Zafar Mahmood
Report – 9

Question 1 : **Logistics Regression On Iris Data Set**

**1 ) Load the Iris dataset, randomly split it into training set 90% and test set 10%.**

- Split the data using 135 , 15 as train and test

2 )

- Matrix multiplication of placeholder, with weights and bias
- multiplication with y to get the output loss over the training dataset

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

which allows us to interpret the outputs as probabilities, while cross-entropy loss is what we use to measure the error at a softmax layer, and is given by[1]

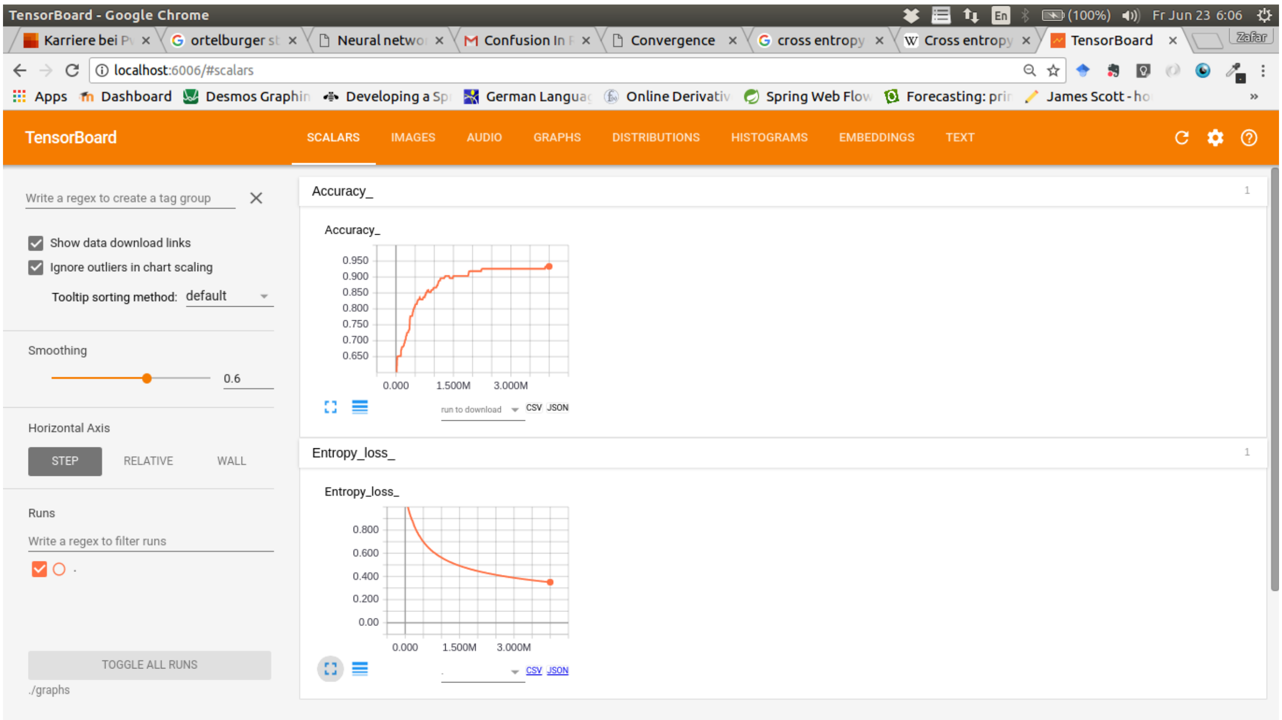
$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N} \left[ y_n \log \hat{y}_n + (1 - y_n)\log(1 - \hat{y}_n) \right],$$

