Pattern Lab Final Project Report

Group – 2

**Topic**: Human Activity Recognition with Smartphones

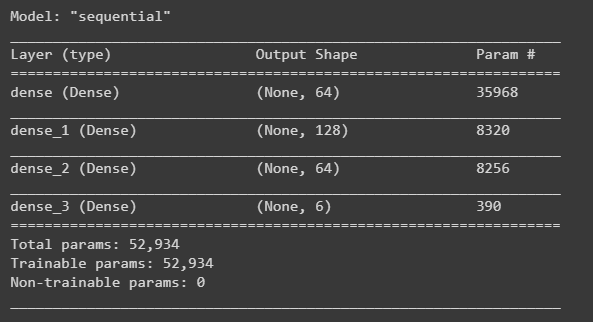
|  |  |
| --- | --- |
| **Name** | **ID** |
| Md Pavel Rahman | 011 172 017 |
| Ashraful Alam | 011 172 005 |
| A.K.M. Ifranul Hoque | 011 172 189 |
| Iftekhirul | 011 172 006 |
| Faysal Ahmed | 011 172 030 |

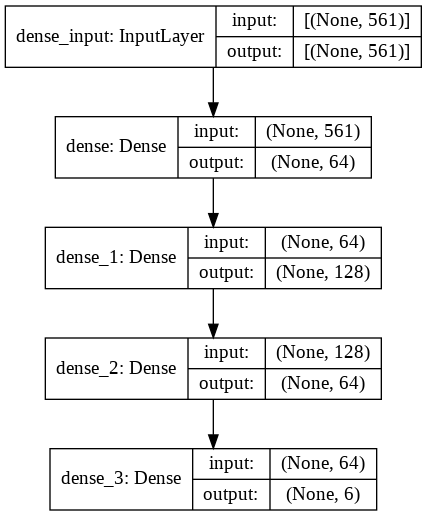
1. **Problem Definition:**

The Human Activity Recognition database was built from the recordings of 30 study participants performing activities of daily living while carrying a waist-mounted smartphone with embedded inertial sensors. The objective is to classify activities into one of the six activities performed**.**

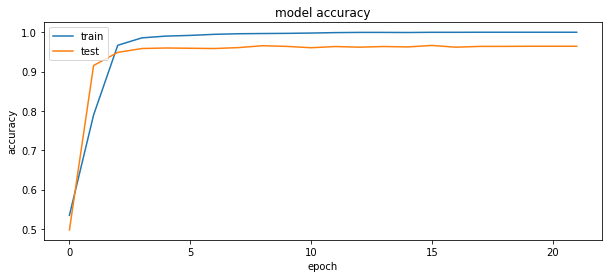
The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKINGUPSTAIRS, WALKINGDOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

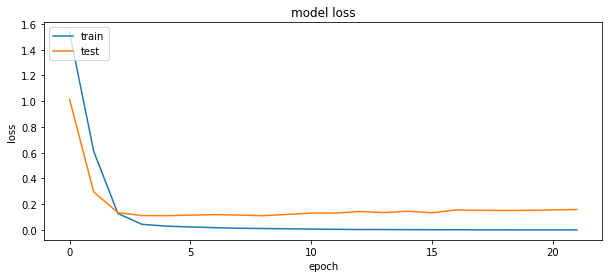
1. **Dataset Description:**
   1. Number of instances: 7352(train set), 2947(test set) separated as different number of subjects were stored for each. 21 subjects for train set and 9 subjects for test set
   2. Number of Attributes: 563 (561 features, 1 for subject id and 1 for class label).
2. **State of the art:** 
   1. Deep Residual Bidirectional Long Short-term Memory LSTM (Deep-Res-Bidir-LSTM) network. Used for OPPORTUNITY Activity Recognition Data Set
3. **Proposed Approach:**
   1. Prepare the data
   2. Extract features
   3. Choose models to train data
   4. Test/Predict the models
   5. Evaluate the results
4. **Actual Approach:**
   1. We used Google drive to import the specified train set and test sets inside Google Colab.
   2. We checked for any null values present in the datasets.
   3. The dataset was divided into X\_train, y\_train, X\_test, y\_test using pandas iloc function.
   4. Total counts for each category and their labels were used to design a pie chart. It was seen that each of the 6 labels were equally distributed in the train dataset.
   5. A calculation of the columns that had data of accelerometers, gyroscopes and others were done. Total count was taken and a bar chart was formed. It was seen that there were more columns that had accelerometer data
   6. Standard Scaling Preprocessing was done to normalize the X\_train and X\_test data.
   7. LabelEncoder was used to convert the string Labels into numbers. Then they were converted into 0s and 1s array with 6 values of 0s and 1s per row for each of the 6 labels.
   8. PCA with various n\_components was used to determine if the accuracy increased or not after each of the predictions. It was seen than n\_components with None had the best accuracy whereas, if we didn’t use PCA the accuracy decreased. Here is a list of PCA accuracies measured:
      1. with pca(All) acc = .959
      2. with pca(n = 2) acc = .583
      3. with pca(n = 8) acc = .83
      4. with pca(n = 16) acc = .86
      5. with pca(n = 64) acc = .918
      6. with pca(n = 256) acc = .949
      7. without pca acc = .946
   9. Keras Library was used with Sequential Model to train the data. We used a total of 4 layers. The first 3 layers had units 64, 128 and 64 with ReLu as activation. The last layer had 6 units and a SoftMax Activation which is normally used for multi class label data. This is a brief summary below:



