REPORT

HOME WORK – 2

Aim:

Implement and study the DNN model for multi class classification for Activity Recognition using Motion Sense Dataset from github Protecting Sensory Data against Sensitive Inferences.

1. Build DNN and tune hyperparameters using grid search to obtain the best fit for the data with high accuracy.

   Model:

   Linear layers with ReLU as an activation function for all the hidden and the input layers whereas the activation function for the output layer is Softmax.

   Loss Function: Cross Entropy Loss

   Optimizer: Adam optimizer

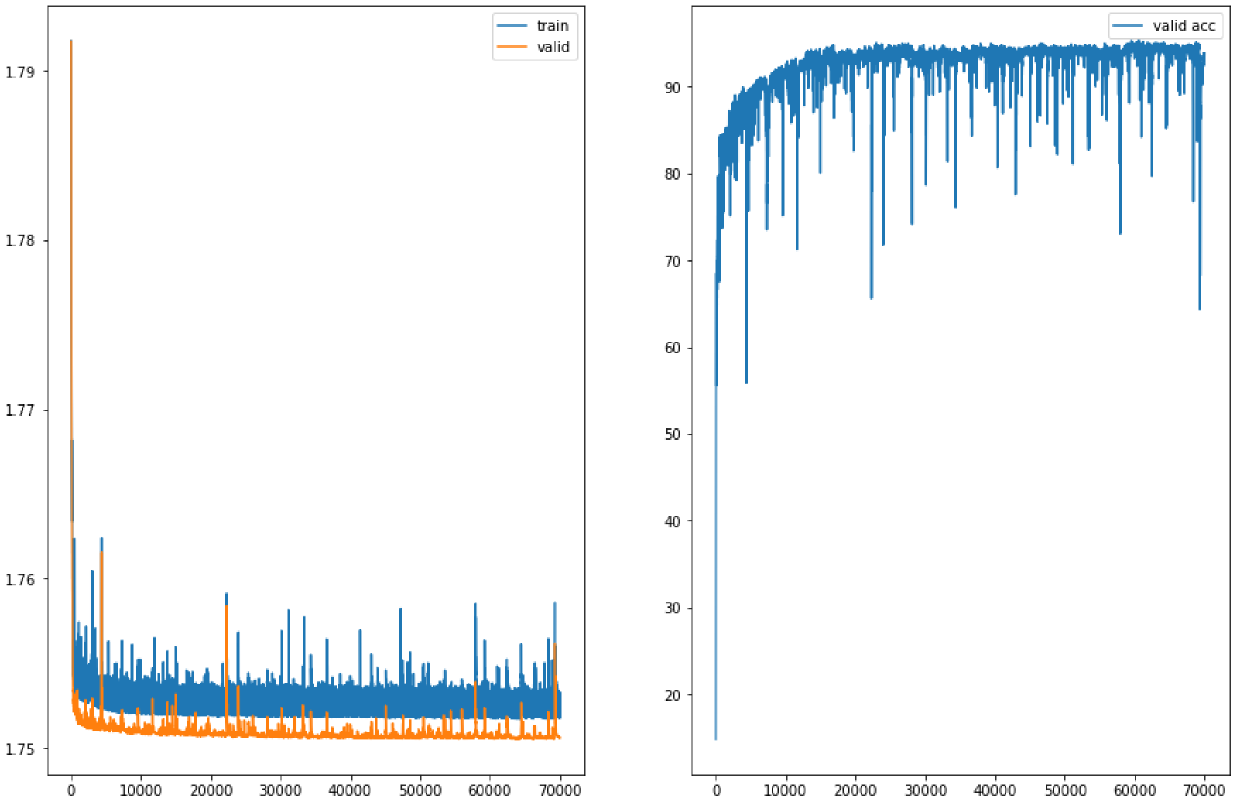
   Hyperparameters:

   - Number of hidden layers
   - Number of hidden units
   - Learning rate
   - Number of iterations
   - Batch size

   Five different values are given to each parameter and the combination is formed randomly to create a set of five values each hyperparameter set. So there are total of 5^5 sets to test on. Many sets gave good accuracy but poor F1 score. The best hyperparameters in terms of accuracy and F1 score are reported below after performing the grid search,

   Number of hidden layers=1
   Number of neurons=10
   epochs=2000
   learning Rate=0.01
   batch size=128

```
epoch : 2000, train loss : 1.7517, valid loss : 1.7506, val
     id acc : 93.83%
precision, recall and F1 micro: (0.9382940108892922, 0.9382
940108892922, 0.9382940108892922, None)
precision, recall and F1 macro: (0.9250371296179493, 0.9341
```