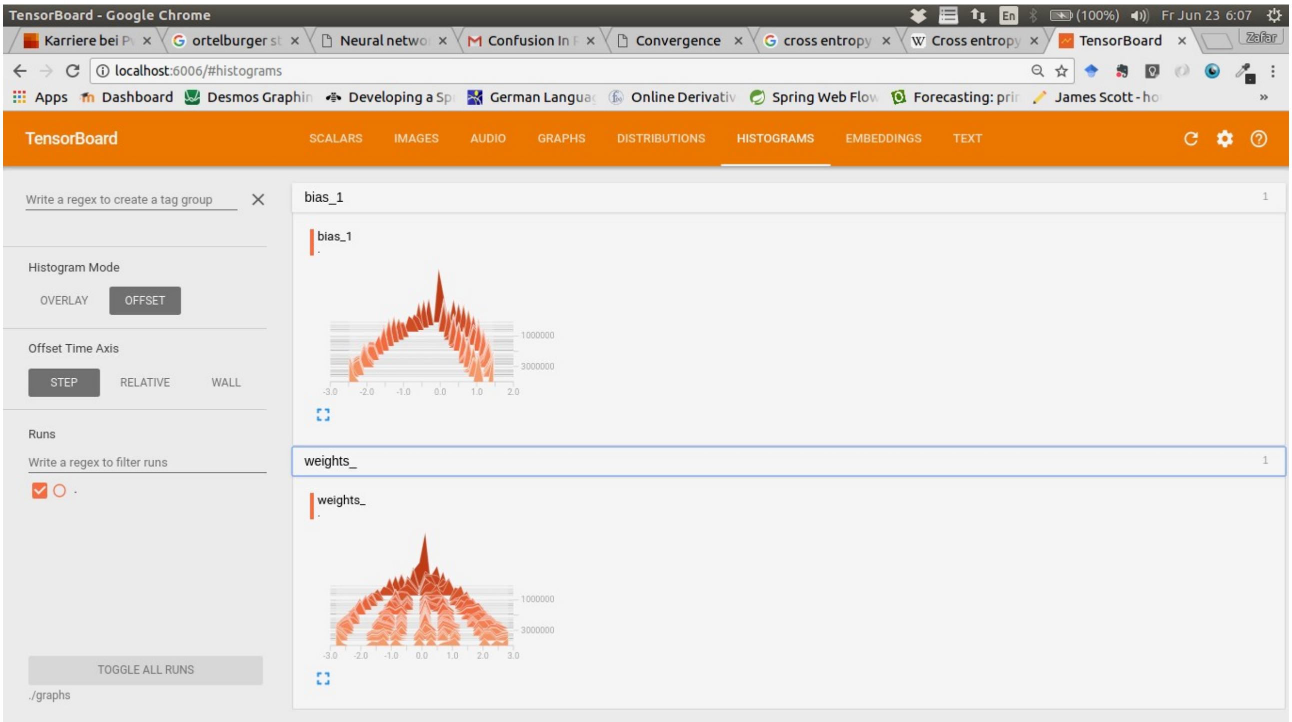
```
y = tf.nn.softmax(tf.matmul(x, W) + b)

cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
```