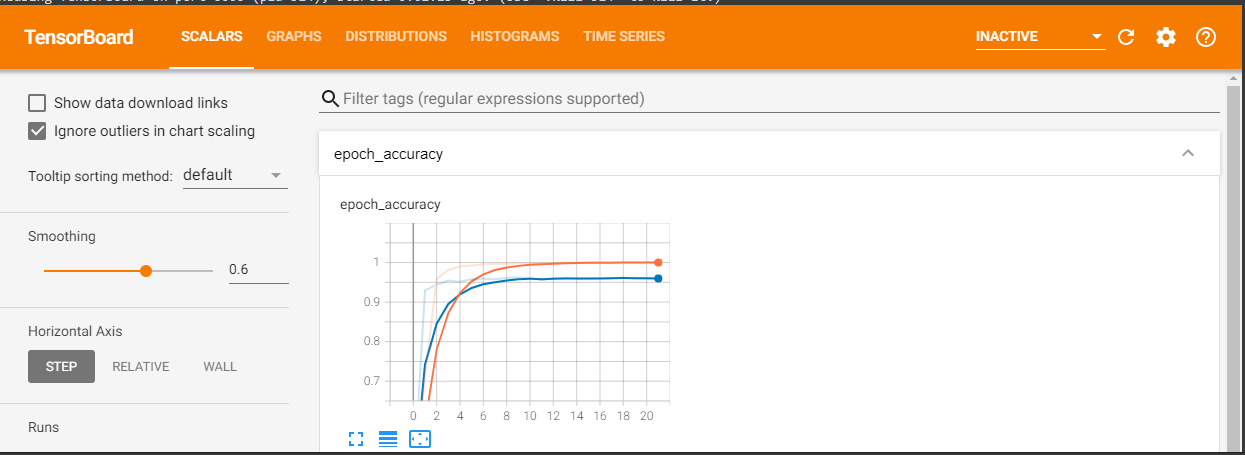


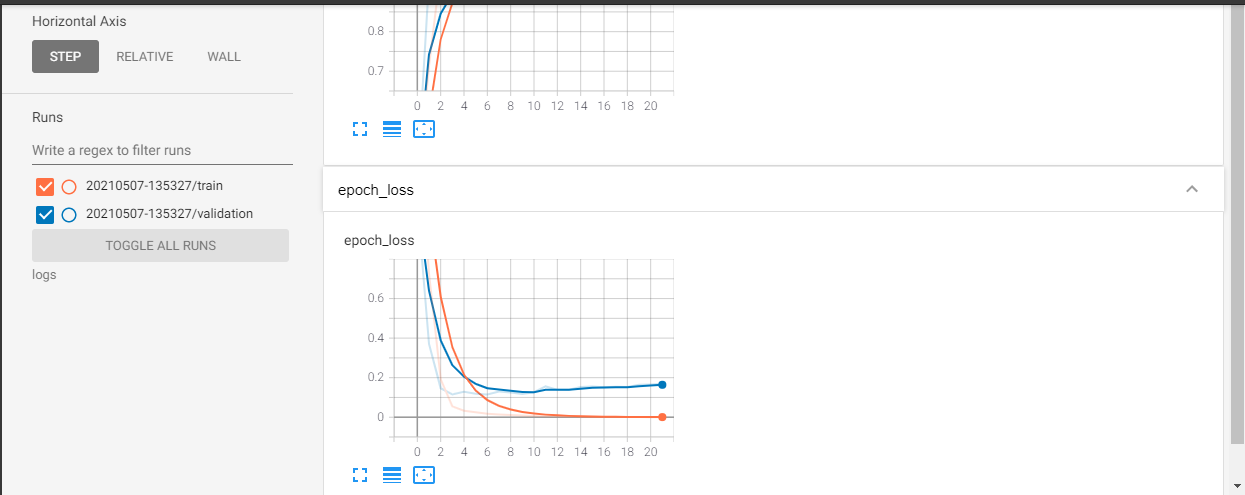
* 1. The model was fitted with a batch size of 256 and 22 epochs.
  2. Afterward two graphs of Model Accuracy and Model Loss was drawn.





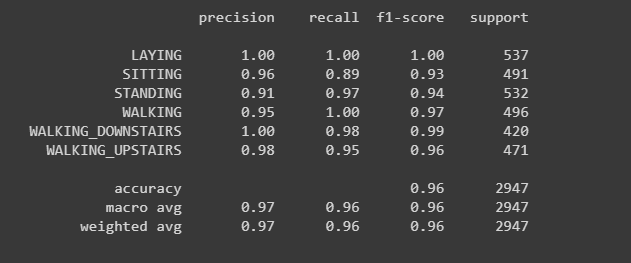
* 1. We also used TensordBoard to get more dynamic results.



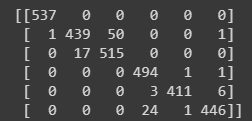


* 1. The model was used to predict the test data
  2. We converted the encoded data of 0s and 1s arrays back to normal class labels and then used various classification metrics on them.
  3. The following metrics were used: Confusion Matrix, Accuracy Score, Classification Report (Precision, Recall, F1-Score, Support)
  4. Additionally, we used several sklearn models to predict the test data. We have shown the results in the Comparison Metrics.

1. **Problems Faced:**
   1. Various Coding Debugging problems
   2. Lack of Keras library usage experience
   3. Some metrics were difficult to understand such as explained variance for PCA.
   4. Had to convert label encoder values to 0s and 1s array otherwise model didn’t fit due to the following error: ValueError: Shapes (None, 1) and (None, 6) are incompatible
2. **Result/Comparison Metrics:**
   1. Accuracy: **96.4%**
   2. Precision (tp / (tp + fp)) measures the ability of a classifier to identify only the correct instances for each class.
