{\partial v_1} \frac{\partial \phi(z_1)}{\partial z_1} \frac{\partial (w_1 x_i + b_1)}{\partial w_1}$$

$$= 2(\hat{y}_i - y_i)\, w_2\, x_i \times \phi'(z_1)$$

$$\phi'(z_1) = \frac{\partial}{\partial z_1} \max(0, z_1) = \begin{cases} 1 & z_1 > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial v_1} \frac{\partial v_1}{\partial z_1} \frac{\partial z_1}{\partial b_1}$$

$$= 2(\hat{y}_i - y_i)\, w_2\, \phi'(z_1)\,(1)$$

$$= 2(\hat{y}_i - y_i)\, w_2\, \phi'(z_1)$$

If we set all to 0s, weights will never move away from 0 and could be too slow/to converge.

Now, we train: We sample each weight/bias from $N(0,1)$ and get

$w_2 = -0.15$, $b_2 = 0.85$,

$w_1 = 1.11$, $b_1 = 0.89$

1) Datapoint (1, 3)

- Forward Phase

$z_1 = w_1 x_1 + b_1 = 1.11 + 0.89 = 2$
$v_1 = \phi(z_1) = 2$

$\hat{y}_1 = w_2 v_1 + b_2 = -0.3 + 0.85 = 0.55$

- Backward Phase

$$w_2 = w_2 - 0.01 \times (2(0.55-3)2)$$
$$= -0.15 - 0.04 \times (-2.45)$$
$$= 0.098 - 0.15$$
$$= -0.052$$

$$b_2 = b_2 - 0.01(2(0.55-3))$$
$$= 0.85 + 0.049$$
$$= 0.899$$

$$w_1 = w_1 - 0.01(2(0.55-3)(-0.052)(1)(1))$$

$$= 1.11 - 0.002548$$
$$= 1.107452$$

$$b_1 = b_1 - 0.01(2(0.55-3)(-0.052))$$

$$= 0.89 - 0.002548 = 0.887452$$

2) Datapoint $(2, 4)$

- Forward Phase

$$z_1 = w_1 x_2 + b_1 = 3.102$$
$$v_1 = 3.102$$

$$\hat{y}_1 = w_2 v_1 + b_2 = 0.738$$

- Backward Phase

$$w_2 = w_2 - 0.01(2(0.738-4)(3.102))$$
$$= -0.052 + 0.20237448$$

$$= 0.15037448$$

$$b_2 = b_2 - 0.01(2(0.738-4))$$
$$= 0.899 + 0.06524 = 0.96424$$

$w_1 = w_1 - 0.01\left(2(0.738-4)(0.15)(2)(1)\right)$
$= 1.107452 + 0.019572$
$= 1.127024$

$b_1 = b_1 - 0.01\left(2(0.738-4)(0.15)(1)\right)$
$= 0.887452 + 0.009786$
$= 0.897238$

3) Datapoint (3, 5)

- Forward Phase $\rightarrow$ $z_1 = w_1 x_3 + b_1 = 4.278$
  $v_1 = 4.278$
  $\hat{y}_1 = w_2 v_1 + b_2 = 1.608$

- Backward Phase $\rightarrow$ $w_2 = w_2 - 0.01\left(2(1.608-5)(4.278)\right)$
  $= 0.15037448 + 0.29021952$
  $= 0.440594$

$b_2 = b_2 - 0.01\left(2(1.608-5)\right)$
$= 0.96424 + 0.06784 = 1.03208$

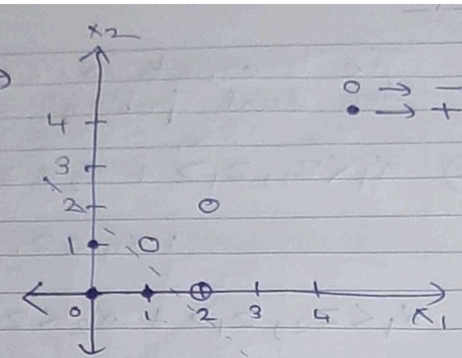
$w_1 = w_1 - 0.01\left(2(1.608-5)(0.44)(3)(1)\right)$
$= 1.127024 + 0.0895488$
$= 1.2165728$

$b_1 = b_1 - 0.01\left(2(1.608-5)(0.44)(1)\right)$
$= 0.897238 + 0.0298496 = 0.9270876$

$$\boxed{\begin{aligned} \therefore \text{So, } w_2 &\approx 0.44 \\ b_2 &\approx 1.032 \\ w_1 &\approx 1.217 \\ b_1 &\approx 0.927 \end{aligned}}$$

(b)

Q1.(b) (a) Plot →



∴ Clearly, the points are linearly separable.

Q1.(b)(b) First finding gram matrix by using

$\alpha_1 \to (0,0)$ $\quad \alpha_3 \to (0,1)$ $\quad \alpha_5 \to (2,2)$

$\alpha_2 \to (1,0)$ $\quad \alpha_4 \to (1,1)$ $\quad \alpha_6 \to (2,0)$

$$\text{Matrix} \to \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 & 2 \\ 0 & 0 & 1 & 1 & 2 & 0 \\ 0 & 1 & 1 & 2 & 4 & 2 \\ 0 & 2 & 2 & 4 & 8 & 4 \\ 0 & 2 & 0 & 2 & 4 & 4 \end{bmatrix}$$

$$Q(\alpha) \to \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6$$
$$-\frac{1}{2}\alpha_2^2 + \alpha_2\alpha_4 + 2\alpha_2\alpha_5 + 2\alpha_2\alpha_6$$
$$-\frac{1}{2}\alpha_3^2 + \alpha_3\alpha_4 + 2\alpha_3\alpha_5$$
$$-\alpha_4^2 - 4\alpha_4\alpha_5 - 2\alpha_4\alpha_6 - 4\alpha_5^2 - 4\alpha_5\alpha_6$$
$$-2\alpha_6^2$$

Differentiating with respect to α,

$$\alpha_2 - \alpha_4 - 2\alpha_5 - 2\alpha_6 = 1$$
$$\alpha_3 - \alpha_4 - 2\alpha_5 = 1$$
$$-\alpha_2 - \alpha_3 + 2\alpha_4 + 4\alpha_5 + 2\alpha_6 = 1$$
$$-2\alpha_2 - 2\alpha_3 + 4\alpha_4 + 8\alpha_5 + 4\alpha_6 = 1$$
$$-2\alpha_2 + 2\alpha_4 + 4\alpha_5 + 4\alpha_6 = 1$$

These are constrained by,

$$\sum \alpha_i y_i = 0 \Rightarrow \alpha_1 + \alpha_2 + \alpha_3 = \alpha_4 + \alpha_5 + \alpha_6$$

$$\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6 \geq 0$$

On solving using online QP calculator,

$\alpha_1 = 0$ $\alpha_4 = 3$
$\alpha_2 = 3$ $\alpha_5 = 0$
$\alpha_3 = 1$ $\alpha_6 = 1$

So, 4 support vectors, $(1,0)$, $(0,1)$, $(1,1)$, $(2,0)$

$$w = \sum \alpha_i y_i x_i$$

$$= \begin{bmatrix}3\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}3\\3\end{bmatrix} - \begin{bmatrix}2\\0\end{bmatrix}$$

$$= \begin{bmatrix}-2\\-2\end{bmatrix}$$

$$b = \frac{1}{4} \sum (y_i - w^T x_i)$$

$\downarrow$

for each support vectors

$$= \frac{1}{4}\Big((1-(-2)) + (1-(-2)) + (-1-(-4)) + (-1-(-4)) \Big)$$

$$= \frac{1}{4}(12) = 3 = b$$

$\therefore$ Hyperplane $\rightarrow w^T x + b$

$$= -2x_1 - 2x_2 + 3$$

Weight vector (without bias) $\rightarrow \begin{bmatrix} -2 \\ -2 \end{bmatrix}$ With bias $\rightarrow \begin{bmatrix} -2 \\ -2 \\ 3 \end{bmatrix}$

Support Vectors $\rightarrow$ $(1,0)$, $(0,1)$, $(1,1)$, $(2,0)$.

