CS577 Assignment 1: Sample Report

Shivi Srivastava
Department Of Computer Science
Illinois Institute of Technology
February 16, 2021

This report has the details of the 4 Assignment programs

Problem 1-

1. **Problem Statement**
   On tensorflow playground webpage,train a classifier to classify all the four examples and keep the classifier simple.
   This will give the idea how to tune the hyperparameters based on the output shape we intend to get.
2. **Proposed Solution**
   For each of the diagram we're going to make some changes in the learning rate, batch size, number of neurons in a layer and number of layers itself. If required we will be changing the activation function also.
3. **Implementation details**
   for the **first diagram** change the below hyperparameters-
   -Provided 2 hidden layers
   -First layer having three neurons second layer having two neurons
   -Batch size decreased to 6

   For the **second diagram** change the below hyperparameters to check weather the model can predict with low neurons and less batch size
   - removed one neuron from the second layer and reduced the batch size to 4,this did not work so change the activation to relu, but that also did not work.
   -finally gave the batch size as 10 and added 2 neurons to 1st hidden layer and with any activation as relu.

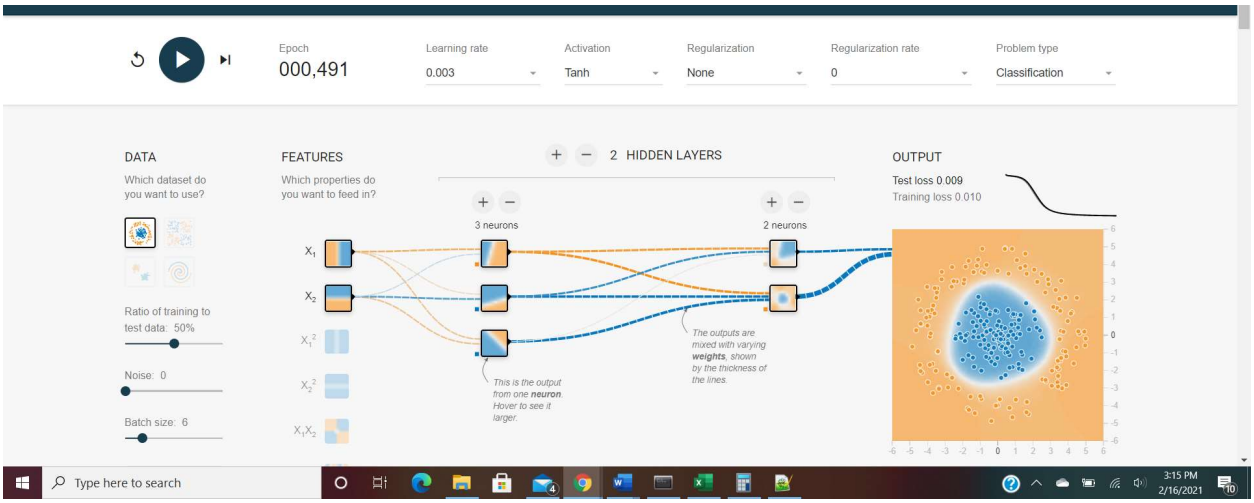
   The **third diagram**
   -changed the batch size 10
   - added three neurons to the 1st hidden layer and 1 neuron in the 2nd hidden layer.
   - activation function to Tanh.