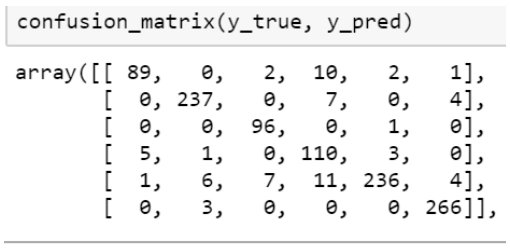
```
480804978616, 0.9279136571988366, None)
```

Below the plot for training loss, validation loss and validation accuracy :

2. **Evaluating Model Performance**:

Micro
   Precision: 0.93
   Recall: 0.938
   F1 Score: 0.93

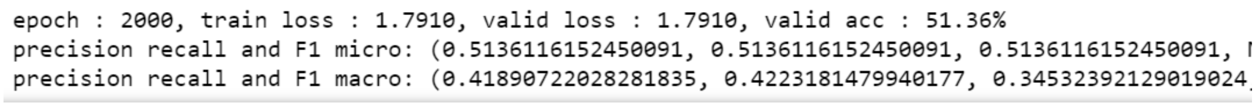
Macro
   Precision: 0.92
   Recall: 0.93
   F1 Score: 0.927



```
confusion_matrix(y_true, y_pred)
```

```
array([[ 89,   0,   2,  10,   2,   1],
       [  0, 237,   0,   7,   0,   4],
       [  0,   0,  96,   0,   1,   0],
       [  5,   1,   0, 110,   3,   0],
       [  1,   6,   7,  11, 236,   4],
       [  0,   3,   0,   0,   0, 266]],
```

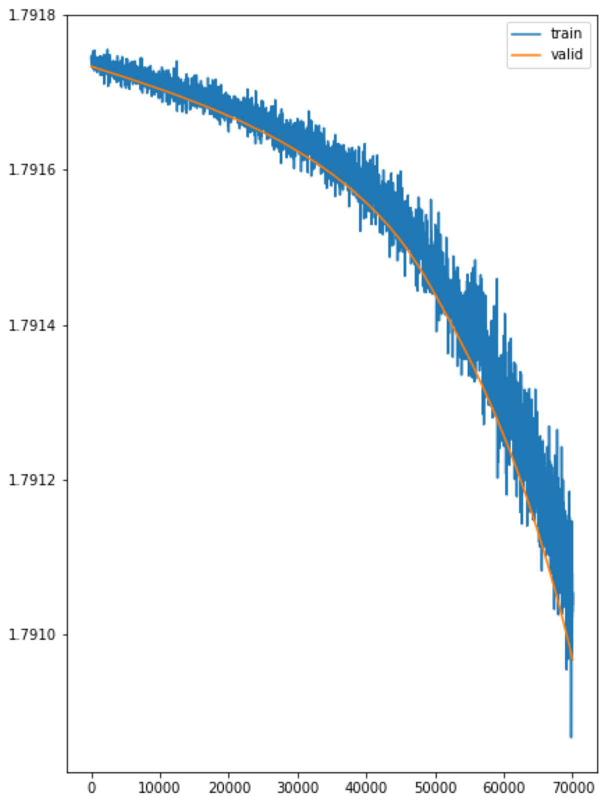
3. Training model with different optimizer :

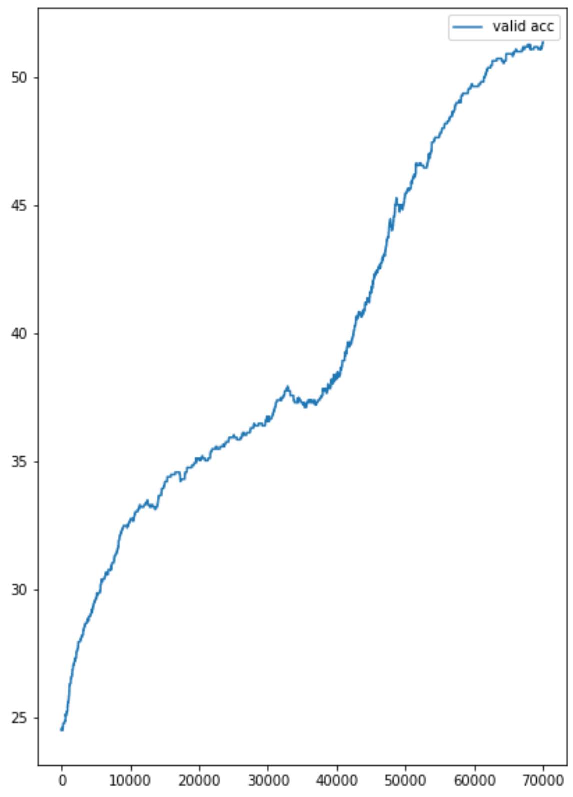


```
epoch : 2000, train loss : 1.7910, valid loss : 1.7910, valid acc : 51.36%
precision recall and F1 micro: (0.5136116152450091, 0.5136116152450091, 0.5136116152450091,
precision recall and F1 macro: (0.41890722028281835, 0.4223181479940177, 0.34532392129019024
```