(c)

Q1.(c) (a) Margin $= \frac{2}{||w||} = \frac{2}{\sqrt{w_1^2 + w_2^2}}$

$= \frac{2}{\sqrt{4}} = 1$

$\therefore$ Margin $= 1$ unit

(b) Support vectors are the ones with a geometrical margin $= \frac{\text{Margin}}{2}$ because they form the 2 hyperplanes that define the bondaries of the classes during training. So, they are at distance $\frac{\text{margin}}{2}$ from the $w^T x + b$ hyperplane.

$$G.M. = \frac{y_i (w^T x_i + b)}{||w||}$$

$\frac{\text{Margin}}{2} = 0.5$ units

$$w^T x_i + b = w_1 x_1 + w_2 x_2 + b$$
$$= -2x_1 + 5$$

G.M. for sample 1 $\rightarrow$ $\frac{1(-2(1)+5)}{2} = 1.5$

2 $\rightarrow$ $\frac{1(-2(2)+5)}{2} = 0.5$

3 $\rightarrow$ $\frac{-1(-2(3)+5)}{2} = 0.5$

Sample 4 $\rightarrow$ $\frac{-1(-2(4)+5)}{2}$

$= 1.5$

G.M. for Samples 2 & 3 $= \frac{\text{Margin}}{2} = 0.5$

$\boxed{\therefore \text{Samples 2 \& 3 are support vectors}}$

(c) Decision boundary $\rightarrow$ $w^T x_i + b$

$\downarrow$

if for a point,

$w^T x_i + b \geq 0$, $y_i = 1$

$w^T x_i + b < 0$, $y_i = -1$

$w^T x_i + b = -2x_{i1} + 5$

For $(1, 3)$, $-2x_{i1} + 5 = 5 - 2$

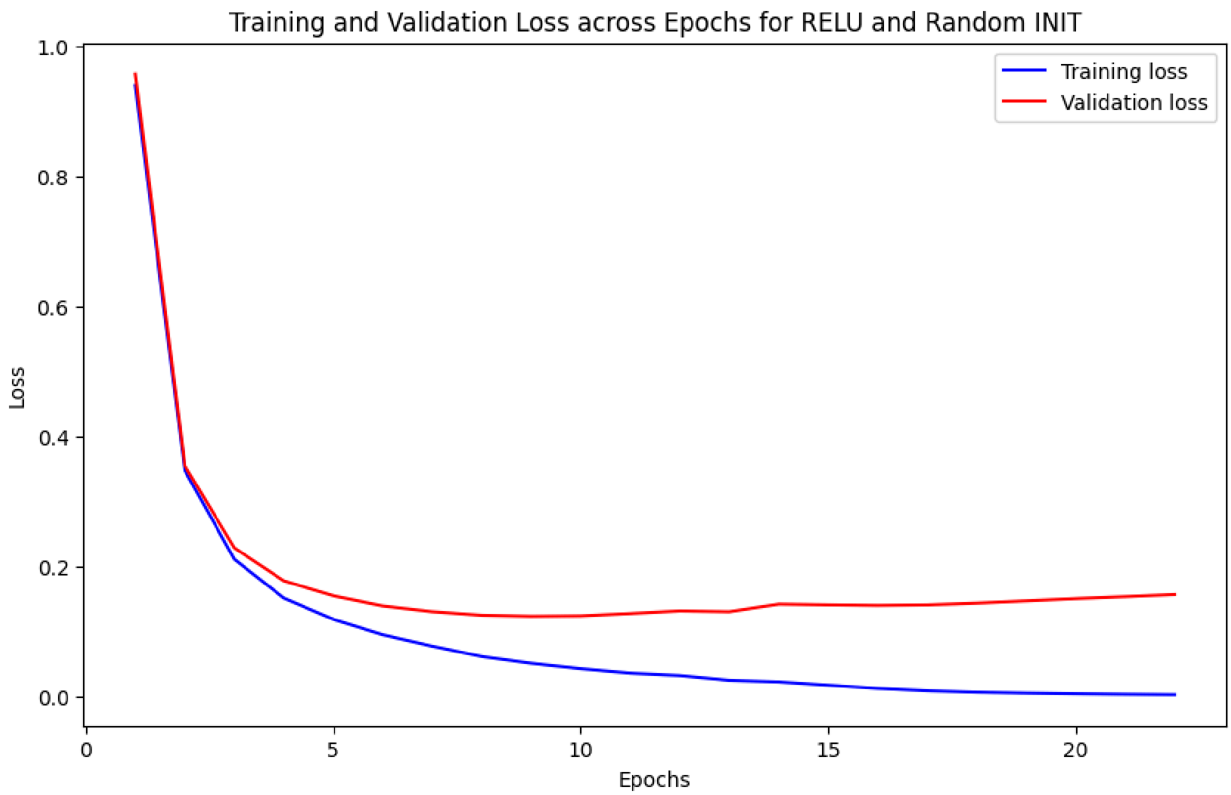
$= 3 \geq 0$

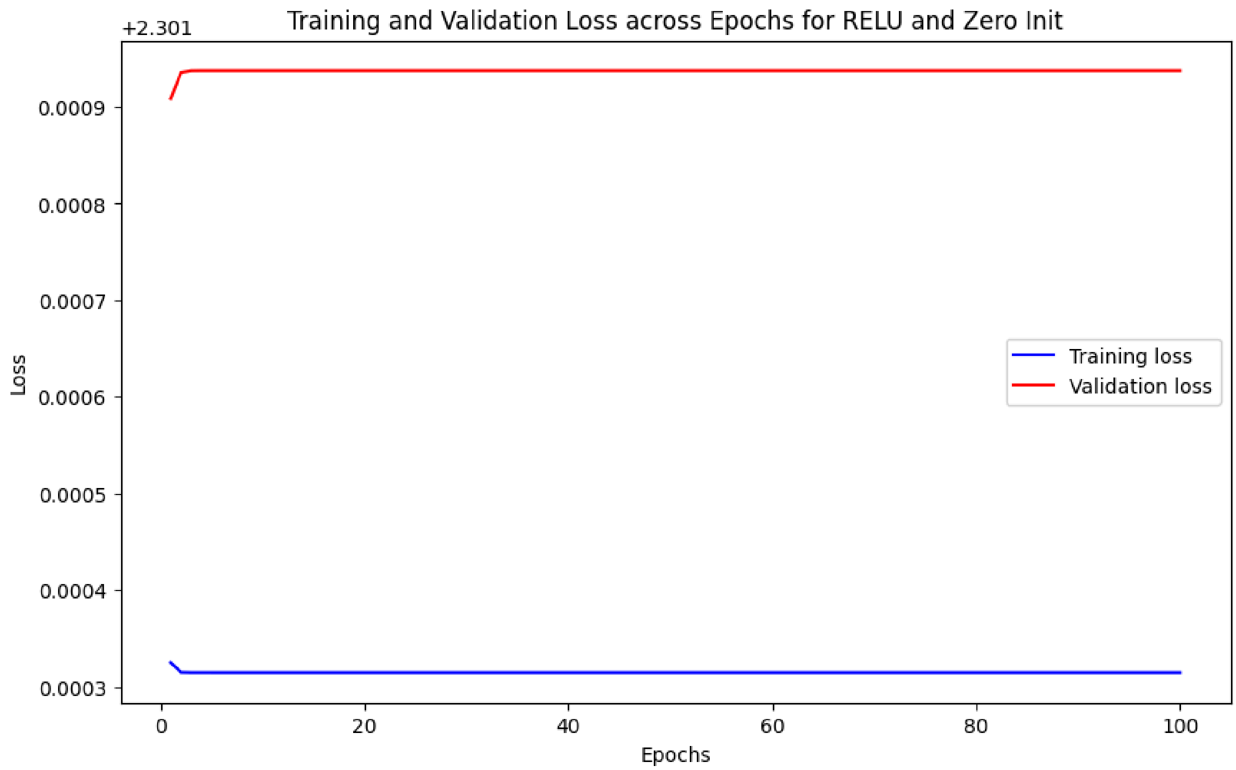
$\boxed{\therefore \text{Belongs to class } y = +1}$