**3) Train the model on the training set and make prediction on the test set**

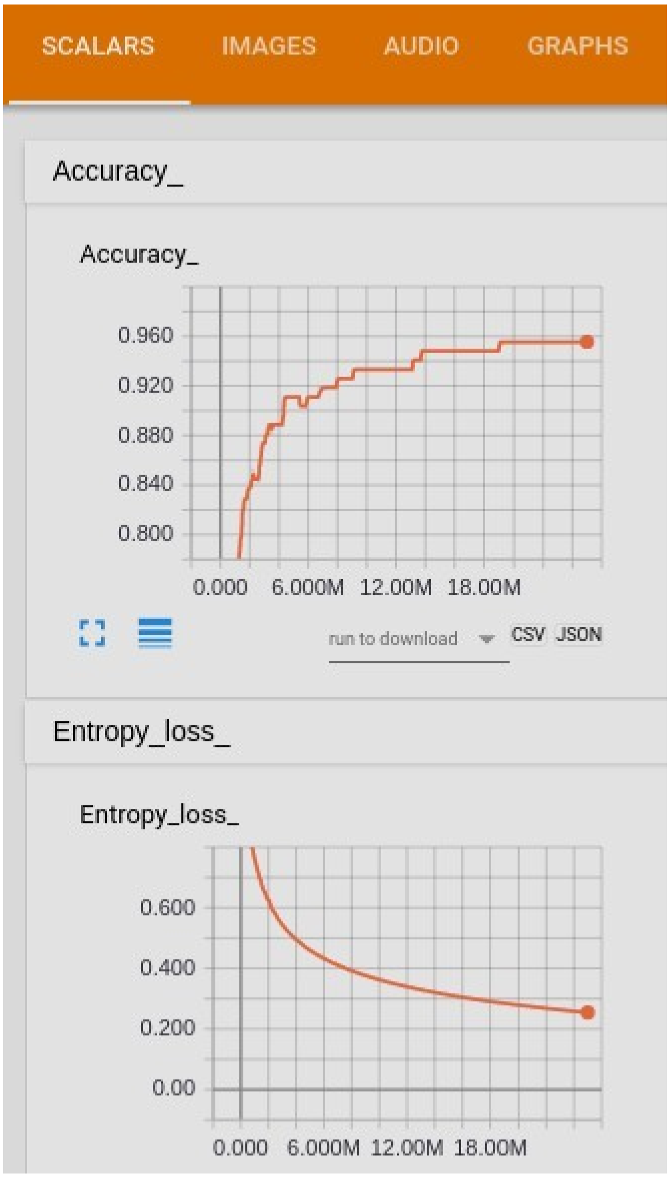
- training the model using two optimizer function
  - AdamOptimizer
  - GradientDecentOptimizer
- Leanining rate 0.05
- The accuracy that I received
  - using the gradient decent It didn’t converged although I randomly shuffled, normalized the dataset, but still the training error was not satisfactory so is the case with test set, received 93 % accuracy , if I try to increase the learning rate the it gave little better accuracy But if i run this Gradient Decent using 5000 epochs then it gives 100 % accuracy
  - using the AdamOptimizser it converged and received 100% accuracy ( lower training error and 100% accuracy )