   3. Recall (tp / (tp + fn) is the ability of a classifier to find all correct instances per class.
   4. F1 Score is a weighted harmonic mean of precision and recall normalized between 0 and 1. F score of 1 indicates a perfect balance as precision and the recall are inversely related. A high F1 score is useful where both high recall and precision is important.
   5. Support is the number of actual occurrences of the class in the specified dataset.



* 1. Confusion Matrix to evaluate the accuracy of a classification.



For example, for the first column “Laying” Label, there has been 537 labels correctly done for “Laying” but 1 label not correctly done which got in “Sitting” Label.

* 1. For Sklearn Models precision, recall, f1 score:
     1. **KNN** (0.8939728910641617, 0.8863718181403843, 0.8877674593824842)
     2. **SVC** (0.9564776422715822, 0.9544090433813274, 0.9551224461699493)
     3. **Decision Tree** (0.3129409075521595, 0.5, 0.36851054649856935)
     4. **Random Forest** (0.8304552215416571, 0.6853472593492415, 0.6571860838342053)
     5. **MLP** (0.9573501496099744, 0.9515723276157008, 0.9526554914698604)
     6. **Ada Boost** (0.3129409075521595, 0.5, 0.36851054649856935)
     7. **Gaussian Naive Bayes** (0.7924655966330829, 0.7690658908588546, 0.7672347781376718)

It is seen that SVC and MLP performs well.