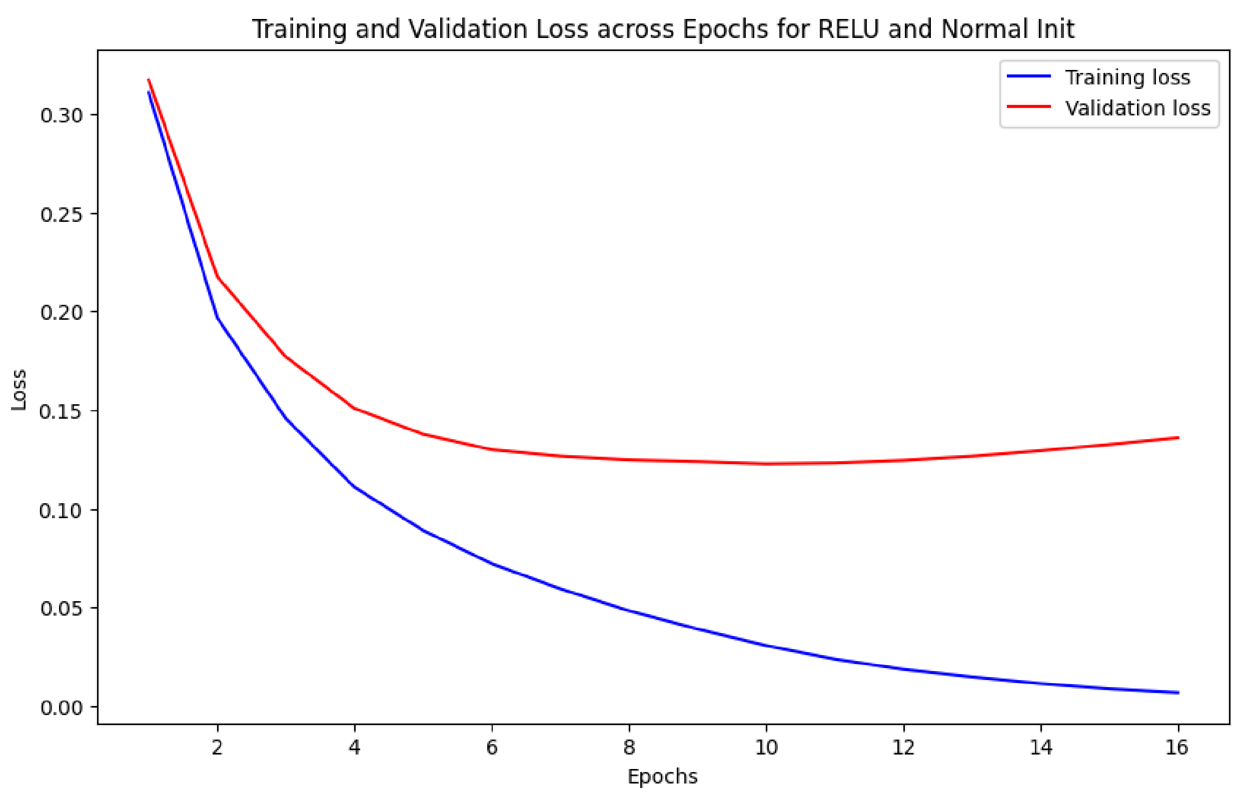
SECTION B

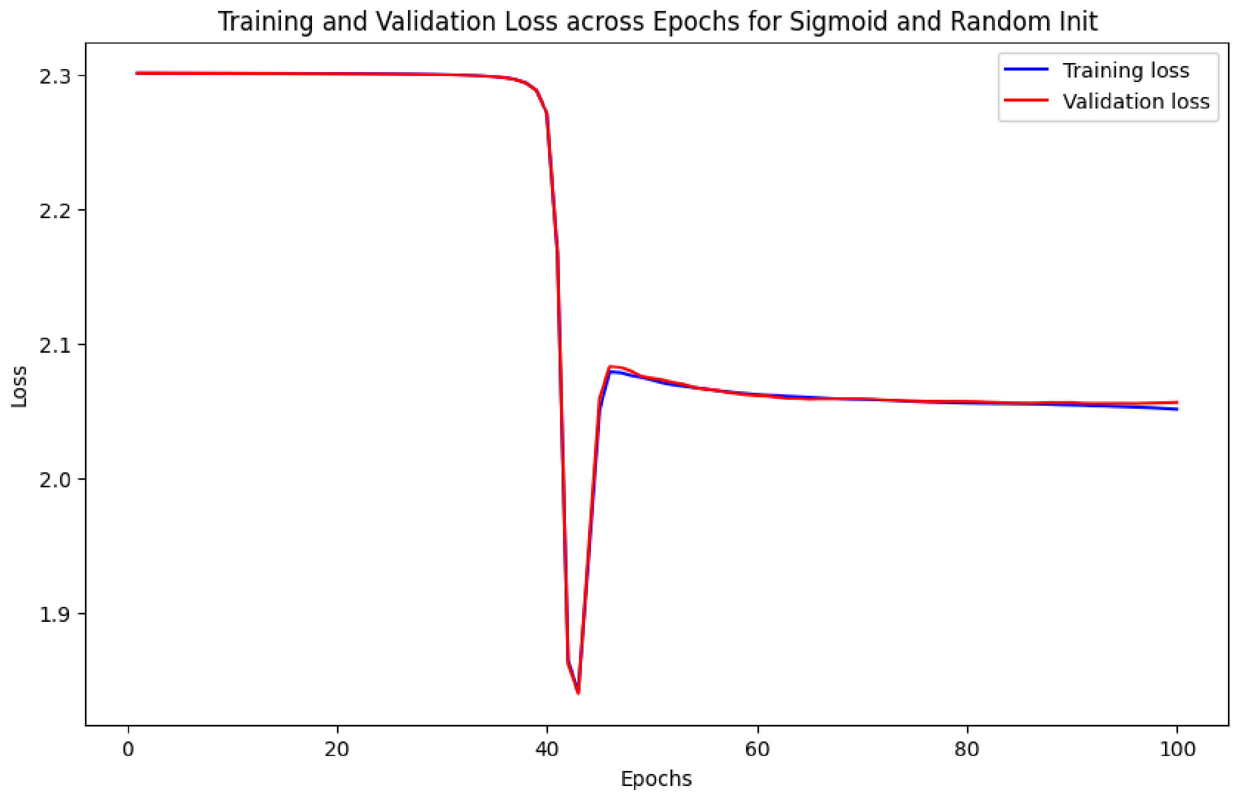
Q3) Initializing random weights from -0.1 to 0.1. Also, scaled the normal weights to have a std deviation of 0.1.

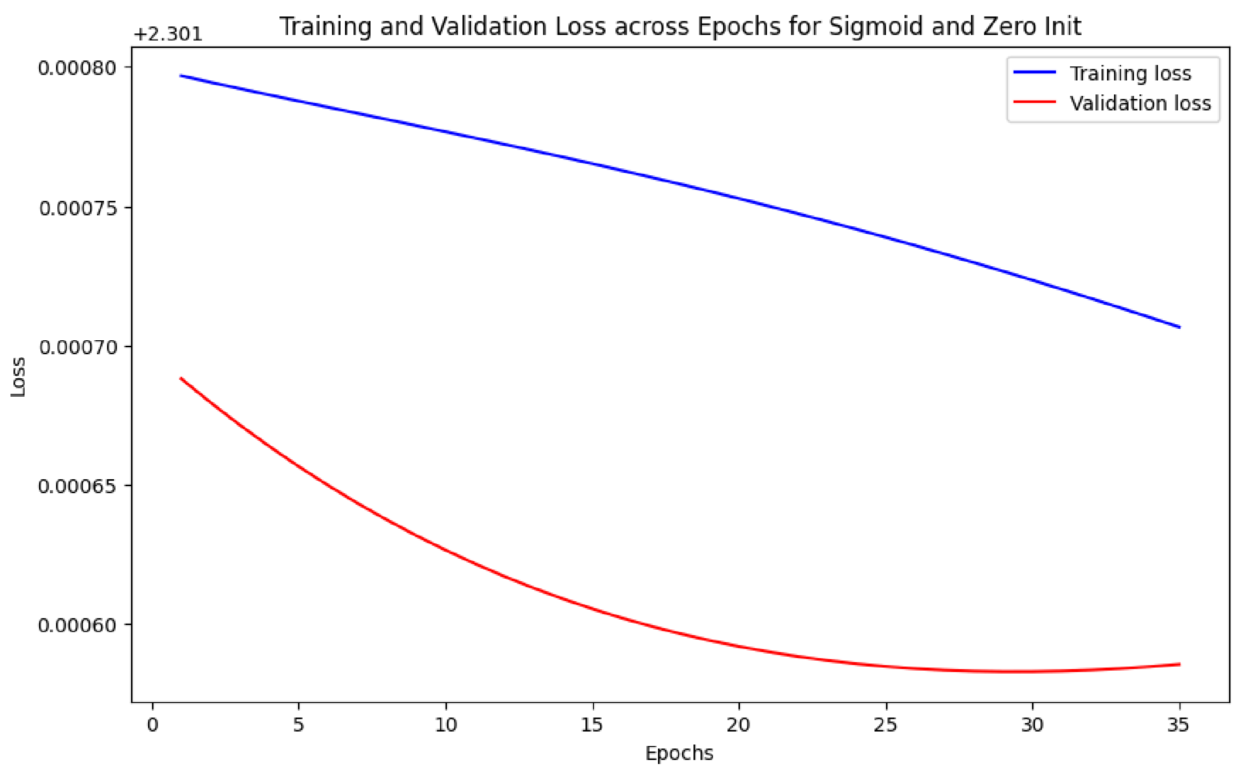
Q4) Did scaling by dividing the pixels by 255, to bring them in the range [0,1].

