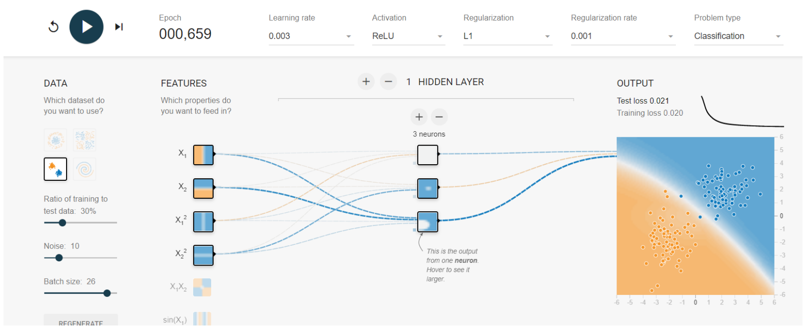
Assignment Report

Syed Alle Mustafa

September 21, 2022

1. TASK 1

   All the neural network present on the tensor playground where trained with minimum testing and validation loss. The configuration and results of each model can be seen in the figures below.

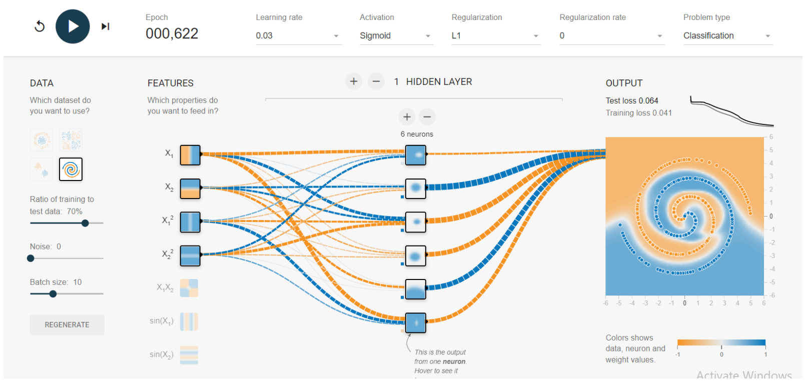

DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
This is the output from one neuron. Hover to see it larger.
REGENERATE

Epoch 000,659 | Learning rate 0.003 | Activation ReLU | Regularization L1 | Regularization rate 0.001 | Problem type Classification

Ratio of training to test data: 30% | Noise: 10 | Batch size: 26

1 HIDDEN LAYER | 3 neurons

OUTPUT: Test loss 0.021, Training loss 0.020

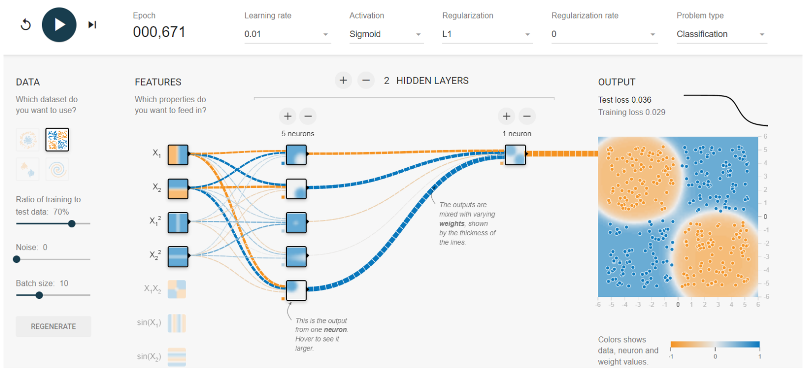

DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
This is the output from one neuron. Hover to see it
REGENERATE
Colors shows data, neuron and weight values.
Activate Windows

Epoch 000,622 | Learning rate 0.03 | Activation Sigmoid | Regularization L1 | Regularization rate 0 | Problem type Classification