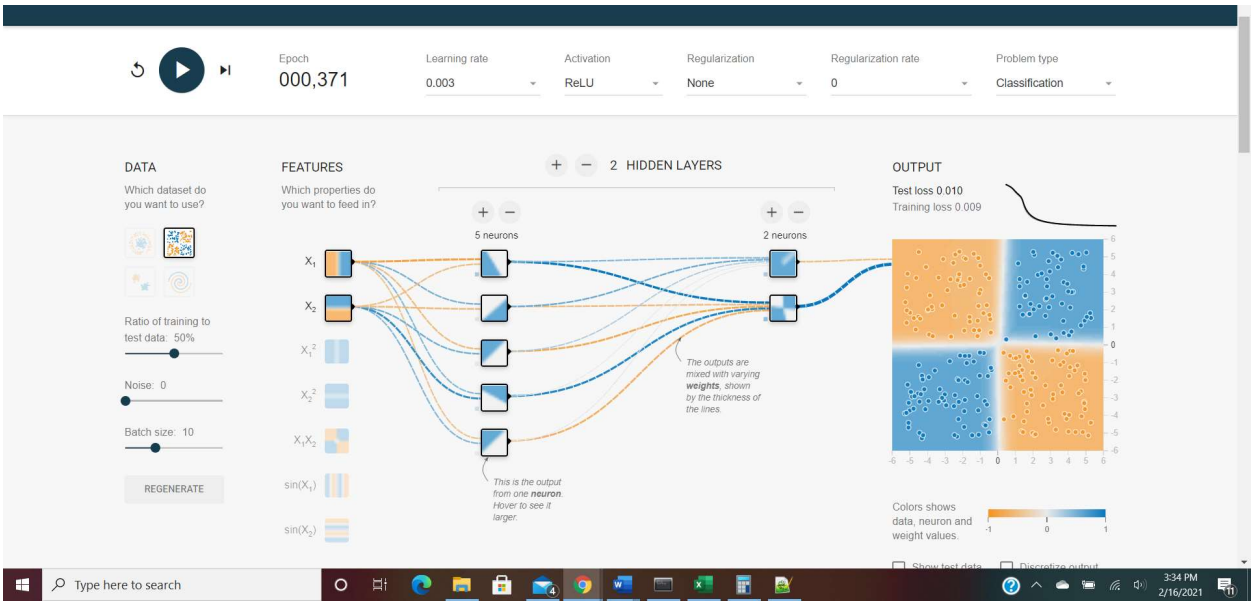
   for the **fourth diagram**
   -increased the number of hidden layers
   -changed the learning rate to 0.01
   - added 5 neurons to the 1st layer 3 neurons to the 2nd layer and 2 neurons to the last layer . This did not help.

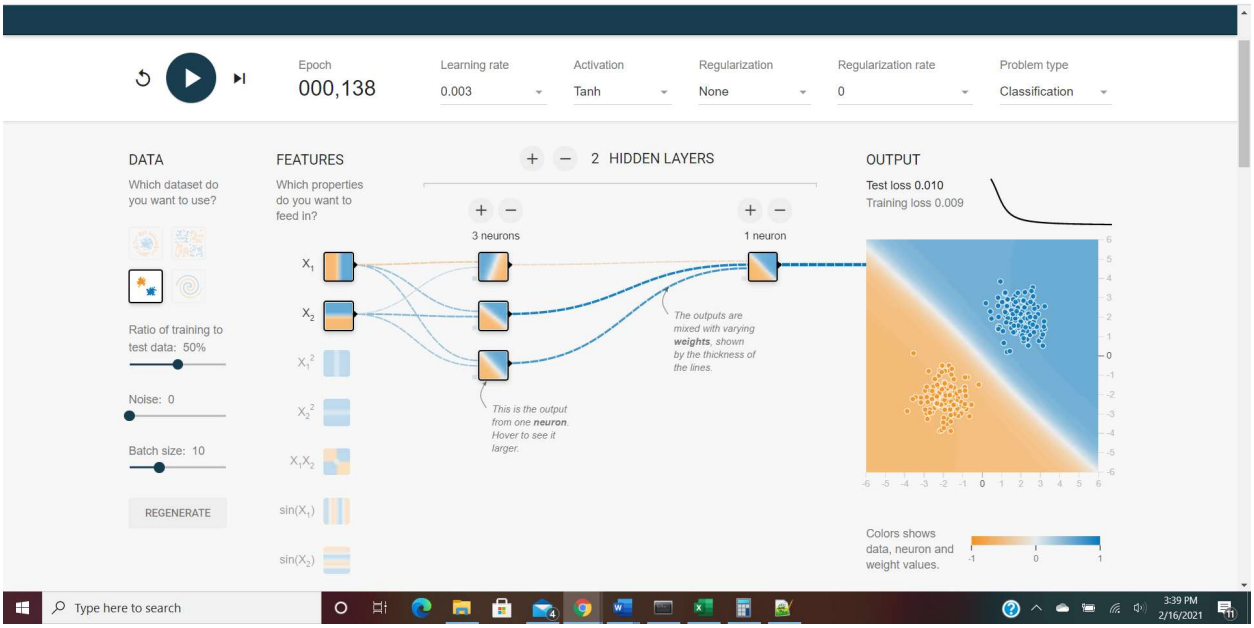
So changed the hyperparameters as below,

