Report - Assignment 3 (ML - CSE343)

- **Arnav Goel (Roll No: 2021519)**

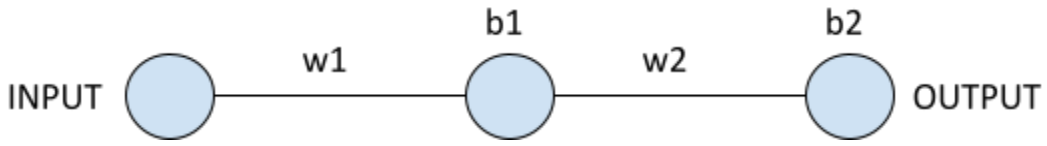
Section A:

Question 1:

We are given the following points with their corresponding targets:

Input	Target
1.2	3.0
0.8	2.5
2.0	4.0

We are asked to consider this a regression problem and solve this using an MLP with one hidden layer activated by ReLU activation. Thus, we assume the following architecture (as allowed):



We assume that the ReLU function activates one neuron in our hidden layer. Since our input is one-dimensional, we have one input neuron. Also, we have one neuron in the output layer as we have a regression task and need to output a linear term. Thus, in a forward pass, the following operations will be performed:

Let input be x_0 and target be y_0
Therefore, input to hidden layer neuron:
$w_1 x_0 + b_1 = z_1$
ReLU activates this:
$a_1 = ReLU(z_1)$
Thus, to the output layer:
$w_2 a_1 + b_2 = z_2$
$a_2 = z_2$

a_2 is our output, which we use to compute MSE loss with target output y_0 and thus perform back-propagation.

Now, for performing a single training iteration, we assume the following values of the weights and the bias:

$$w_1 = 1.0, b_1 = 1.0$$
$$w_2 = 2.0, b_2 = 2.0$$

The equations for backpropagation are as follows:

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2} = (a_2 - y_{\text{true}}) \cdot 1 \cdot a_2$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2} = (a_2 - y_{\text{true}}) \cdot 1 \cdot 1$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} = (a_2 - y_{\text{true}}) \cdot 1 \cdot w_2 \cdot (1 \text{ if } z_1 > 0 \text{ else } 0) \cdot x$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1} = (a_2 - y_{\text{true}}) \cdot 1 \cdot w_2 \cdot (1 \text{ if } z_1 > 0 \text{ else } 0) \cdot 1$$

$$\begin{aligned} w_1 &= w_1 - \eta \frac{\partial L}{\partial w_1} \\ b_1 &= b_1 - \eta \frac{\partial L}{\partial b_1} \\ w_2 &= w_2 - \eta \frac{\partial L}{\partial w_2} \\ b_2 &= b_2 - \eta \frac{\partial L}{\partial b_2} \end{aligned}$$

Here η is the learning rate equal to 0.01 and L is the MSE loss computed using the following equation:

$$L = \frac{1}{2} (y_{\text{pred}} - y_{\text{true}})^2$$

Iteration 1:
1st sample:
$x_1 = 1.2$, $y_t rue = 3.0$

$$z_{1_1} = 1.0 \cdot 1.2 + 1.0 = 2.2$$

$$\begin{aligned} a_{1_1} &= \max(0, 2.2) = 2.2 \\ z_{2_1} &= 2.0 \cdot 2.2 + 2 = 6.4 \end{aligned}$$

$$a_{2_1} = 6.4$$

$$y_{\text{pred}_1} = a_{2_1} = 6.4$$

$$L_1 = \tfrac{1}{2}\left(y_{\text{pred}_1} - y_{\text{true}_1}\right)^2 = \tfrac{1}{2}(6.4 - 3.0)^2 = 11.56$$

$$\frac{\partial L}{\partial w_2} = (6.4 - 3.0) \cdot 1 \cdot 2.9 = 1.16$$

$$\frac{\partial L}{\partial b_2} = (6.4 - 3.0) \cdot 1 \cdot 1 = 3.4$$

$$\frac{\partial L}{\partial w_1} = (6.4 - 3.0) \cdot 1 \cdot 1 \cdot 1 \cdot 1.2 = 4.08$$

$$\frac{\partial L}{\partial b_1} = (6.4 - 3.0) \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 3.4$$

$$w_1 = 1.0 - 0.01 \cdot 4.08 = 0.9592$$

$$b_1 = 1.0 - 0.01 \cdot 3.4 = 0.966$$

$$w_2 = 2.0 - 0.01 \cdot 7.48 = 1.9252$$

$$b_2 = 2.0 - 0.01 \cdot 3.4 = 1.966$$

2nd sample:
$x_2 = 0.8,\ y_true = 2.5$

$$z_{1_2} = 0.9592 \cdot 0.8 + 0.966 = 1.733$$

$$a_{1_2} = \max(0, 1.733) = 1.733$$
$$z_{2_2} = 1.9252 \cdot 1.733 + 1.966 = 5.302$$

$$a_{2_2} = 5.302$$

$$y_{\text{pred}_2} = a_{2_2} = 5.302$$

$$L_2 = \tfrac{1}{2}\left(y_{\text{pred}_2} - y_{\text{true}_2}\right)^2 = \tfrac{1}{2}(5.3 - 2.5)^2 = 3.92$$

$$\frac{\partial L}{\partial w_2} = (5.3 - 2.5) \cdot 1 \cdot 1.733 = 4.85$$

$$\frac{\partial L}{\partial b_2} = (5.3 - 2.5) \cdot 1 \cdot 1 = 2.8$$

$$\frac{\partial L}{\partial w_1} = (5.3 - 2.5) \cdot 1 \cdot 1 \cdot 1 \cdot 0.8 = 2.24$$

$$\frac{\partial L}{\partial b_1} = (5.3 - 2.5) \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 2.8$$

$$w_1 = 0.9592 - 0.01 \cdot 2.24 = 0.9368$$

$$b_1 = 0.966 - 0.01 \cdot 2.8 = 0.938$$

$$w_2 = 1.9252 - 0.01 \cdot 4.85 = 1.8767$$

$$b_2 = 1.966 - 0.01 \cdot 2.8 = 1.938$$

3rd sample:

$x_3 = 2.0$, $y_true = 4.0$

$$z_{1_3} = 0.9368 \cdot 2.0 + 0.938 = 2.811$$

$$a_{1_3} = \max(0, 2.811) = 2.811$$

$$z_{2_3} = 1.8767 \cdot 2.811 + 1.938 = 7.213$$

$$a_{2_3} = 7.213$$

$$y_{\text{pred}_3} = a_{2_3} = 7.213$$

$$L_2 = \frac{1}{2}\left(y_{\text{pred}_2} - y_{\text{true}_2}\right)^2 = \frac{1}{2}(5.3 - 2.5)^2 = 3.92$$

$$\frac{\partial L}{\partial w_2} = (7.2 - 4.0) \cdot 1 \cdot 2.811 = 9.00$$

$$\frac{\partial L}{\partial b_2} = (7.2 - 4.0) \cdot 1 \cdot 1 = 3.2$$

$$\frac{\partial L}{\partial w_1} = (7.2 - 4.0) \cdot 1 \cdot 1 \cdot 1 \cdot 2.0 = 6.4$$

$$\frac{\partial L}{\partial b_1} = (7.2 - 4.0) \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 3.2$$

$$w_1 = 0.9368 - 0.01 \cdot 6.4 = 0.8728$$

$$b_1 = 0.938 - 0.01 \cdot 3.2 = 0.906$$

$$w_2 = 1.8767 - 0.01 \cdot 9.00 = 1.7867$$

$$b_2 = 1.938 - 0.01 \cdot 3.2 = 1.906$$

Thus after a single training iteration, this is the value of weights and biases:

$$w_1 = 0.8728$$

$$b_1 = 0.906$$

$$w_2 = 1.7867$$

$$b_2 = 1.906$$

Question 2:

In a gaussian kernel when training a nonlinear SVM, the Kernel Function is defined as follows:

$$K(x_i, x) = exp(-\frac{||x_i - x||^2}{2\sigma^2})$$

$$\text{where} \quad \sigma \text{ is the parameter of the kernel}$$

Now the following classification rule is applied by a SVM to classify an unseen test sample (X) input to it:
The following function value is calculated:

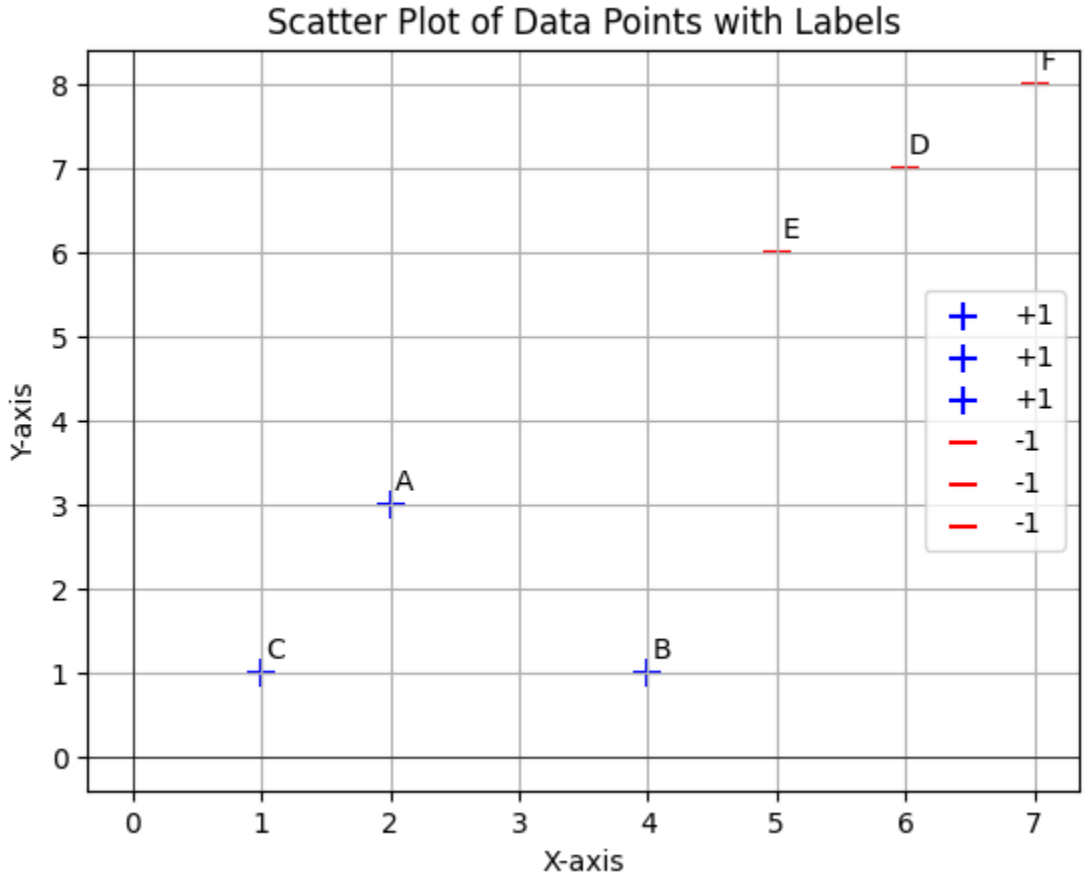
$$f(x) \quad = sign(\sum_{i=1}^{N_s} \alpha_{i,O} y^{(i)} K(x^{(i)}, x) + (1 - \sum_{i=1}^{N_s} \alpha_{i,O} y^{(i)} K(x^{(i)}, x)))$$

N is the number of support vectors
K is the Kernel function defined above
α_i is the optimal lagrangian multipliers
$y^{(i)}$ is the label for the i-th support vector
$x^{(i)}$ is the i-th support vector

Now this function's value and whether it is +1, 0 or -1, it is used to classify the point as we are using the signum function here. 0 indicates that it is on the decision boundary and +1 / -1 could indicate either class.

Question 3:

A) We plot the points on a coordinate plane below:



Thus, we can clearly see that the 6 points given are **linearly separable** as points of both classes can easily be separated by a lot of linear boundaries. We need to find an optimal linear decision boundary now using SVM concepts.

B) Now for finding the optimal margin / decision boundary we first identify that the points C, D and F cannot form linearly separating decision boundaries as evident from the diagram. Now the closest points to the decision boundary can be A (2,3), B(4,1) and E(5,6). Clearly thus these 3 points will influence the decision boundary and are going to be our support vectors. The line passing through A and B will have slope:

$$\frac{1-3}{4-2} = \frac{-2}{2} = -1$$

Thus the line will be:

$$y - 3 = (-1) \cdot (x - 2)$$

$$y - 3 = -x + 2$$
$$x + y = 5$$

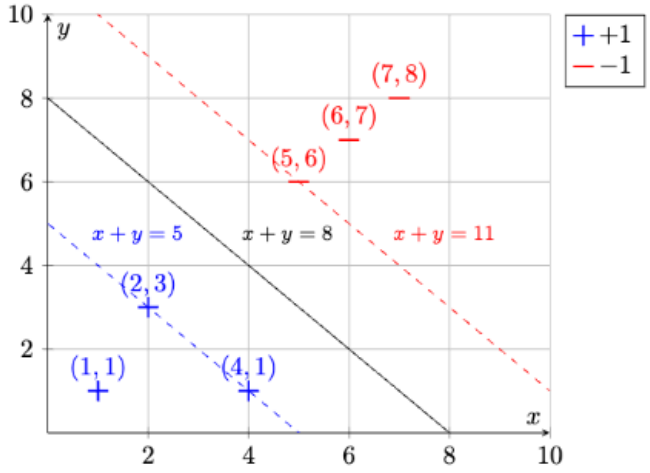
Thus the decision boundary will be parallel to this line. Now since (5,6) is also a support vector, we will find a line parallel to (x+y) = 5 passing through (5,6). That should be:

$$y - 6 = (-1) \cdot (x - 5)$$
$$y - 6 = -x + 5$$
$$x + y = 11$$

Now the decision boundary, which is optimal should be parallel to both these lines and in the center between the two. Thus the intercept of the decision boundary should be:

$$\frac{5+11}{2} = 8$$

Thus the equation of the decision boundary should be $x + y = 8$ for the given dataset. It is also depicted by the plot below:



C) Support vectors are those points which influence the decision boundary of the given dataset and thus from the part above, we identified 3 support vectors i.e. A (2,3), B(4,1) **belonging to class 1** and E(5,6) **belonging to class -1.**

D) The margin of the decision boundary is defined as the perpendicular distance between the support vector and the decision boundary. We can find that by finding the distance between two parallel lines (x+y) = 5 and (x+y) = 8 (i.e. the boundary).