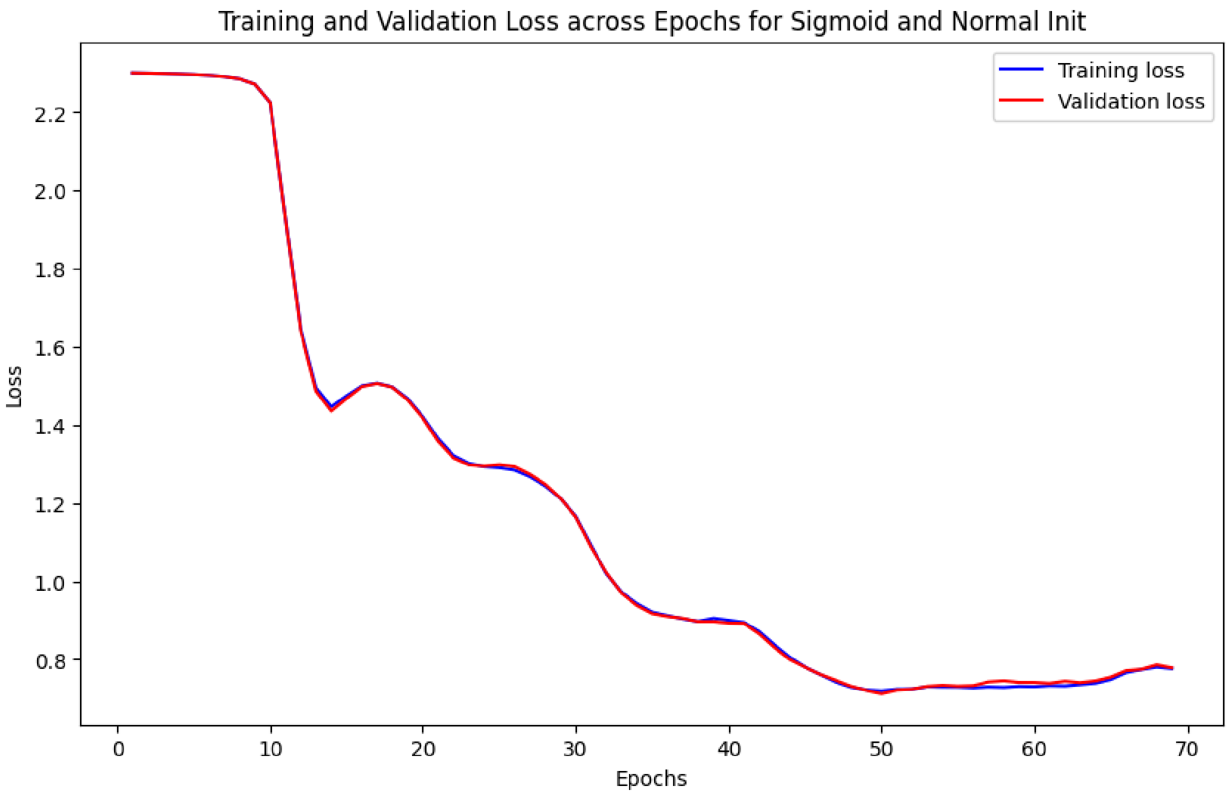


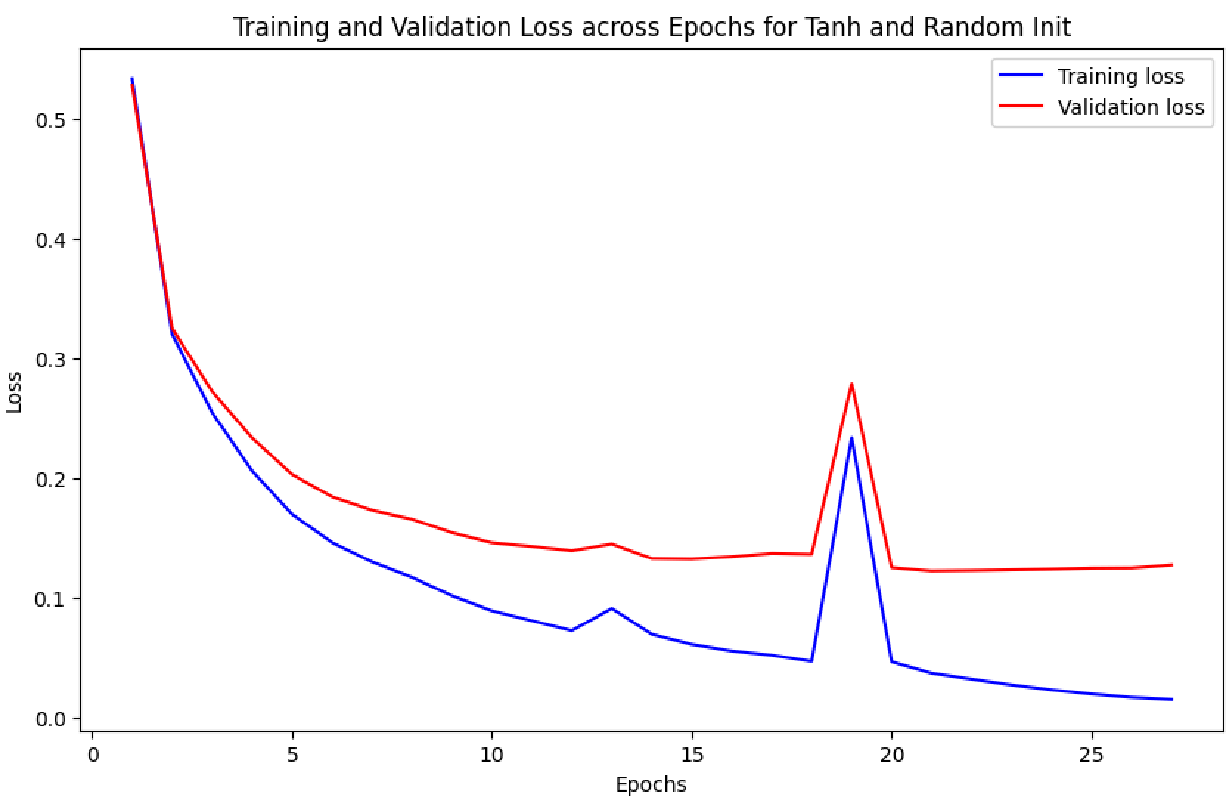

Training and Validation Loss across Epochs for RELU and Zero Init
+2.301
0.0009
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
Loss
0
20
40
60
80
100
Epochs
Training loss
Validation loss