## Gradient Decent Optimizer

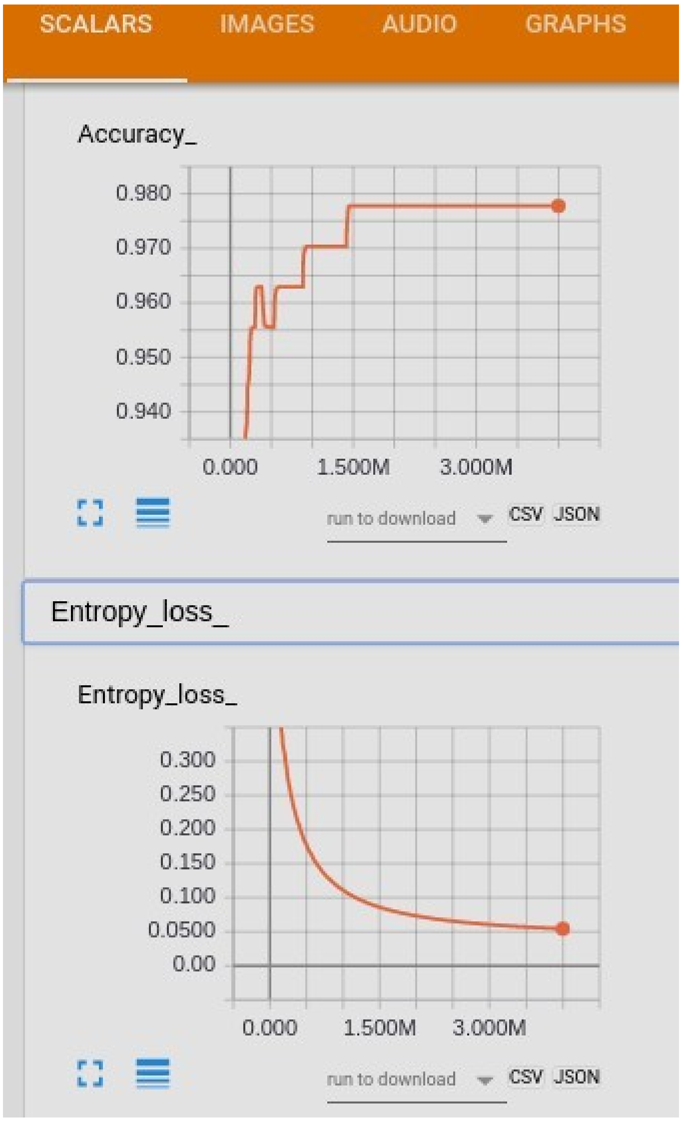


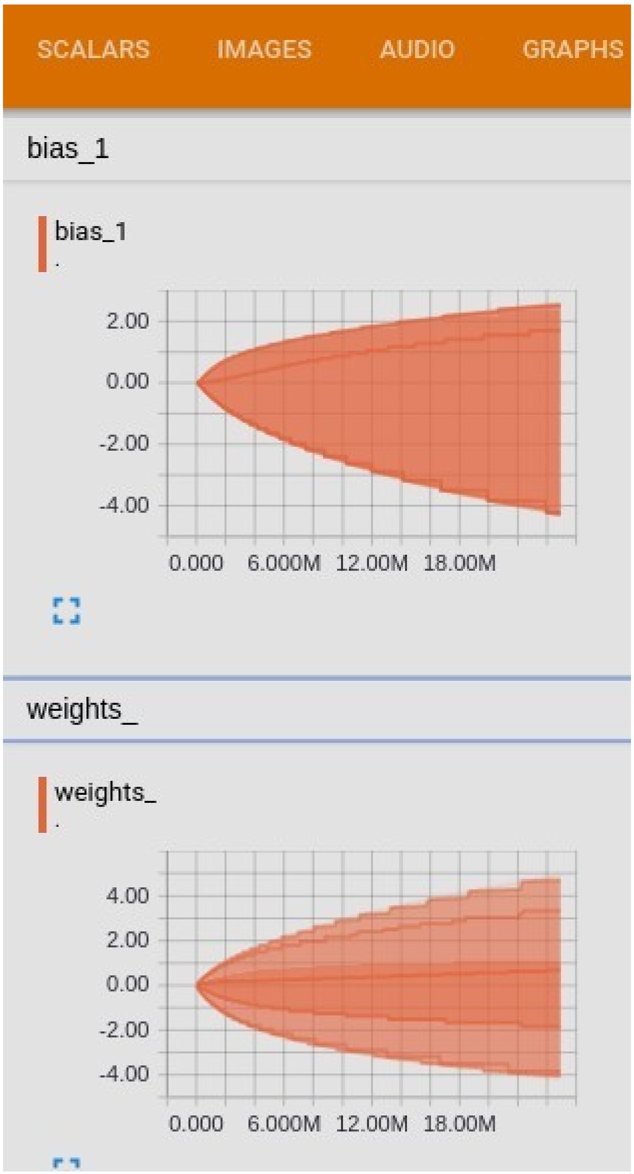


Epochs : 5000


SCALARS
IMAGES
AUDIO
GRAPHS
Accuracy_
Accuracy_
0.960
0.920
0.880
0.840
0.800
0.000
6.000M
12.00M
18.00M
run to download
CSV
JSON
Entropy_loss_
Entropy_loss_
0.600
0.400
0.200
0.00
0.000
6.000M
12.00M
18.00M