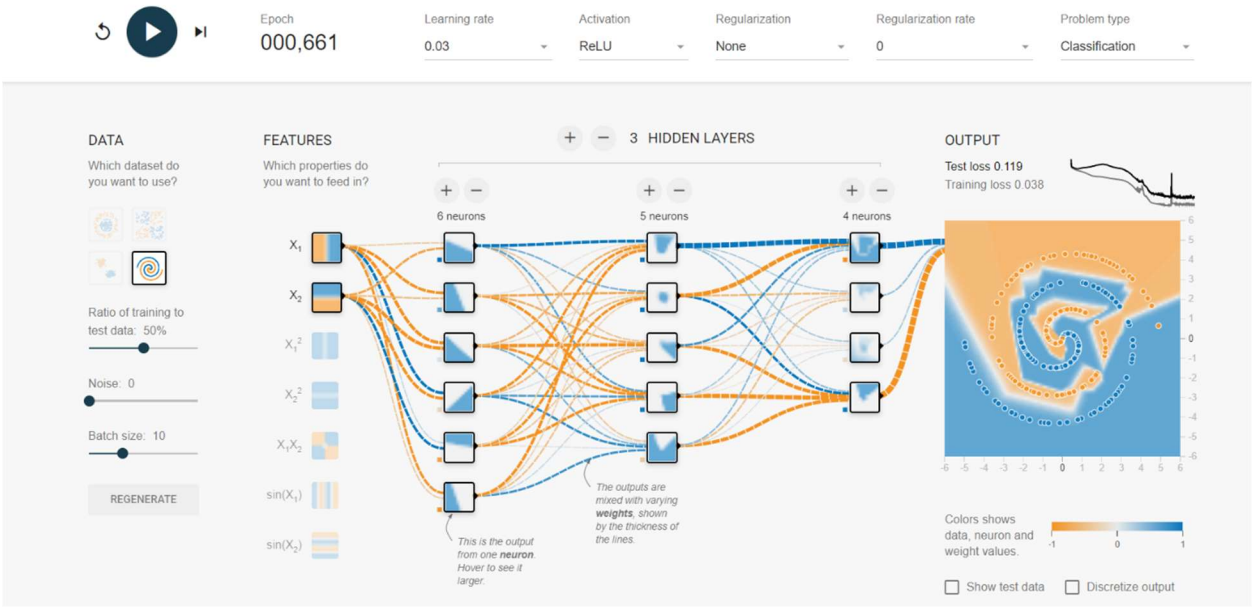
- added 6 neurons to the 1st layer 5 neurons to the 2nd layer and 4 neurons to the last layer.
- Changed activation function as relu
- Learning rate to 0.03, with 661 epochs.

4. Results and discussions









General observation was, changing the learning rate and number of neurons in the layer, helps get the desired result.

Sigmoid function was not useful here.

A large batch size also does not help in getting the proper model.