Ratio of training to test data: 70% | Noise: 0 | Batch size: 10

1 HIDDEN LAYER | 6 neurons

OUTPUT: Test loss 0.064, Training loss 0.041

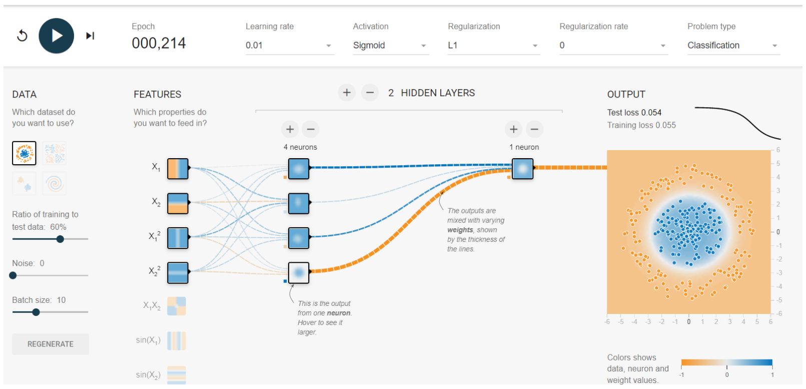

DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
REGENERATE
Colors shows data, neuron and weight values.

Epoch 000,671 | Learning rate 0.01 | Activation Sigmoid | Regularization L1 | Regularization rate 0 | Problem type Classification

Ratio of training to test data: 70% | Noise: 0 | Batch size: 10

2 HIDDEN LAYERS | 5 neurons | 1 neuron

OUTPUT: Test loss 0.036, Training loss 0.029



Epoch 000,214

Learning rate 0.01

Activation Sigmoid

Regularization L1

Regularization rate 0

Problem type Classification

Ratio of training to test data: 60%

Noise: 0

Batch size: 10

Test loss 0.054

Training loss 0.055

The outputs are mixed with varying **weights**, shown by the thickness of the lines.

This is the output from one **neuron**. Hover to see it larger.

Colors shows data, neuron and weight values.

2. TASK 2

   (a) Problem Statement:

      i. Load the CIFAR dataset and select a subset of three classes.

      ii. Split the training set into training and validation subset.

      iii. Vectorize the images and encode the know class labels using categorical encoding.

      iv. Design a fully connected neural network to perform multi-class classification of this data Justify your network design decisions (number of hidden layers, number of units per layer, loss function, and evaluation metric) and compile the network designed. training and validation loss as a function of epochs, plot training and validation accuracy.

      v. Tune model hyper — parameters to improve performance. Retrain the final mode and test performance on the test collection.

      vi. Report the performance obtained.

      vii. Save and load the weights of the trained network

   (b) Proposed Solution:

   The feature part of the dataset was images and the target or the data to be predicted was label. So After cleansing, normalizing and loading and partitioning dataset, a sequential model was used with multiple of Conv2d, Batch Normalization, Activation, and Dense Layers were used to achieve the best performance form the given data set. More details are discussed in the implementation part.

   (c) Implementation Details:

      i. Loaded CIFAR10 dataset into training and testing set.

      ii. Selecting initial three classes due to the requirement of the problem statement.

      iii. Normalizing train and test feature vectors (images), by dividing them by 255.

      iv. Splitting train dataset into 2 parts, one for training and one for validation.

      v. initializing a sequential model for training.

      vi. Add a conv2D layer with activation function relu with input shape of 32,3,3.

      vii. Using dropout to avoid overfitting.

      viii. Using batch normalizing for making neural network faster by normalizing input layers.

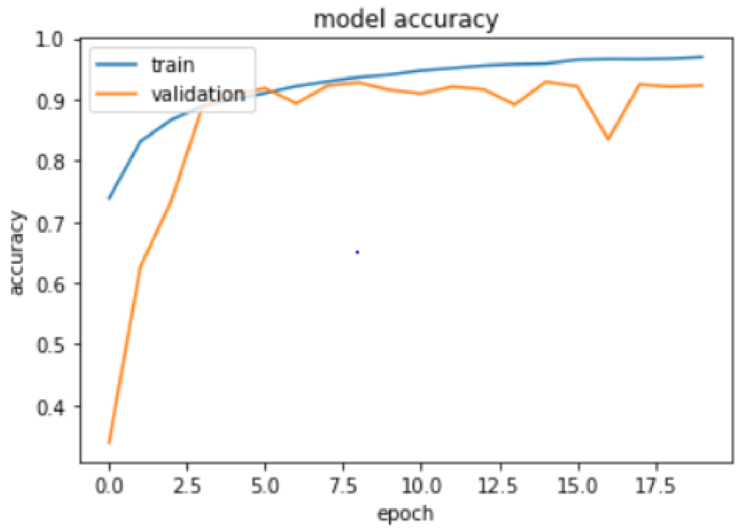
      ix. reusing this architecture three times.