Adam Optimizer






TensorBoard visulization

For this visualization

created scalar variables for entropy loss  and accuracy
while for histogram and distribution using bias and weights

```
tf.summary.scalar("Accuracy:", accuracy)
tf.summary.scalar("Entropy loss:",cross_entropy)
tf.summary.histogram("weights ", W)
tf.summary.histogram("bias",b)
```

**Question 1 : <u>Logistics Regression On the Olivetti faces dataset</u>**

Same execution process

pre-processed the dataset ( same old but without normalizing , as that data is approximately already normalized )

```
def prepData() :
        enc = OneHotEncoder()

        olivetti = datasets.fetch_olivetti_faces()
        X , y = olivetti.data, olivetti.target

        nb_classes = 40
        targets = np.array(y).reshape(-1)
        y_ = one_hot_targets = np.eye(nb_classes)[targets]

        X_train, X_test , Y_train, Y_test = train_test_split(X, y_, test_size=0.10, random_state=42)
```

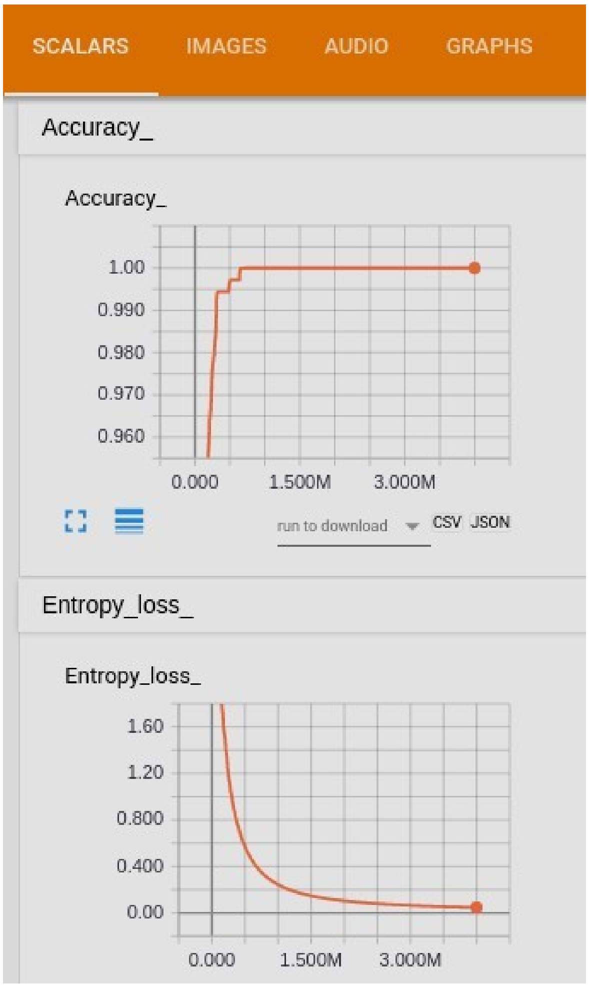
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

which allows us to interpret the outputs as probabilities, while cross-entropy loss is what we use to measure the error at a softmax layer, and is given by[1]

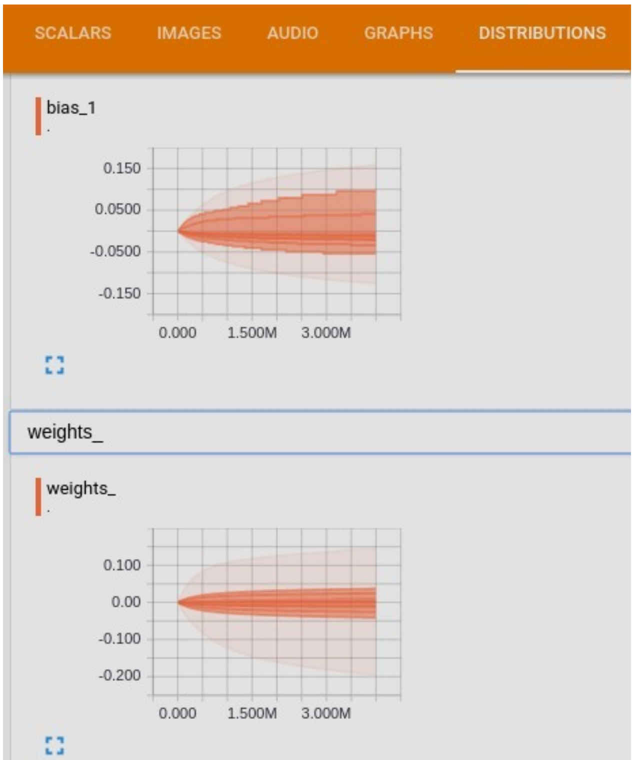
$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N} \left[ y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \right],$$