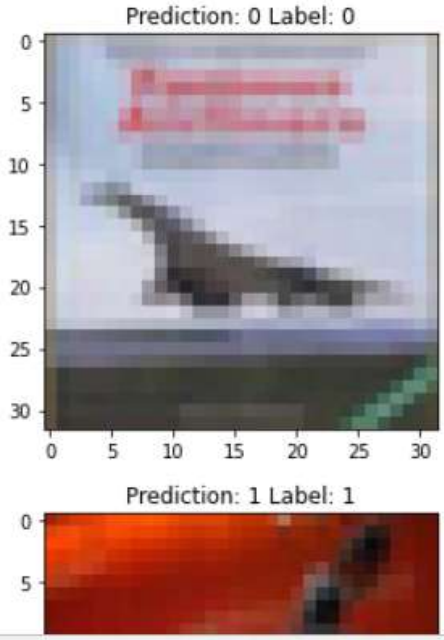
Problem 2 –

1. **Problem statement**

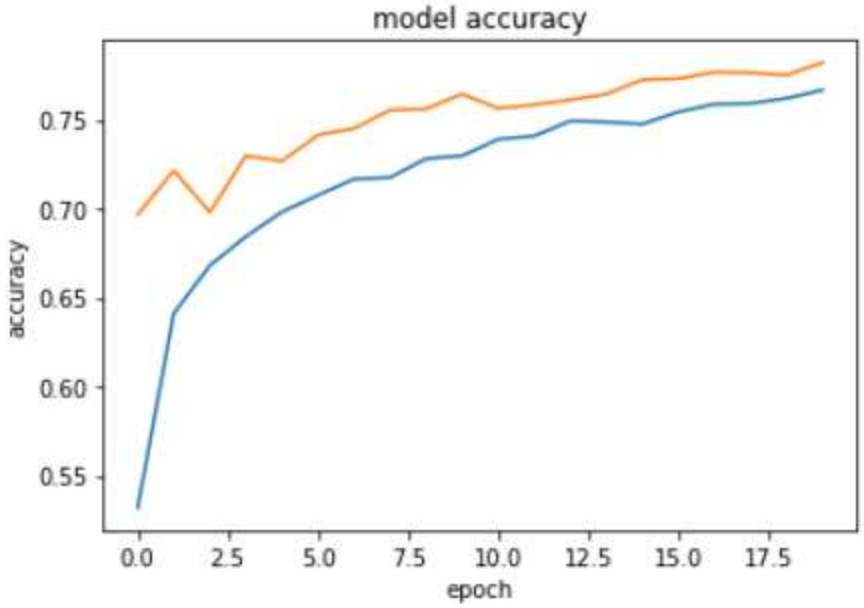
   This problem is for CIFAR10 where we need to get the subset of it as three classes and prepare a model which can identify the images within these three classes by using the training validation and testing set.