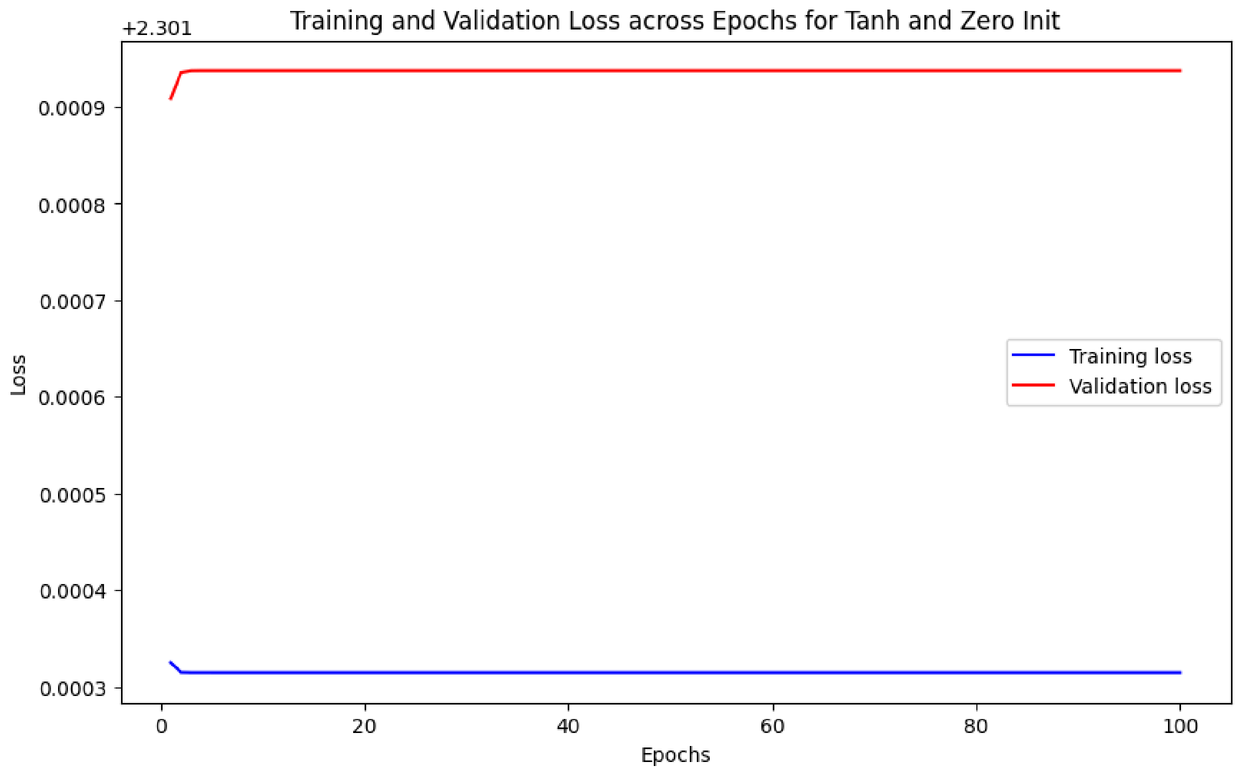

Training and Validation Loss across Epochs for RELU and Normal Init
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Loss
2
4
6
8
10
12
14
16
Epochs
Training loss
Validation loss


Training and Validation Loss across Epochs for Sigmoid and Random Init
Training loss
Validation loss
Loss
Epochs
2.3
2.2
2.1
2.0
1.9
0
20
40
60
80
100

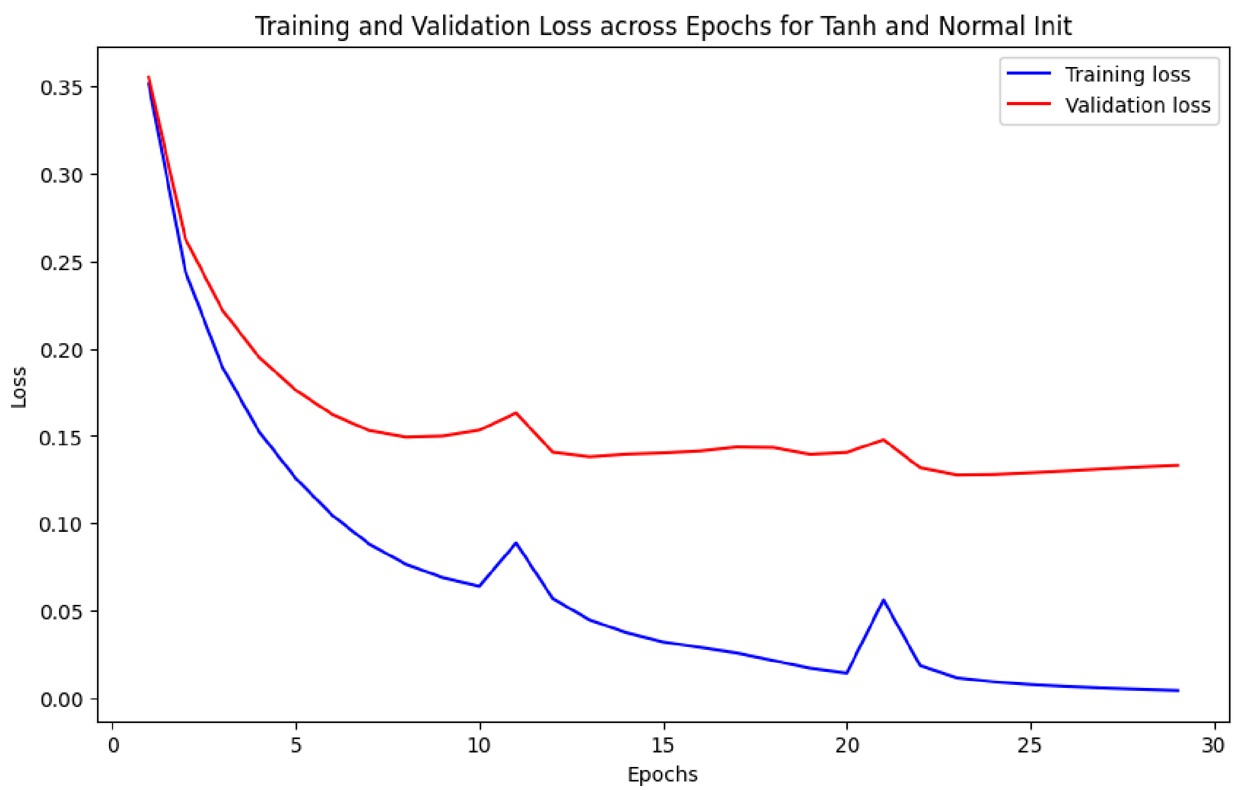

Training and Validation Loss across Epochs for Sigmoid and Zero Init
+2.301
Training loss
Validation loss
Loss
Epochs
0.00080
0.00075
0.00070
0.00065
0.00060
0
5
10
15
20
25
30
35


Training and Validation Loss across Epochs for Sigmoid and Normal Init
Training loss
Validation loss
Loss
Epochs
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0
10
20
30
40
50
60
70

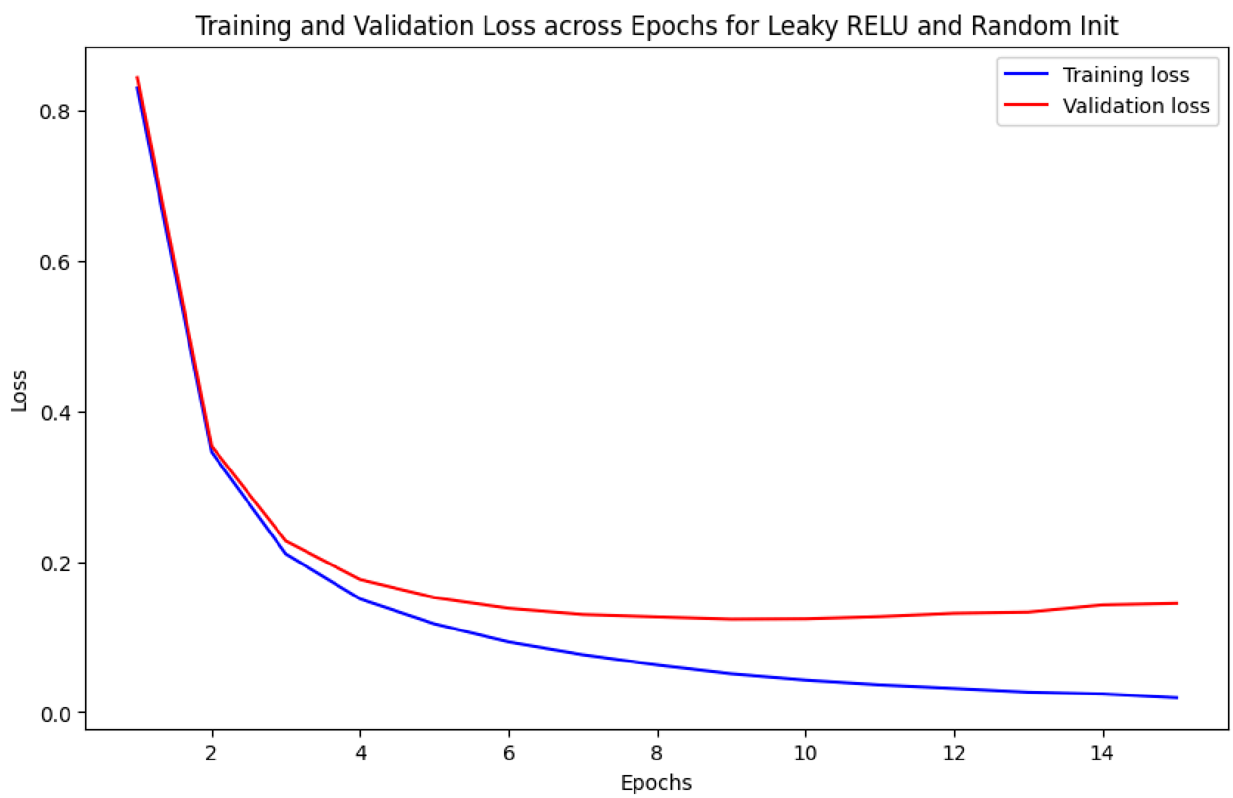

Training and Validation Loss across Epochs for Tanh and Random Init
Training loss
Validation loss
Loss
Epochs
0.5
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25