Distance between two parallel lines is equal to:

$$\frac{C_1 - C_2}{\sqrt{(A^2 + B^2)}} = \frac{8 - 5}{\sqrt{(1 + 1)}}$$

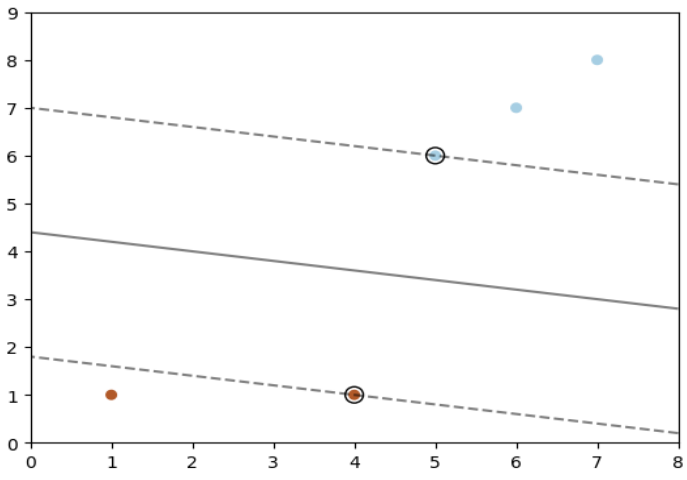
$$= \frac{3}{\sqrt{(2)}}$$

Therefore the margin of the decision boundary is **3 / sqrt(2) ~ 2.12.**

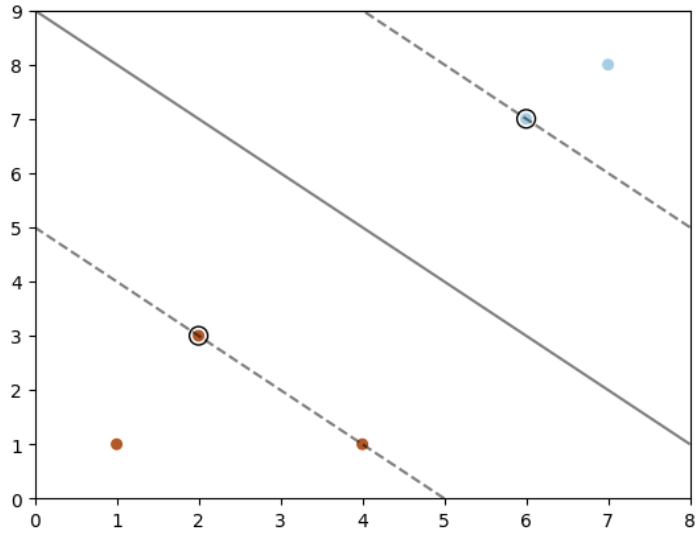
E)

We analyse case-by-case what will happen in case we remove a support vector:

i) Removing A (2,3): If point (2,3) is removed then B and E remain to be support vectors and the line is tilted more parallel to the X-axis. The new boundary and support vectors should look like this:



ii) Removing E (5,6): If we remove point E then D (6,7) becomes a support vector and the decision boundary shifts parallely one unit below as (6,7) also lies on the same line. The decision boundary is of the form x + y - c = 0



Section B:

1. **Implementing a Neural Network Class and given Weight-Initialisation and Activation Functions:**

Implemented a well-documented code in modular fashion in .ipynb file attached with the submission. Every function with parameters is explained in the code itself.

The following activation functions are implemented: **(along with their gradient functions)**

- ReLU
- Sigmoid
- Leaky ReLU
- Tanh
- Linear
- Softmax (only for the last layer)

The following weight-initialization functions are implemented: (**along with appropriate scaling)**

- Normal Init : N(0,1)
- Random Init
- Zero Init

2. **Testing on MNIST Dataset**

- I loaded the MNIST dataset from tensorflow.keras.datasets
- I performed the following **pre-processing** on the dataset:
    i) Flattening the images to 784 features
    ii) Normalising the images by dividing by 255.0

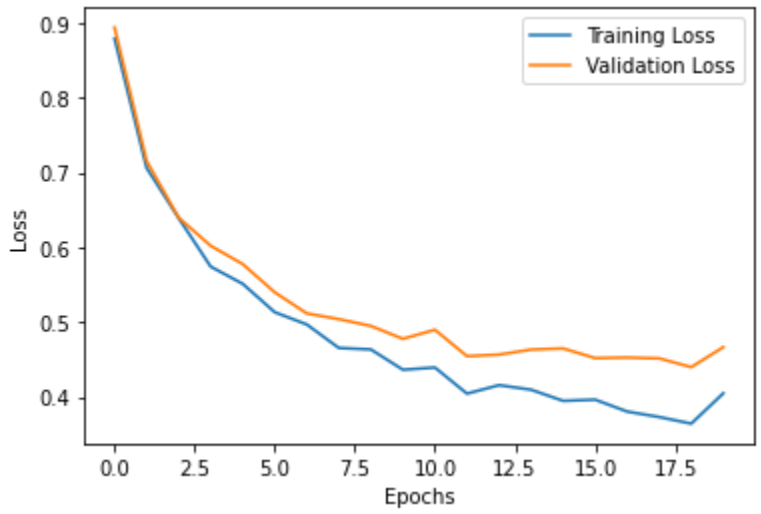
iii) One-Hot Encode the labels for my Neural Network to compute the loss at each step
iv) Perform a 80:20 split on the training dataset to create a validation split.

We then train the model on the training split with each activation function and plot the training loss and validation loss vs epochs curves. We set the following configuration of our neural network:

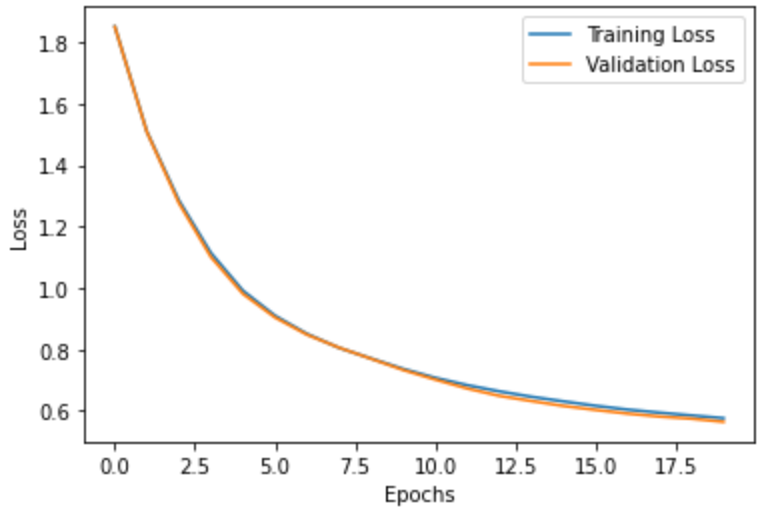
- Number of Hidden Layers (N) = 4
- Layer sizes (A) = [256,128,64,32]
- Number of epochs = 20
- Batch size = 128

Here is the curves for the same: (Note for reference all the models are trained with normal-init and stored in respective .pkl files in the submission folder)