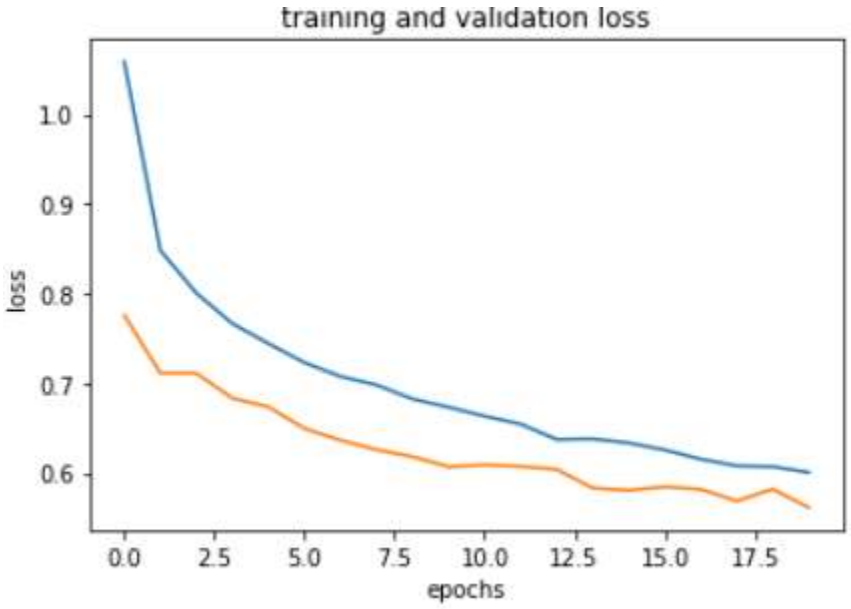
2. **Proposed solution**
   - data in cypher 10 is divided into 10 classes each class having 5000 training images and 1000 testing images so for our solution we will have total 15,000 training set plus validation set and 3000 testing images.
   - We will start by downloading the data from the source and preprocessing it, making it in a format which can then easily be fed to the neural network.
   - Will use softmax as the activation function and categorical_crossentropy as the loss function.
   - We will see which optimizer we are going to use as we tune the model.
   - Also activation function for the middle layers will be changed as we train the model.
   - As this is the fully connected neural network using only the dense layers.
3. **Implementation details**
   - downloading the necessary libraries like tensorflow, keras, matplotlib, numpy etc.
   - Downloading the data
   - taking 3 class subset of the data
   - We will reshape the images into the 1D arrays of 3072 pixels that we expect, and the label data into one-hot-encoded categorical format
   - Then we will normalize the text images by dividing by 255 so that they fall between 0 and 1.
   - we will divide the training data and two validation data and training data.
   - Sequential model is used with four 10 dense layers it is mandatory to give the put to the first player which will be 3072.
   - The middle layer has activation function linear, as relu was bringing down the accuracy.
   - For compiling the model we used Adagrad as the other optimizers were either giving NAN or not giving a good accuracy rate. The learning rate is set to 0.003.
4. **Results and Discussions**
   - Testing gave 78% accuracy, and it took less than 1 sec to run for 3000 dataset.
   - Below code snippet will give the result of what was predicted and actual.

```
for x in range(20):
    test_image = test_images[x,:].reshape(1,32*32*3)
    predicted_cat = model.predict(test_image).argmax()
    label = new_test_labels[x].argmax()
    if (predicted_cat == label):
        plt.title('Prediction: %d Label: %d' % (predicted_cat, label))
        plt.imshow(test_image.reshape([32,32,3]))
        plt.show()
```



Below is the graph for Model Accuracy and Loss for 20 epochs-





The major problem occurred do identify how many layers are needed and which activation function to use for the middle layers to get good validation accuracy.

Problem 3-

1. **Problem statement**

   The purpose of this model is to identify the spam mails, for this we are going to use this Spambase data set. This data set has 58 features the last column of this data set contains 1 or 0 this indicates whether the email is spam or not. Total number of data is 4601 which we will be dividing into training, validation and testing set.

2. **Proposed solution-**
   - We will use the pandas library to read the CSV file and make any changes to the dataframe.
   - Load_spam_data is the method used for downloading, cleaning, normalizing and distributing the data
   - Downloading the data using wget so that we don't have to maintain the CSV file externally.
   - Dividing the data as below

     ```
     training set (3500, 57)
     validation set (500, 57)
     test set (601, 57)
     ```
   - We will start with three dense layers and work on adding the units and activation function for it.
   - Will use sigmoid activation function for output layer, as output can be 0 or 1 and binary_crossentropy as the loss function.
   - We will do the normalization of data using mean normalization.
3. **Implementation details**
   - Load_ spam_ data method is used to download the data set.
   - Any missing values as '?' is replaced by NAN, to be replaced by zero or mean if required.

- Normalization is done using min max, as mean normalization was giving the negative values.
- It is most common to use 32 bit precision when training a neural network, so converted to 32 bit floats from 64 bit.
- Used three dense layer with hundred units for the first layer and 50 units for the second layer with activation function as relu
- Compiled the model with rmsprop optimizer and binary cross entropy as loss function. As a result got a very lower accuracy rate nearly 12%, tried increasing the learning rate but that did not help.
- Reduced the dense layer to 2, changed the optimizer to Adam and got a really good accuracy rate of 95% with validation data.