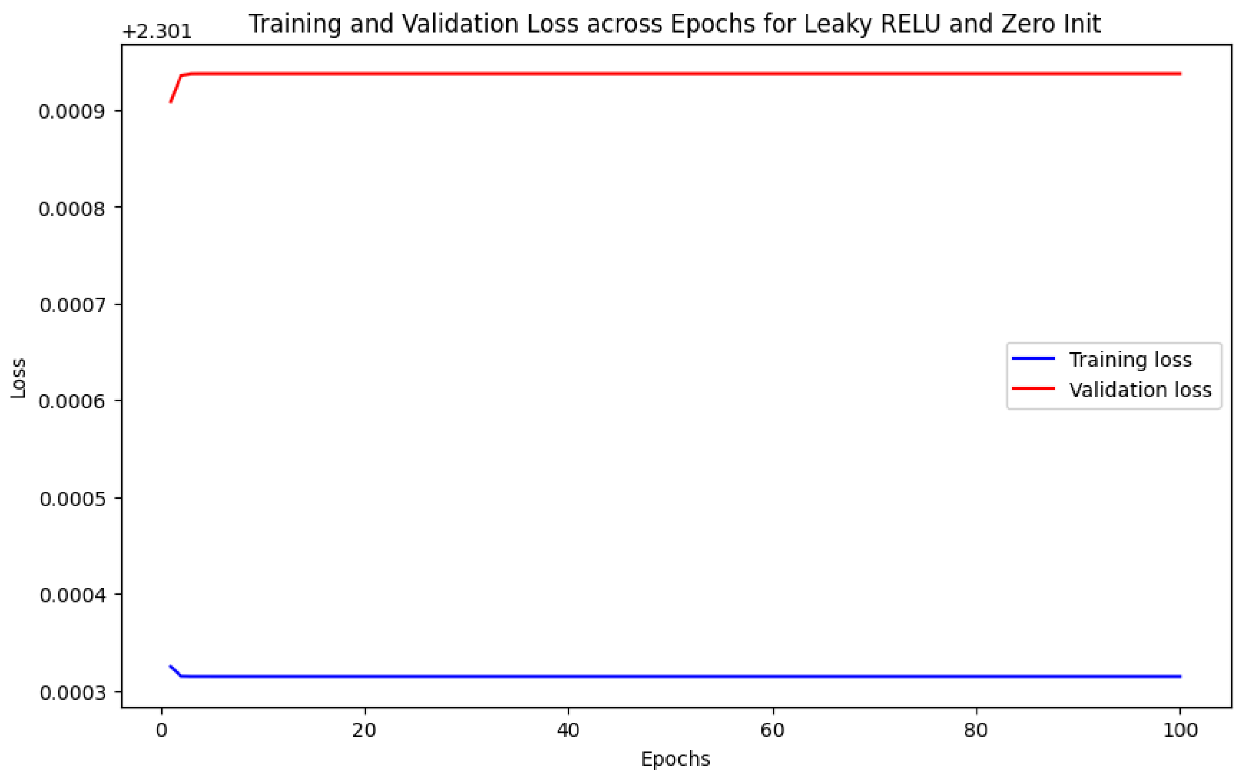


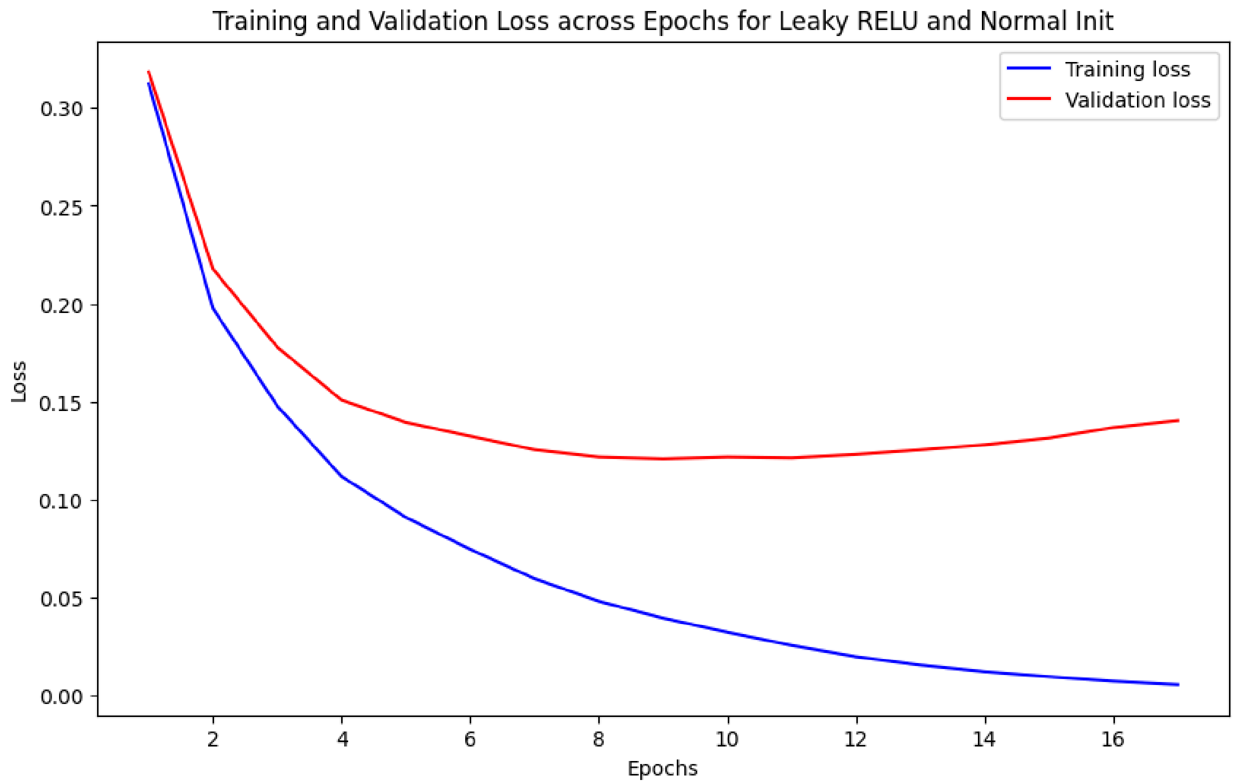
Training and Validation Loss across Epochs for Tanh and Zero Init
+2.301
0.0009
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
Loss
0
20
40
60
80
100
Epochs
Training loss
Validation loss



Training and Validation Loss across Epochs for Tanh and Normal Init
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Loss
0
5
10
15
20
25
30
Epochs
Training loss
Validation loss


Training and Validation Loss across Epochs for Leaky RELU and Random Init
Training loss
Validation loss
Loss
Epochs
0.0
0.2
0.4
0.6
0.8
2
4
6
8
10
12
14


Training and Validation Loss across Epochs for Leaky RELU and Zero Init
+2.301
Training loss
Validation loss
Loss
Epochs
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0.0009
0
20
40
60
80
100



Accuracies:-

Score by Relu_Random: The accuracy is 96.28%.
Score by Relu_Random: The accuracy is 11.1%.
Score by Relu_Normal: The accuracy is 96.6%.

Score by Sigmoid_Random: The accuracy is 26.77%.
Score by Sigmoid_Zero: The accuracy is 11.1%.
Score by Sigmoid_Normal: The accuracy is 77.8%.

Score by Tanh_Random: The accuracy is 96.43%.
Score by Tanh_Zero: The accuracy is 11.1%.
Score by Tanh_Normal: The accuracy is 96.38%.

Score by LeakyReLU_Random: The accuracy is 96.12%.
Score by LeakyReLU_Zero: The accuracy is 11.1%.
Score by LeakyReLU_Normal: The accuracy is 96.3%.

Findings:-

- When initializing the weights with 0, they never manage to stay away from 0, which leads to poor training, and hence, have rather poor accuracy.

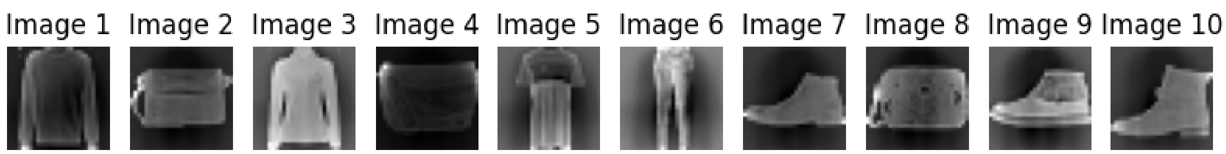
- In terms of the rest, apart from sigmoid, all the others have trained equally as well and have around 96% accuracy. The best was with relu as the activation function and weights initialized normally.