Same entropy function

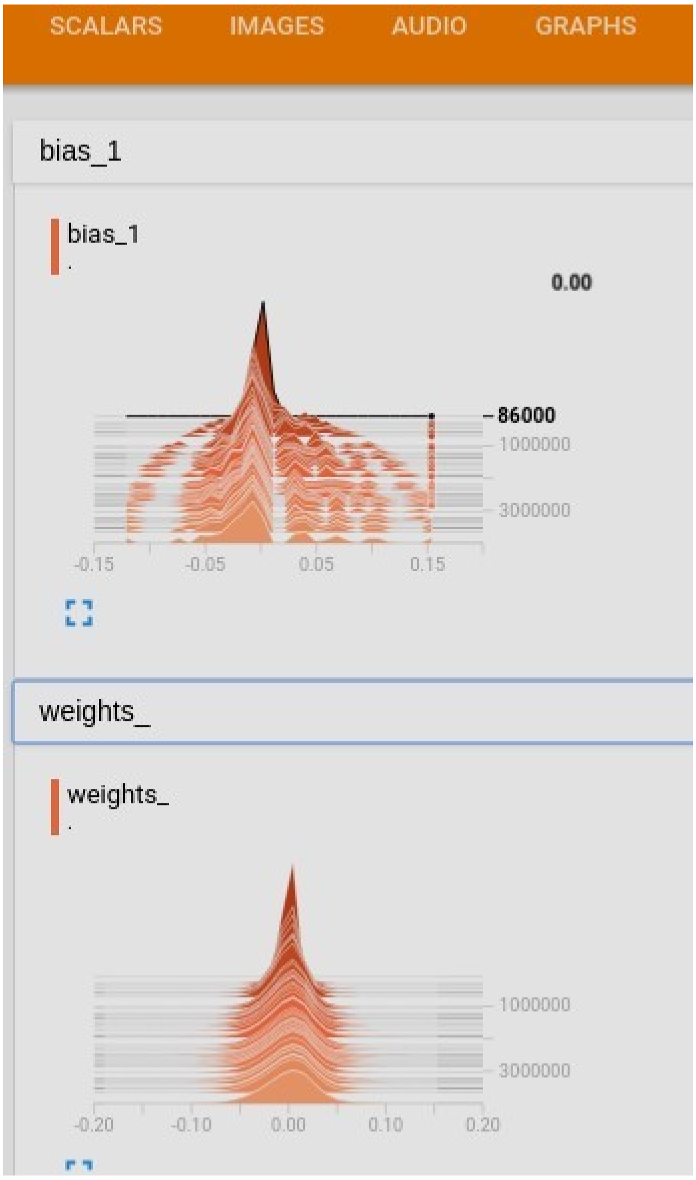
But the only difference is it converges on gradientDecent even with smaller learning rate




Distributions :



Histograms :


SCALARS
IMAGES
AUDIO
GRAPHS
bias_1
bias_1
0.00
86000
1000000
3000000
-0.15
-0.05
0.05
0.15
weights_
weights_
1000000
3000000
-0.20
-0.10
0.00
0.10
0.20