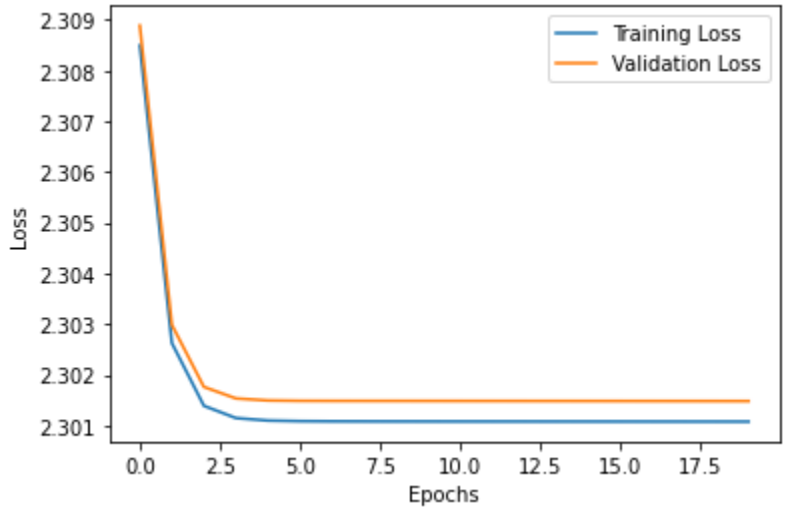
1. **Tanh Activation:**



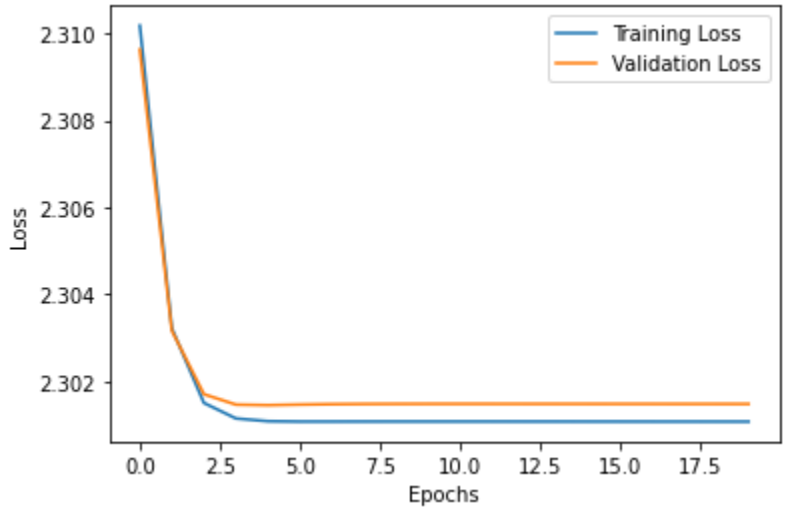
2. **Sigmoid Activation:**



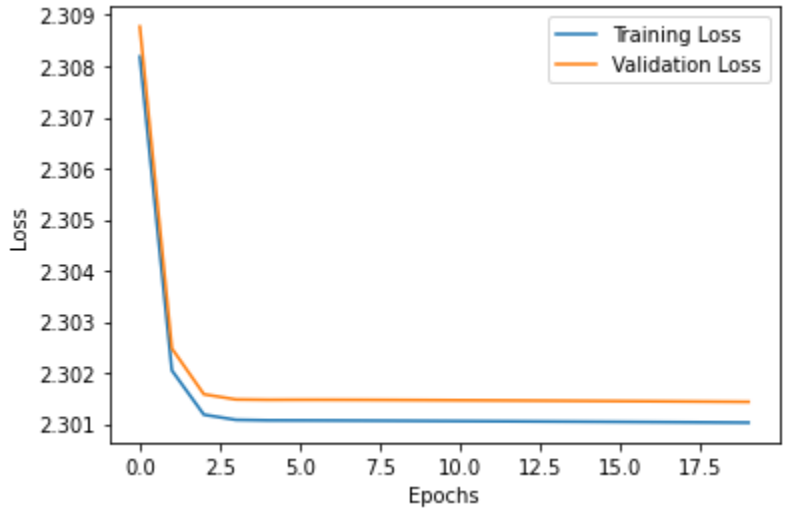
3. ReLU Activation:



4. Leaky ReLU Activation:



5. **Linear Activation:**



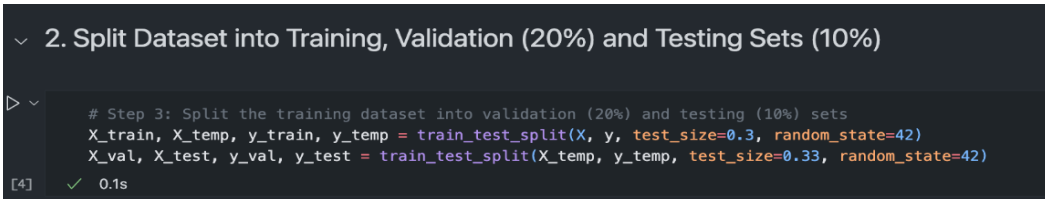
Section C:

1. Data Preparation:

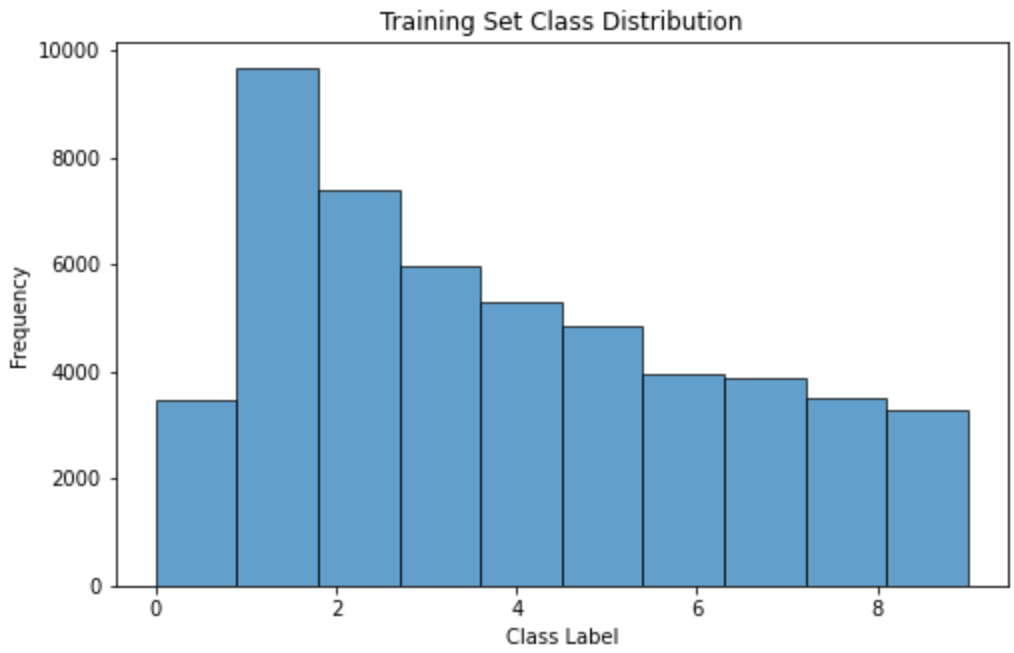
i) Downloading Data

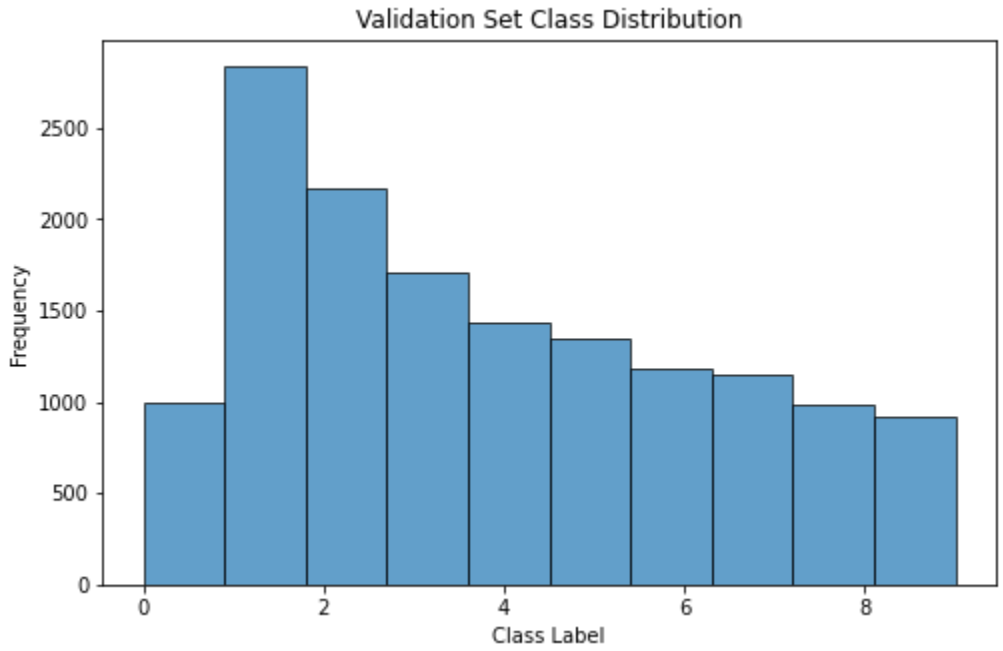
We downloaded the SHVN Dataset as a .mat file and stored it in the assignment folder. We used loadmat to load this into a numpy array of size (73257 x 32 x 32 x 3) for each image indicating an image of 32 x 32 resolution and 3 channels.