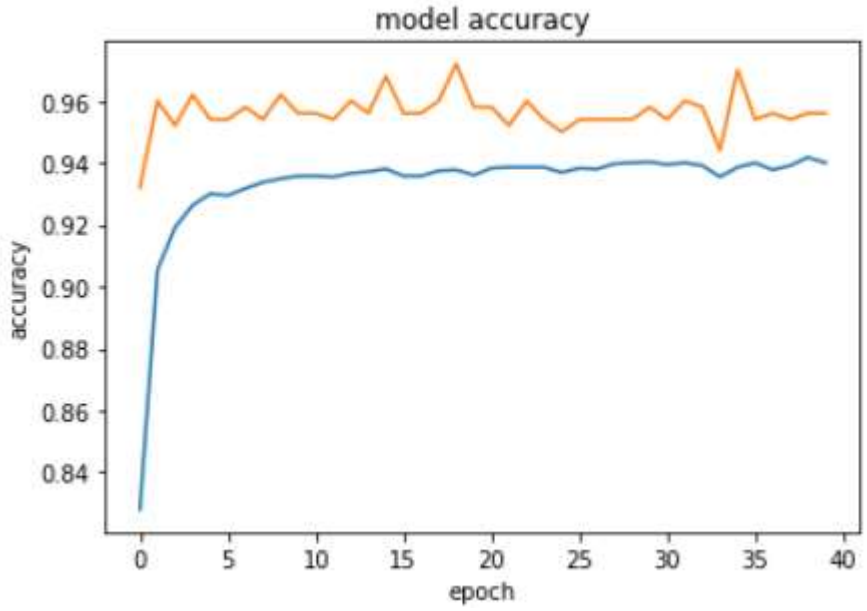
5. Results and Discussions

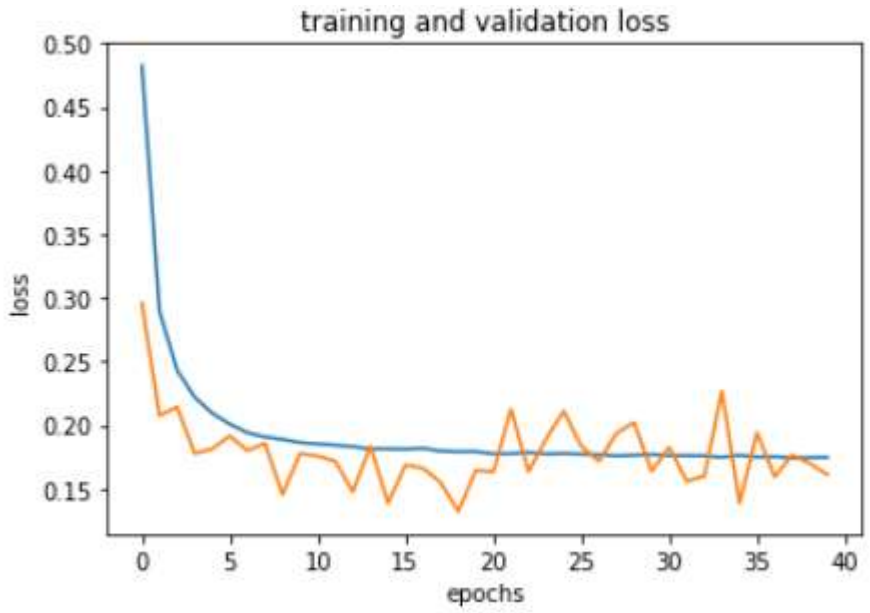
- The model is slightly overfitting, but changing the number of layers, learning rate etc did not help.
- 1 ms to process 410 elements with accuracy rate of 77%-78%

```
[195] test_loss, test_accuracy = model.evaluate(x_test, y_test)

      19/19 [==============================] - 0s 1ms/step - loss: 2.9157 - binary_accuracy: 0.7770
```

- Used 40 epochs





- The testing loss is still huge, tuning the hyperparameter is making 2-3 % difference.