- Finally, for sigmoid, it trains too slowly, because of the nature of its function and because the initial values of the pixels and weights are small, it never improves in time within the 100 iterations, and so the accuracies are rather low.

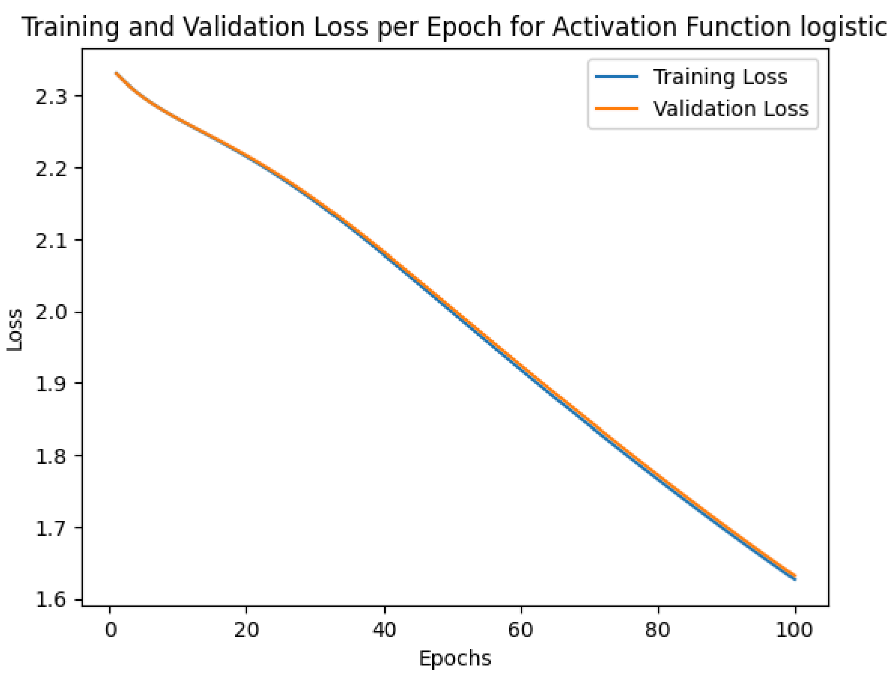
- A higher learning rate with early stopping had a much better effect on the model as compared to higher number of iterations, largely in order to overcome the small weights/activations and help it train faster.

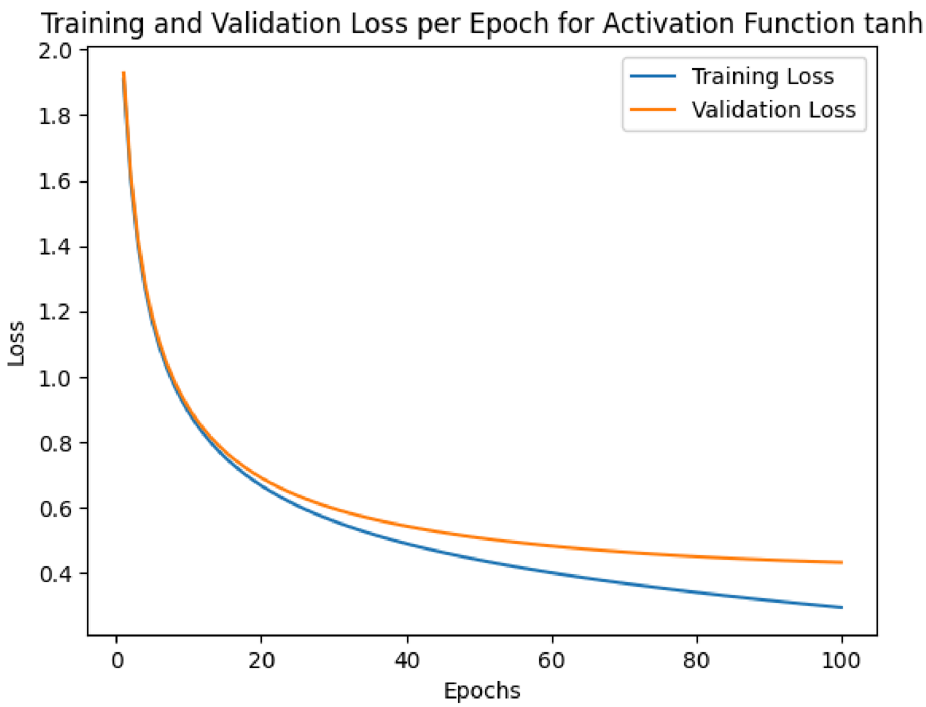
SECTION C

Q1)

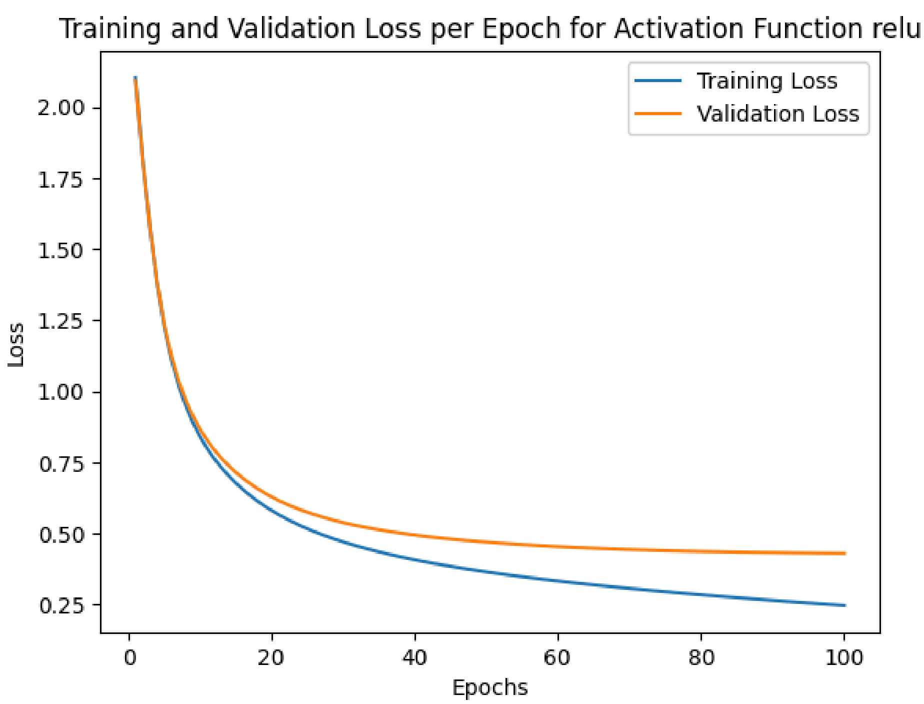


Q2)

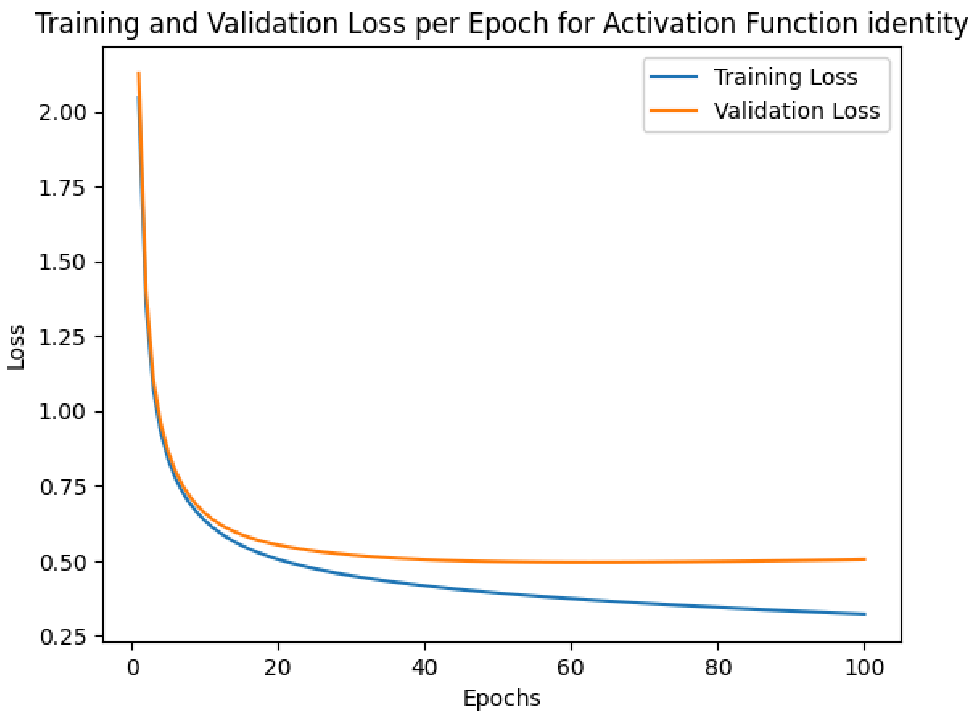




Training and Validation Loss per Epoch for Activation Function tanh
Training Loss
Validation Loss
Loss
Epochs
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
20
40
60
80
100