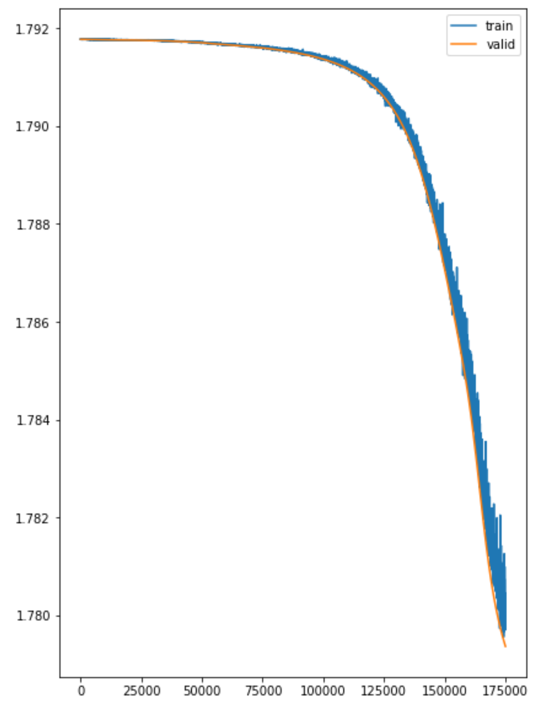
The plots are shown below on training with the best performing hyperparameters with different optimizers

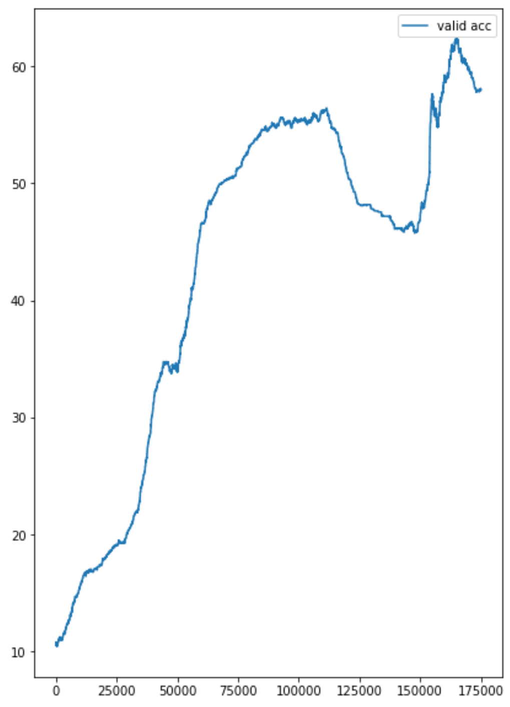
Gradient Descent:



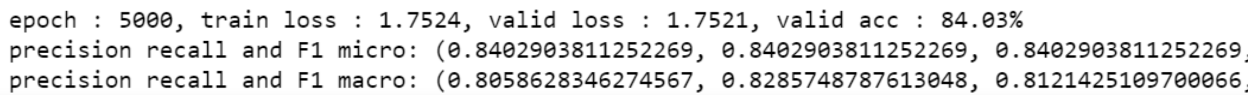


On increasing epochs to 5000:

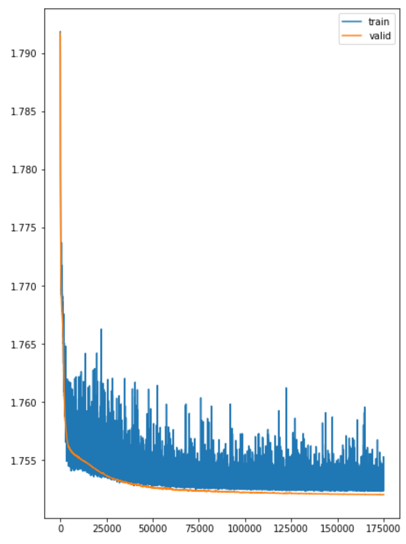


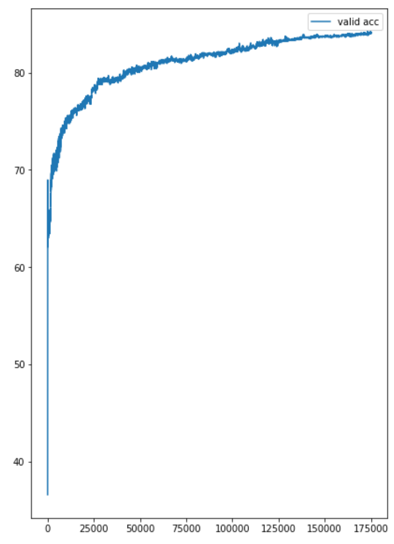


Adagrad:

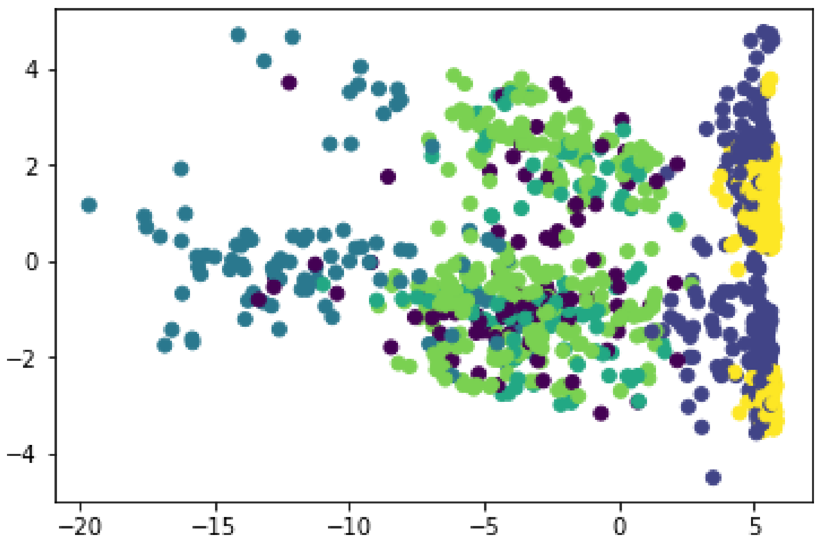


```
epoch : 5000, train loss : 1.7524, valid loss : 1.7521, valid acc : 84.03%
precision recall and F1 micro: (0.8402903811252269, 0.8402903811252269, 0.8402903811252269,
precision recall and F1 macro: (0.8058628346274567, 0.8285748787613048, 0.8121425109700066,
```

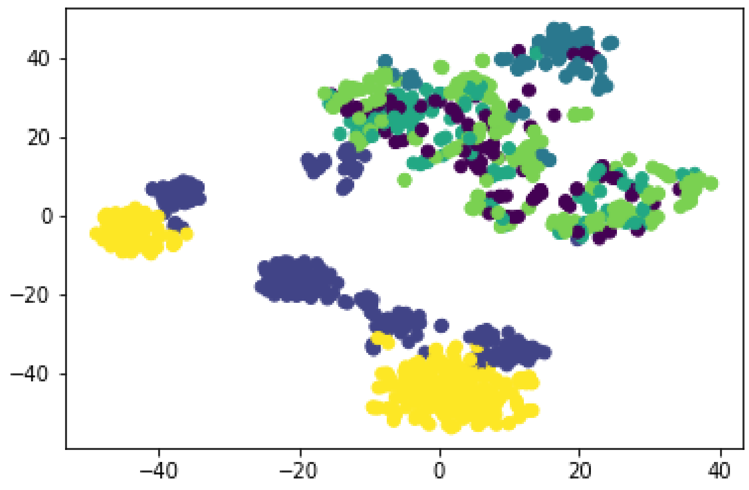




4. **Visualizing through PCA**:



5. **Visualizing through t-SNE**:



6.

After observing the visualization, it is clear that there is not enough to set apart the different classes from each other in the PCA visualization using first two principal components. Whereas the other t-SNE is a probabilistic approach of reducing dimension which provides comparatively better results in setting apart the different classes each other.