Problem 3-

1. **Problem statement-**
   Given the communities and crime dataset. The goal is to predict ViolentCrimesPerPop.
   This is a regression problem.
   We will use k fold cross validation for this problem

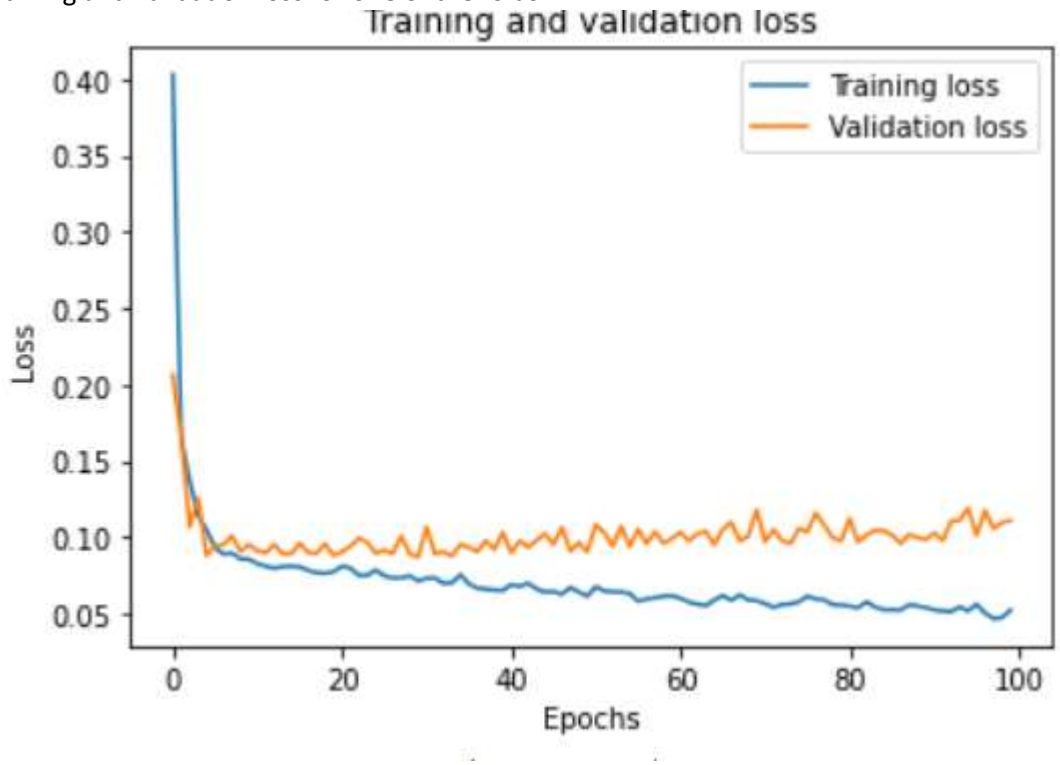
2. **Proposed solution-**
   - We will use the pandas library to read the CSV file and make any changes to the dataframe.
   - Load_crime_data is the method used for downloading, cleaning, normalizing and distributing the data
   - Downloading the data using wget so that we don't have to maintain the CSV file externally.
   - We will be using a combination of relu and linear activation functions on the dense layers.
   - We will use the loss function as MeanAbsoluteError and metrics as mse
   - K fold value is 10

3. **Implementation details**
   - Load_ crime_ data method is used to download the data set.
   - Any missing values as '?' is replaced by NAN, to be replaced by zero or mean if required.
   - Normalization is done using min max
   - Any missing value is replaced by 0
   - We are using a sequence model inside k -fold and compiling the model for each fold.
   - 100 epochs was taken to compile and run the validation and got the average validation error around 0.096.
   - The final model is validated on the entire non-testing set and performance is evaluated by testing set to be 0.097.

- Observed the average validation error and testing error are similar.

4. **Results and Discussions-**

Training and validation loss for one of the folds-



Average Validation Loss-

```
avg_validation_loss:  0.09477133378386497
```

Test loss-

```
(testloss,testaccuracy) = model.evaluate(x_test,y_test)
```

```
32/32 [==============================] - 0s 1ms/step - loss: 0.0974 - mae: 0.0974
```

Test loss vs epoch