Training and Validation Loss per Epoch for Activation Function relu
Training Loss
Validation Loss
Loss
Epochs
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
20
40
60
80
100



Accuracies:-

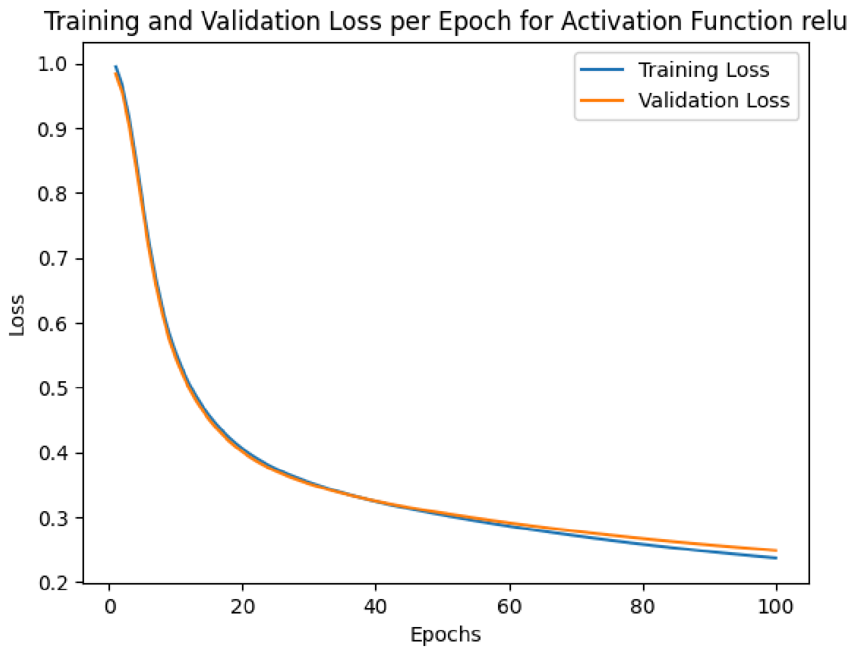
Logistic -> 53.85%
Tanh -> 84%
Relu -> 83.85%
Identity -> 82.8%

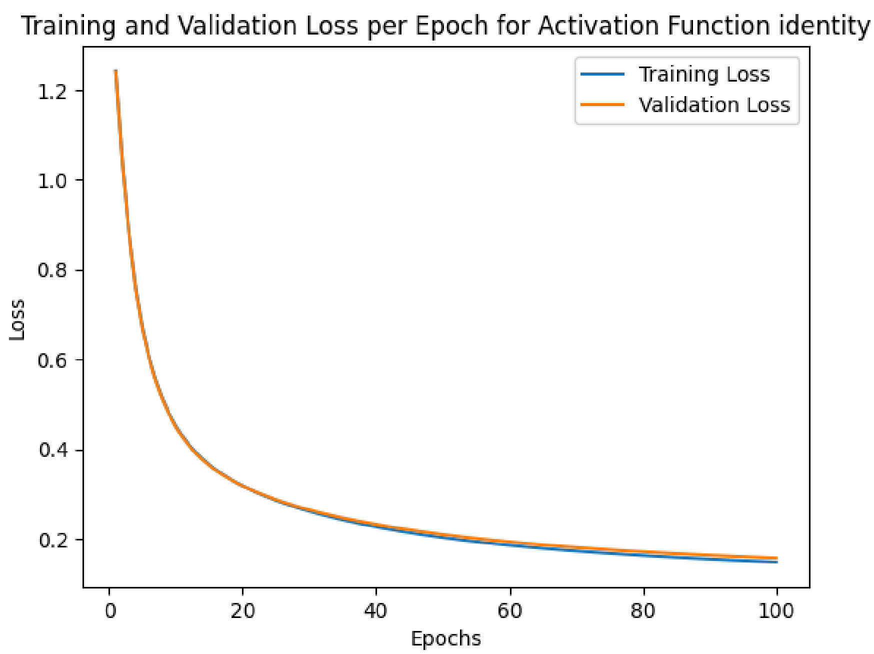
Tanh was the best activation function, based on comparing the accuracies on the test set.

Q3) {'alpha': 0.0001, 'batch_size': 64, 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 100, 'solver': 'adam'}

Q4.(a) c = 512, b = 256, a = 128

(b)


Training and Validation Loss per Epoch for Activation Function relu
Training Loss
Validation Loss
Loss
Epochs
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100


Training and Validation Loss per Epoch for Activation Function identity
Training Loss
Validation Loss
Loss
Epochs
1.2
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100

(d) Relu:

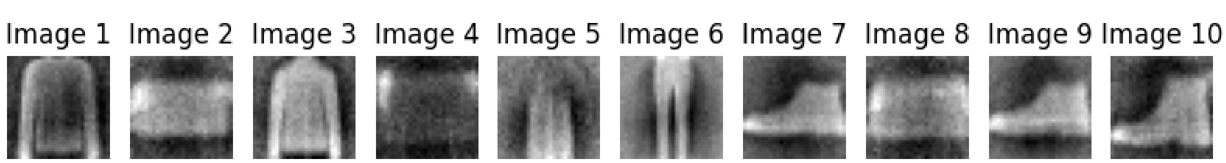
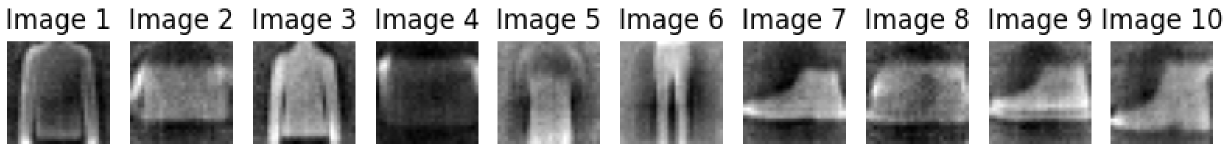


Image 1
Image 2
Image 3
Image 4
Image 5
Image 6
Image 7
Image 8
Image 9
Image 10

Identity:




```
Observations for Relu:-

1) The edges are not very clear.
2) The reconstruction reconstructed most of the features well, but there's
a lot of noise/blurness which leads to a not-so-great reconstruction.
3) It lacks the intensity and sharpness of the initial image in the test
set.
4) The lighter colours have translated well, but the darker colours
haven't regenerated as well.
5) The intensity/spectral stuff around the objects in the test set have
regenerated and transformed in the form of a dark column for each instead,
instead of mapping to the outskirts of the object, so the model hasn't
trained/captured that well.
```

```
Observations for Identity activation fn:-

1) The edges are much more clear compared to the ones with the relu model.
2) The reconstruction reconstructed most of the features well, but there's
clearly a lot of noise/blurness which leads to a not-so-great
reconstruction.
3) It lacks the intensity and sharpness of the initial image in the test
set.
4) It does better with the darker colours, but still not great.
5) The intensity/spectral stuff around the objects in the test set have
regenerated and transformed in the form of a dark column for each instead,
instead of mapping to the outskirts of the object, so the model hasn't
trained/captured that well.
```

Q5) Accuracy with Relu: 75.25%
Accuracy with Identity: 75.45%

Observations:-

```
This method still is a decent classifier(8.55 % less than in Part 2) since
the hidden layers capture crucial information with regards to the details
```

```
of the image, like its shape, intensity, the nature of the object,
otherwise it wouldn't be able to reconstruct the image at all. Because of
this, when we use the hidden layer's representation/embedding, the
classification works pretty well, as those features itself are used to
identify an object.

So, to summarise, reasons:-

1) Hidden Layers capture crucial info like shape, borders, gradients, type
of object, etc.
2) These details themselves are used in the classifier in Part 2, where we
slowly reduced the size of the hidden layer.
3) So, during classification, since they carry the info regarding the
features, they are a pretty decent classifier.
```