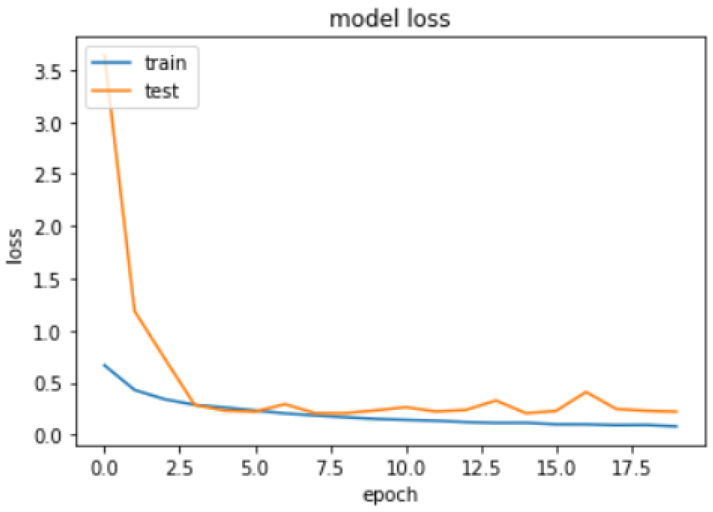
      x. Adding a dense layer twice with relu as an activation function, and batch normalization and dropout.

      xi. using the last layer as dense and using softmax as an activation function.

xii. Result of Model is quite impressive and we could achieve minimum loss for both training and validation sets.

(d) Results: The result of training and validation is as follows in the graph, the result shows that the accuracy and loss of both training and validation set is good.





3. TASK 3

(a) Problem Statement:

    i. Load the spam email data from UCI repository ”spambase”
    ii. Prepare the data you load as tensor suitable for neural network.
    iii. Normalize the features as needed.
    iv. Repeat the steps for Task 1.

(b) Proposed Solution:

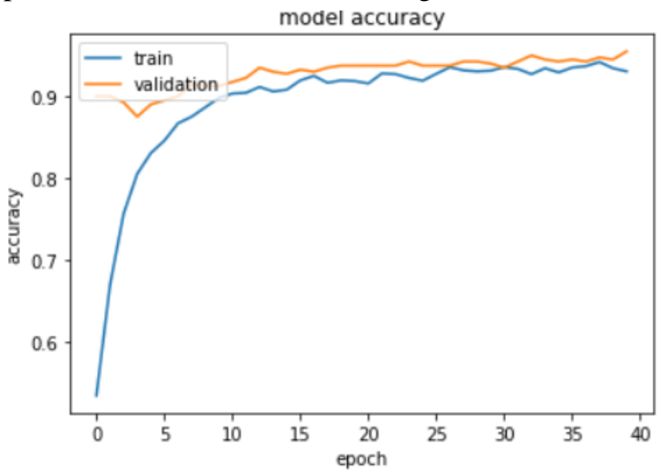
The feature part of the dataset was images and the target or the data to be predicted was label. So After cleansing, normalizing and loading and partitioning dataset, a sequential model was used with multiple dense layers with relu and sigmoid as activation functions. More details are in the implementation part.

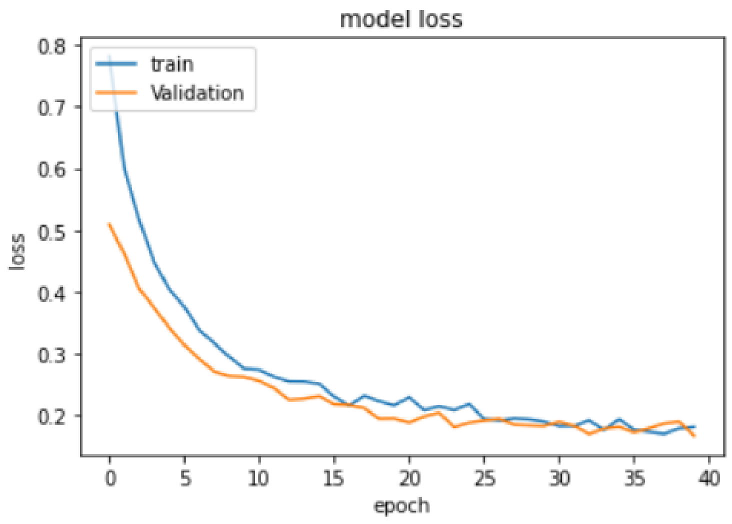
(c) Implementation Details:

    i. Loaded the spam data and partitioned it into all spam and all ham set. and then splitted them into training and testing sets and combined both sets. By doing so, we can make sure that both training and testing sets have equal spam and ham emails.
    ii. We then normalize the features by subtracting mean of the data and dividing it by standard deviation.
    iii. split the training data into training and validations set.
    iv. initializing a sequential model for training.
    v. add a dense layer to the model to with relu function.
    vi. Using dropout to avoid overfitting.
    vii. Using batch normalizing for making neural network faster by normalizing input layers.
    viii. reusing this architecture two times.
    ix. using the last layer as dense and using sigmoid as an activation function.
    x. Previewing the graph to measure the accuracy which seems pretty fine.
    xi. save the model.