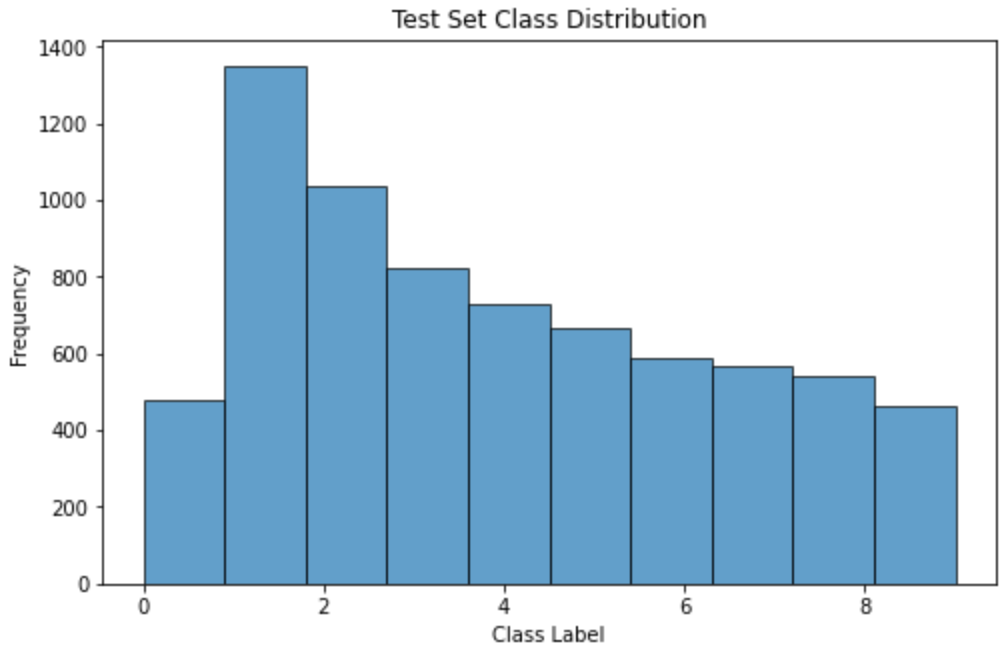
ii) Splitting the Data in the ratio asked:



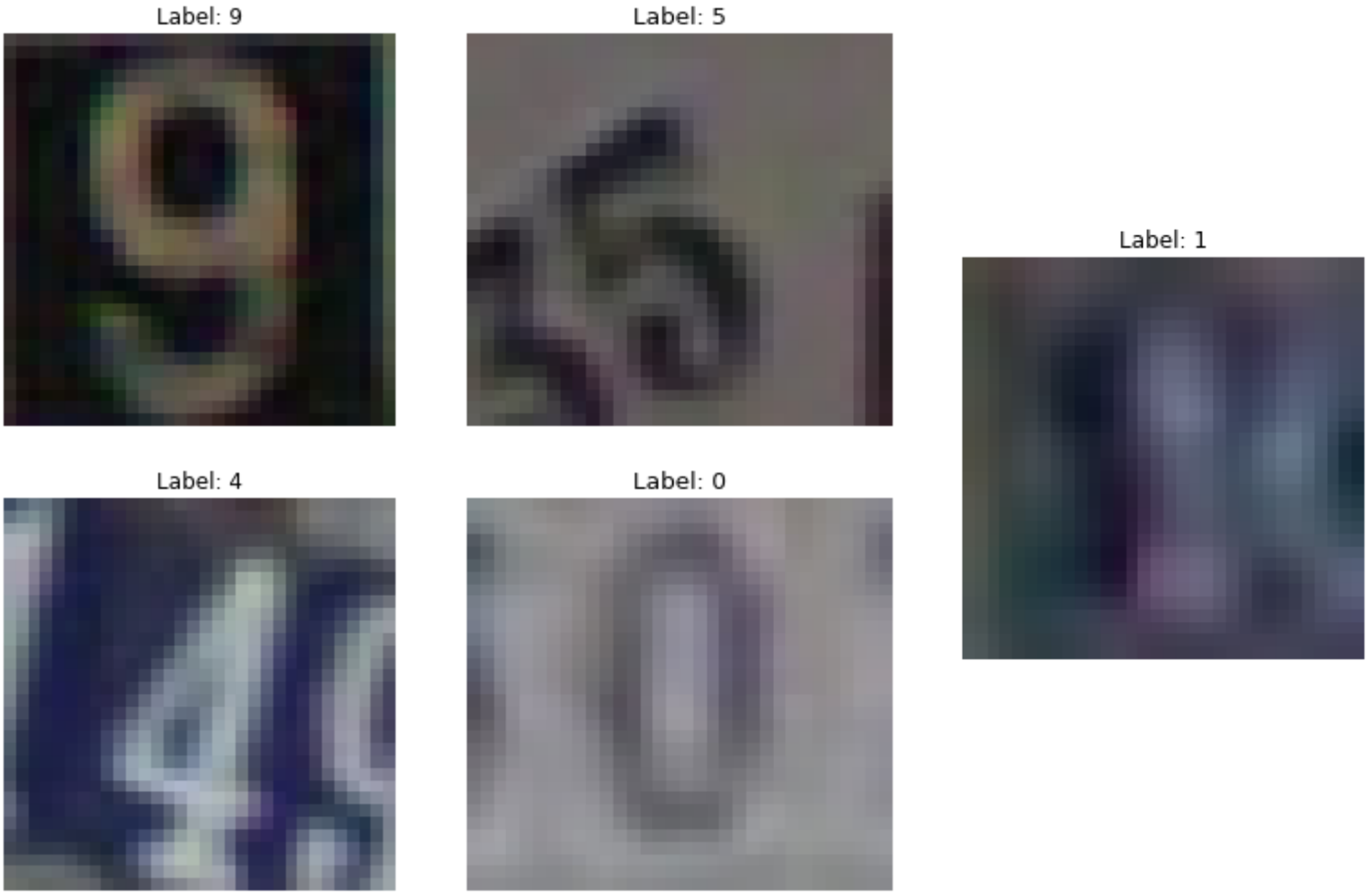
iii) Visualization of Label Distribution in each of the Splits:


Training Set Class Distribution
Frequency
Class Label
0
2000
4000
6000
8000
10000
0
2
4
6
8


Validation Set Class Distribution
Frequency
Class Label
0
500
1000
1500
2000
2500
0
2
4
6
8

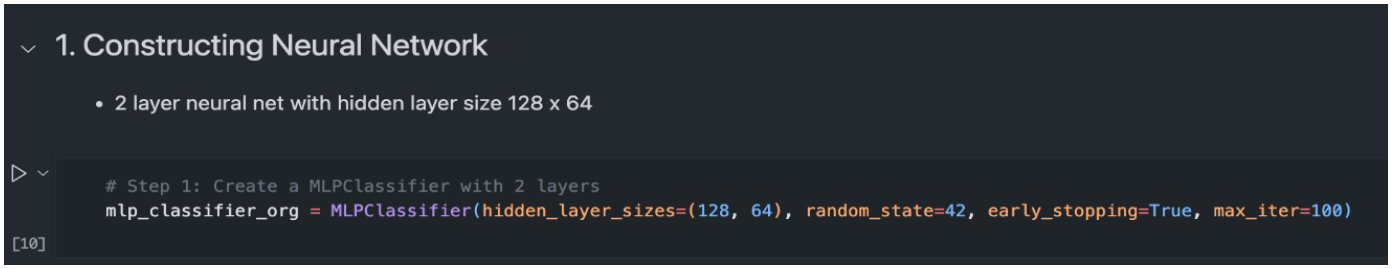


iv) Visualization of 5 unique samples from the Training Data



2. Model Training and Activation Functions:

i) Constructiong Neural Network with 2 layers:



Validation Accuracy of this Model: 81.33
Testing Accuracy of this Model: 80.78

ii) Grid-Search to find Optimal Hyperparameters:

```
param_grid = {
    'batch_size': [32, 64, 128],
    'learning_rate_init': [0.001, 0.01, 0.1]
}

# Perform GridSearchCV with 3-fold cross validation
grid_search = GridSearchCV(mlp_classifier_org, param_grid, cv=3, n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)
```

Best Hyperparams:
`{'batch_size': 128, 'learning_rate_init': 0.001}`

iii) Various Activation Functions to Train the Model:

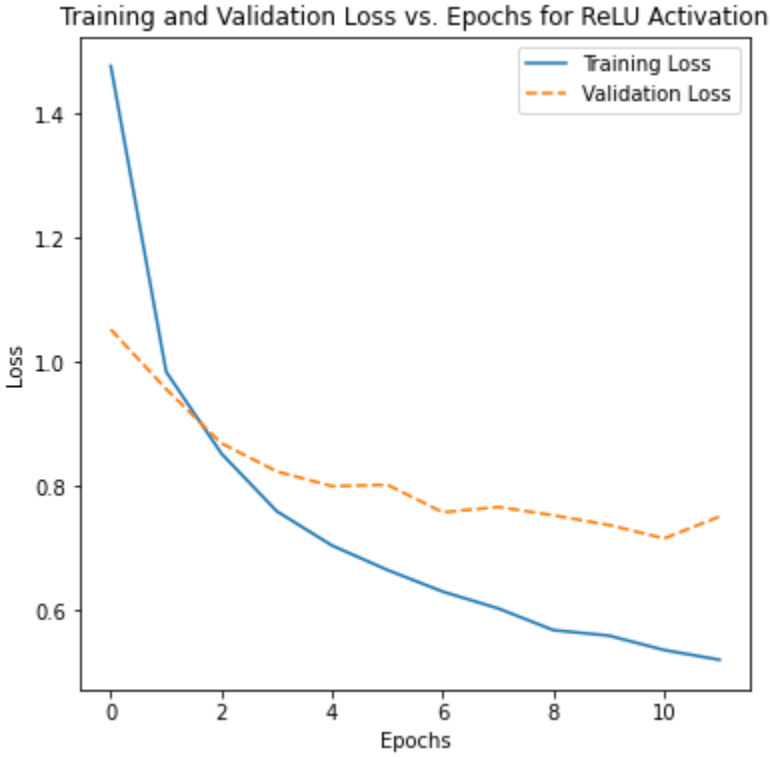
We trained the given MLP classifier with the hyperparameters for the following activation functions:

- ReLU
- Sigmoid (‘logistic’)
- Tanh
- Linear (‘identity’)

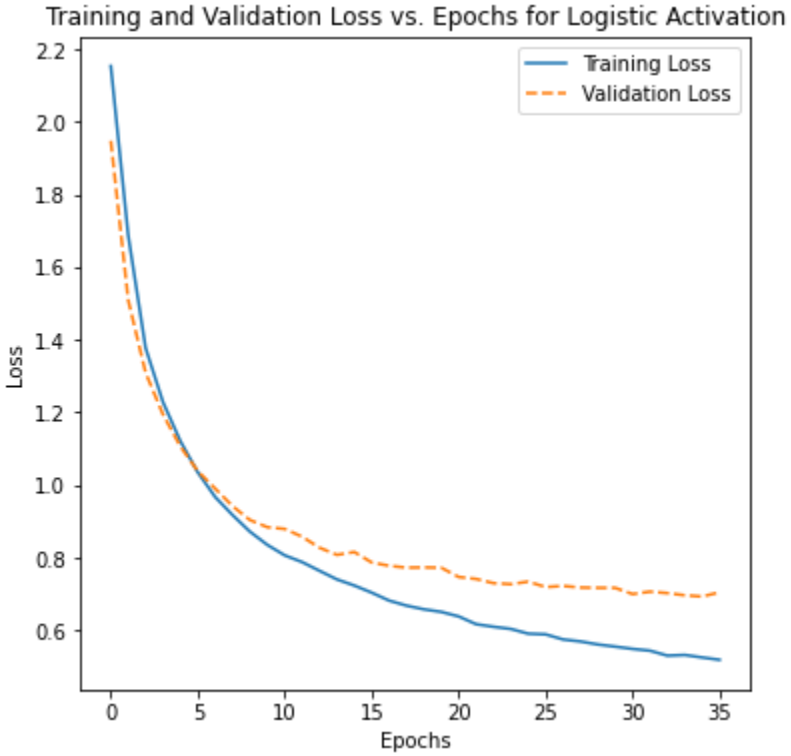
iv) Plotting the Training Loss and Validation Loss vs Epochs for each:

We choose 100 epochs and plotted the two losses. Since validation losses are not returned, we adopted the method of partially fitting the model for each iteration, sending in the validation data and calculating the validation loss. We also implemented early stopping manually for this as partial fit does not support early stopping hyperparameter.

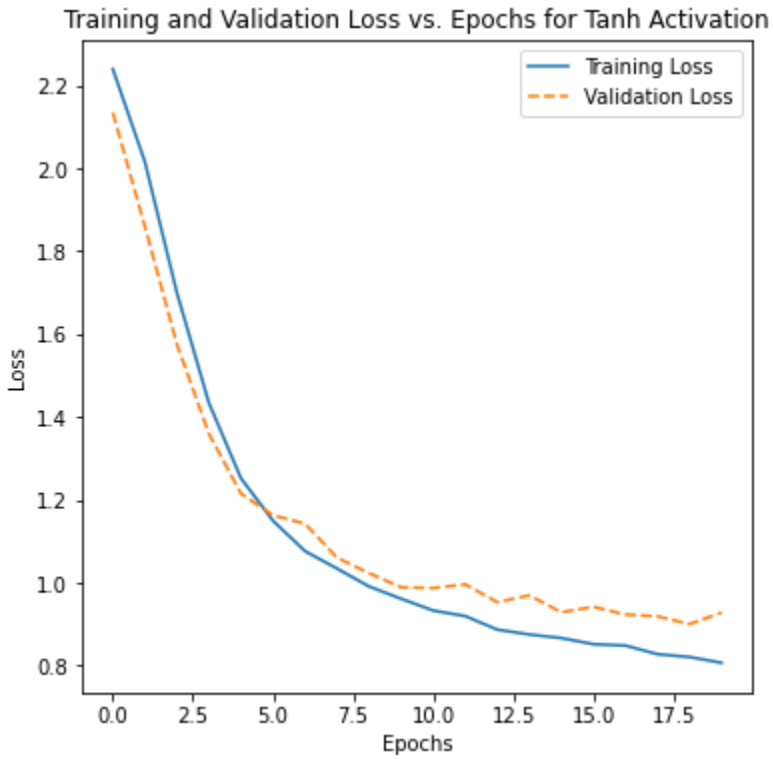
ReLU:



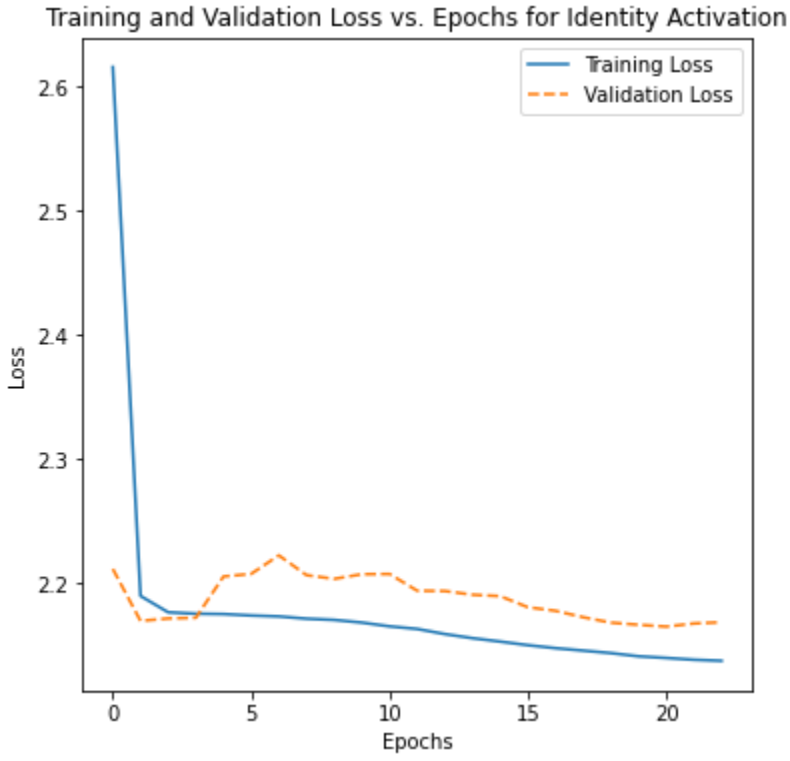
Sigmoid:



Tanh:


Training and Validation Loss vs. Epochs for Tanh Activation
Training Loss
Validation Loss
Loss
Epochs
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Linear:


Training and Validation Loss vs. Epochs for Identity Activation
Training Loss
Validation Loss
Loss
Epochs
2.6
2.5
2.4
2.3
2.2
0
5
10
15
20

v) Analysis of Effective Learning of the Model

We can see from the plots that ReLU, Sigmoid and Tanh almost similarly learn the training data well. The ReLU function is able to reduce the loss very well and sigmoid, tanh have very smooth curves because of having smooth gradients.

However linear activation function is very bad at reducing the loss. After a initial decrease, it plateaus very quickly and the loss stays significantly above the losses in ReLU, Sigmoid and Tanh which means that it is not very well in learning. This is because it keeps our network **linear** and linear decision boundaries are not affective in classification always. Other 3 activation functions introduce non-linearity in the network and thus are much better at learning the patterns in the training data.

This is also validated by the testing accuracies of the 3 models as shown below:

```
Test Accuracy of the model with ReLU activation:  0.7865710740383289
Test Accuracy of the model with Logistic activation:  0.7762305252998759
Test Accuracy of the model with Tanh activation:  0.7067420377774714
Test Accuracy of the model with Identity activation:  0.2523093892182545
```

vi) Explanation of Hyperparameters:
As seen above, **ReLU** achieves the best accuracy on the test set (~78%) which is a very good accuracy noting that we have a basic 2 layered network with no advanced image processing techniques or regularisation. **Sigmoid, Tanh** also achieve 70+ accuracy on the testing data.

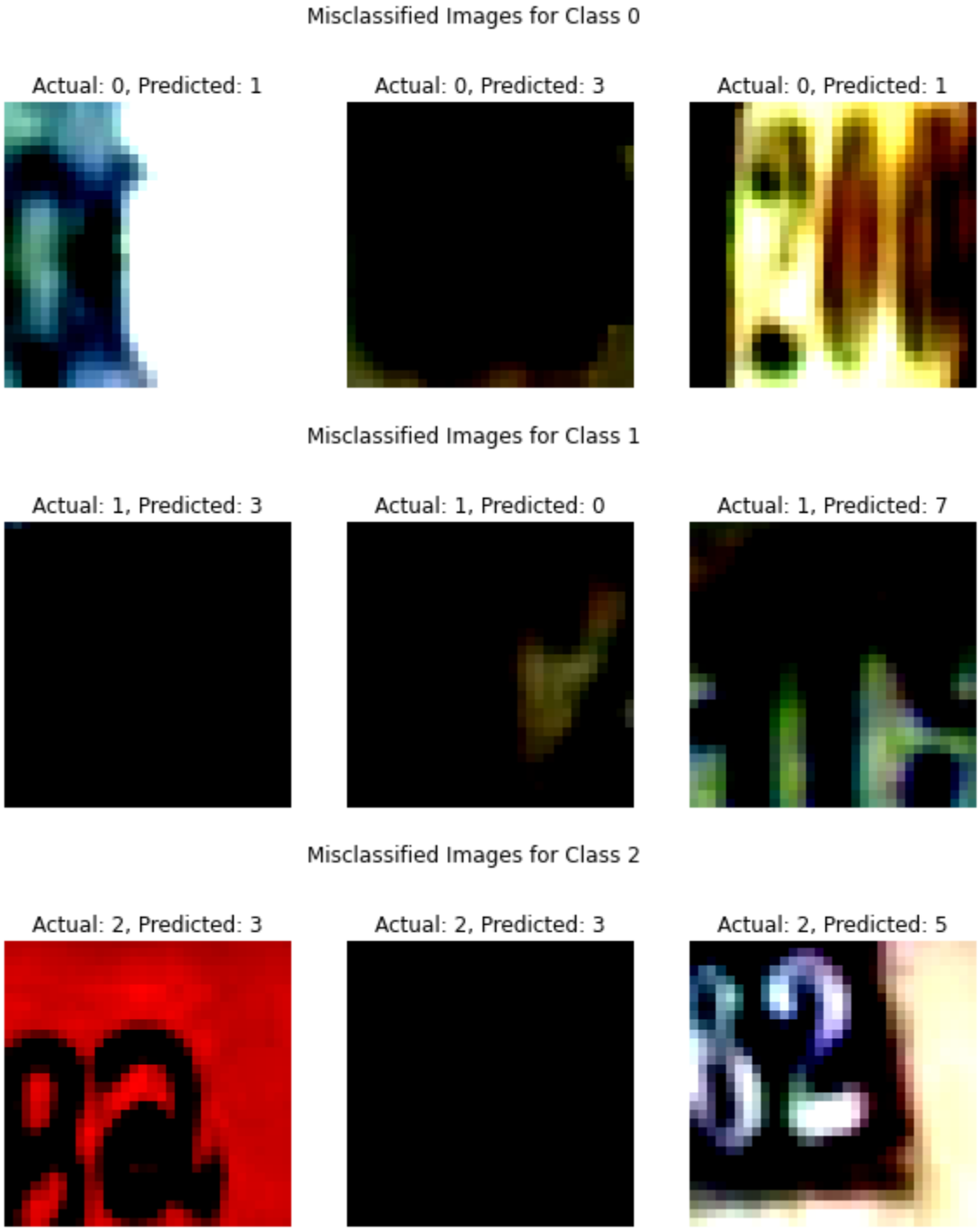
We found initial hyperparameters about layer size, early stopping and random_state through trial and error. Remaining hyperparameters were found using 3-fold cross validation based Grid Search. Thus our best model with its configuration is given below:

```
MLPClassifier(hidden_layer_sizes=(128, 64),
              random_state=42,
              batch_size = 128,
              learning_rate_init = 0.001,
              activation="relu",
              max_iter=100)
```

3. Visualisation of Incorrect Predictions:

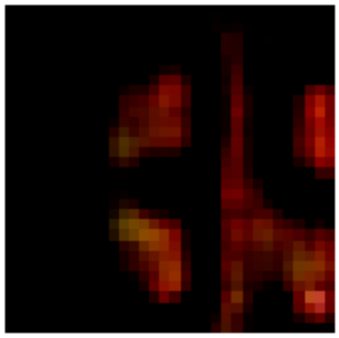
We use our best model i.e. ReLU based model for the tasks in this section:

i) Visualing 3 Misclassified Images for Each Class:



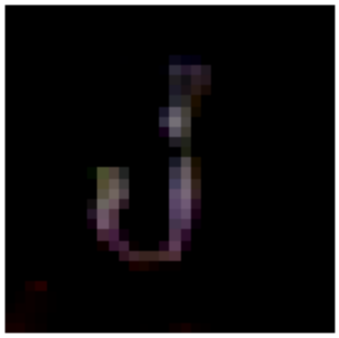
Misclassified Images for Class 3

Actual: 3, Predicted: 9



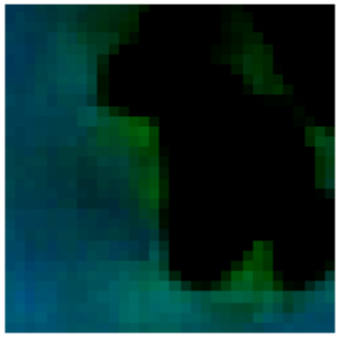
Actual: 3, Predicted: 5

Actual: 3, Predicted: 1

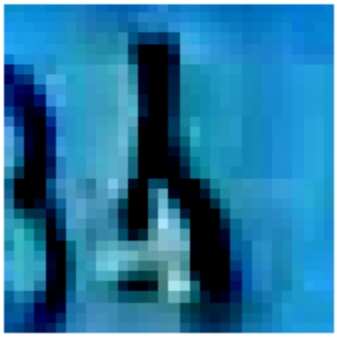


Misclassified Images for Class 4

Actual: 4, Predicted: 1



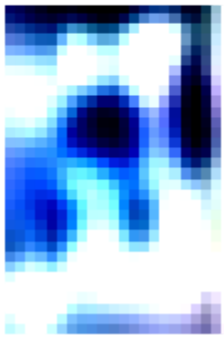
Actual: 4, Predicted: 6



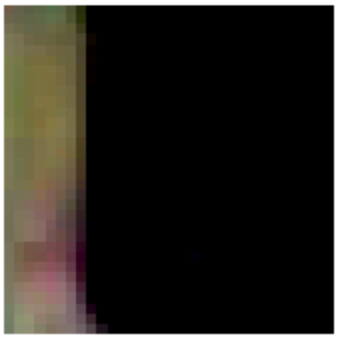
Actual: 4, Predicted: 1

Misclassified Images for Class 5

Actual: 5, Predicted: 3



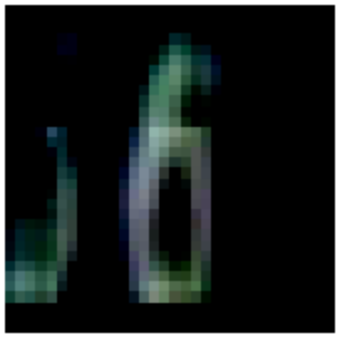
Actual: 5, Predicted: 3



Actual: 5, Predicted: 3

Misclassified Images for Class 6

Actual: 6, Predicted: 8

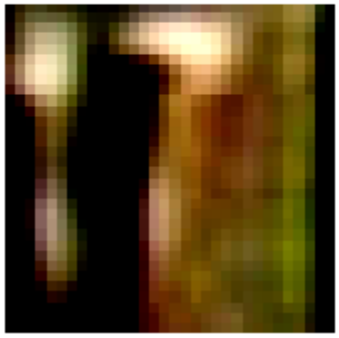


Actual: 6, Predicted: 5

Actual: 6, Predicted: 7

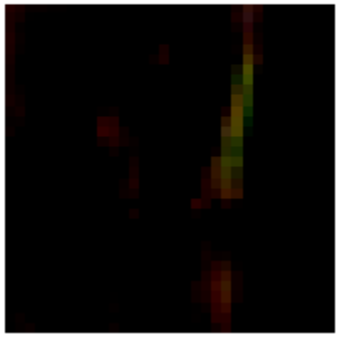
Misclassified Images for Class 7

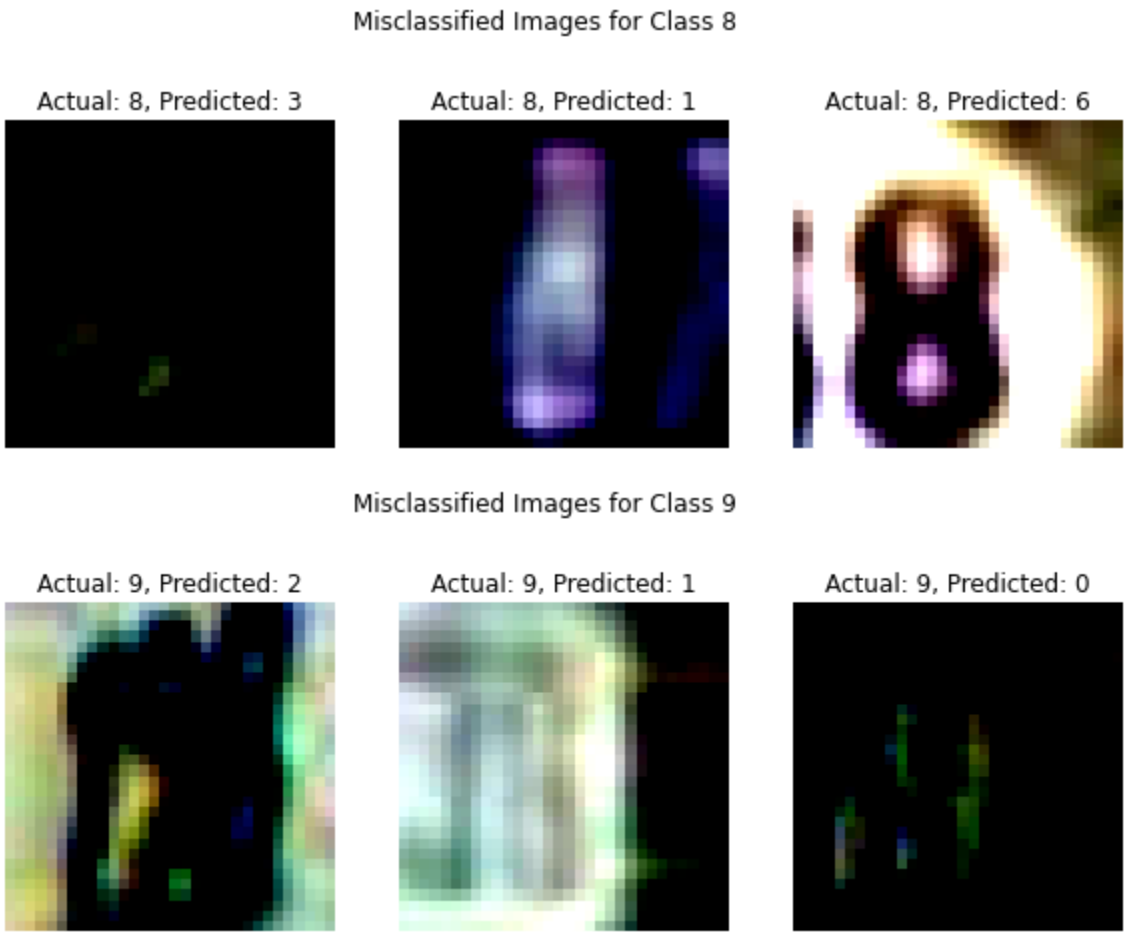
Actual: 7, Predicted: 0



Actual: 7, Predicted: 3

Actual: 7, Predicted: 1





ii) Analysing Reasons for Misclassification:

As we can see some images which have one label attached to them also have another number in some proportion in that image. This is a severe noise component in the dataset and thus confuses the trained model severely.

Additionally, a lot of the images have noise in the form of dark colours or the images are not clearing which is hard even for the human eye to classify. Hence the model is misclassifying them as it is identifying an incorrect pattern.

Additionally, our MLP classifier did not really have any techniques implemented for preventing a network and thus there is a chance that the model overfit on the training data and is misclassifying these images in the testing set.