(d) Result:

result of both the validation set during the training and testing set of the final model are quite impressive, it can be found in the figures below.





```
True Positive(TP)  =  662
False Positive(FP) =  57
True Negative(TN)  =  1059
False Negative(FN) =  64
Accuracy Of The Model =  93.43105320304016
```

4. TASK 4

   (a) Problem Statement:

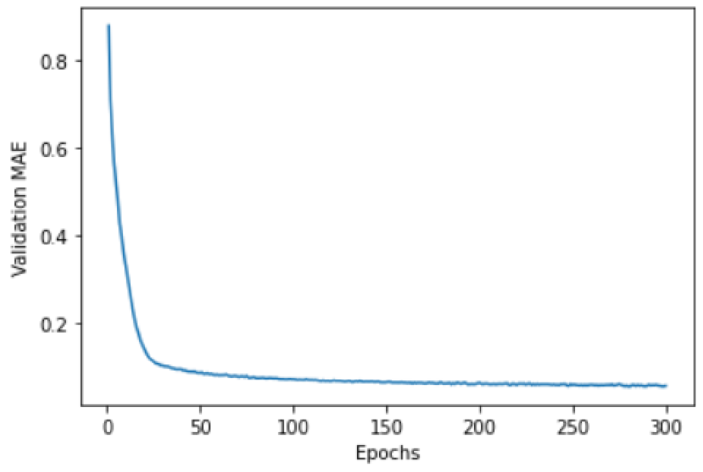
      i. Load the spam crime data from repository ”Communities and Crime”
      ii. Prepare the data you load as tensor suitable for neural network.
      iii. Normalize the features as needed.
      iv. Repeat the steps for Task 1.
      v. Perform K-Fold Cross Validation

   (b) Implementation Details:

      i. Import the crime data through a function, this function removes some unwanted columns and replaces ? with mean, and does one hot encoding of some columns as well. Lastly it returned the dataset into 4 parts, training features, testing features, training labels, testing labels.
      ii. We then normalize the features by subtracting mean of the data and dividing it by standard deviation.
      iii. initializing a sequential model for training.
      iv. add a dense layer to the model to with relu function.
      v. Using dropout to avoid overfitting.
      vi. Using batch normalizing for making neural network faster by normalizing input layers.
      vii. reusing this architecture two times.
      viii. using the last layer as dense and using sigmoid as an activation function.
      ix. Then we use k-fold cross validation to verify the parameter and datasets size. We split data into k=7 folds. and for each fold we make new validation and training set. fit the model and record the performance of each fold.
      x. After applying k-fold we look at the models performance at each fold through a graph.
      xi. After analyzing the graph the main training is done on whole training data set.
      xii. Then the performance is tried on the test data which seems pretty good.

5. Results: Following are the Results of K-fold cross validation. It shows that even the dataset isnt huge, we could achieve low loss and better accuracy.

```
Fold  0
6/6 [==============================] - 0s 3ms/step - loss: 0.0239 - mae: 0.1049
0.10492009669542313
Fold  1
6/6 [==============================] - 0s 3ms/step - loss: 0.0294 - mae: 0.1138
0.1137658879160881
Fold  2
6/6 [==============================] - 0s 3ms/step - loss: 0.0170 - mae: 0.0886
0.08860623836517334
Fold  3
6/6 [==============================] - 0s 3ms/step - loss: 0.0205 - mae: 0.0993
0.09932903945446014
Fold  4
6/6 [==============================] - 0s 3ms/step - loss: 0.0232 - mae: 0.1037
0.10368524491786957
Fold  5
6/6 [==============================] - 0s 3ms/step - loss: 0.0244 - mae: 0.0995
0.0994618609547615
Fold  6
6/6 [==============================] - 0s 3ms/step - loss: 0.0319 - mae: 0.1145
0.11451437324285507
```


Validation MAE
Epochs