TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Human Activity Recognition (HAR) is a process aimed at the classification of human actions for a given period based on discrete measurements (acceleration, rotation speed, geographical coordinates, etc.) collected from various sources or devices. These devices can be wearable sensors [1], smart phone inertial sensor [2][3], vision based sensor like Kinect [4][5] and Closed Circuit Television [6]. Such widespread adoption of sensors provide unparalleled potential for data collection to study human behavior and use of HAR in domains such as healthcare [1][3], surveillance [7][8], remote care to the elderly [5][7], smart surroundings [3][7], Sports & Exercise [9] etc. to improve the safety and quality of human life.

Currently, more than 5 billion mobile devices with several sensors (e.g., accelerometer and GPS) are in use, capturing detailed, continuous, and objective measurements on different aspects of our life, including physical activity. Smartphones, with appropriate storage, strong processors, and wireless transmission, may collect massive amounts of data about huge groups of people over long periods of time without the use of extra hardware or instruments. Applying suitable machine learning models on this collected data is the essence behind obtaining the desired results for solving the task of HAR, which in by recognizing the right activity.

One such solution to HAR is discussed ahead. The data for this task has been obtained from WISDM (Wireless Sensor Data Mining) Labs which is concerned with collecting the sensor data from smart phones and other modern mobile devices (e.g., tablet computers, music players, etc.) and mining this sensor data for useful knowledge [10]. The data is collected from 36 different users performing six types of human activities (ascending and descending stairs, sitting, walking, jogging, and standing) for specific periods of time using accelerometers.

The report discusses exploration and analysis of this data, and application of various machine learning (ML) models to recognize the six activities. This being a classification problem with labels present for training, several supervised ML models have been tested starting from the classical models like Logistic Regression, Support Vector Machine, Random Forrest Classifiers, to the neural network models like the deep neural network, convolution neural network etc., along with changing the input features and tuning the parameters to obtain the desired results. The best result has been obtained using a model with combination of multiple layers of convolutional neural network and deep neural network (dense) with suitable hyperparameters.

DATA DESCRIPTION

The dataset contains information about 36 users who performed six different human activities, including ascending and descending stairs, sitting, walking, jogging, and standing. This data was acquired from accelerometers, which are able of detecting the orientation of the device measuring the acceleration along the three different dimensions and determine device orientation. They were collected using a sample rate of 20 Hz (1 sample every 50 millisecond) that is equivalent to 20 samples per second. The time series data was divided into 10-second snippets and statistical features, or metadata were extracted from each snippet.

Both the timeseries data (signals) and metadata provided was pre-split into the train and test data set. The training data set comprised of 28 users with their activities known. The test data set comprised of the remaining 8 users of whom we must predict the activity.

DESCRIPTION OF FILES AND FOLDERS

1. 'signals.csv' file: contains the time-series data organized by users and snippets with following columns.
    a. user_snippet: User id + snippet id.
    b. timestamp: in milliseconds
    c. x-axis: The acceleration in the x direction as measured by the phone's accelerometer. Values are floating-point between -20 and 20. A value of 10 = 1g = 9.81 m/s^2, and 0 = no acceleration. The acceleration recorded includes gravitational acceleration toward the center of the Earth, so that when the phone is at rest on a flat surface the vertical axis will register +-10.
    d. y-axis: same as x-axis, but along y axis.
    e. z-axis: same as x-axis, but along z axis.
2. The 'signals_test.csv' file contains the time-series data corresponding to the test subset. It follows the same structure as the 'signals.csv' file.
3. The 'metadata.csv' file is a CSV file contains statistical features extracted from the signals and the target (activity) with the following columns:
    a. user_snippet: the identifier of the user who acquired the data (integer) + the snippet identifier (integer).
    b. Activity: the activity that the user carried out at the corresponding snippet. This is the target, which could be one of the following class labels:
        i. Walking
        ii. Jogging
        iii. Sitting
        iv. Standing
        v. Upstairs
        vi. Downstairs
    c. Extracted features – the rest 30 columns correspond to 10 features extracted from the x, y and z acceleration time series. Each column name has the format D-axis__FEATURE, where D is either x, y or z; and FEATURE is one of the following:
        i. sum_values
        ii. median
        iii. mean
        iv. length
        v. standard_deviation
        vi. variance
        vii. root_mean_square
        viii. maximum
        ix. absolute_maximum
        x. minimum
4. The 'metadata_test.csv' file has all the columns as 'metadata.csv' without the 'activity' column.

DATA EXPLORATION, PRE-PROCESSING AND ANALYSIS

Having four data sets (files) available, two each (train + test) for signals and metadata, first task was to add the activity column to the training set of time series data (signals) as it was unavailable in the provided dataset. This was done by merging the signals data with subset of metadata (activity), on the 'user_snippet' attribute which acted as the unique user identifier.

The data now contained the following number of entries (rows) and attributes (columns) including the target feature.

Data Set (Files - csv)	No. of Entries (Rows)	No. of Features (Columns)
signals	811637	6
signals_test	261686	5
metadata	8129	32
metadata_test	2621	31

Table 1. Shape of the Data Set

One key observation here was that there were no missing values found. Apart from user identifier and target feature which were objects/strings, rest all features in both sets of data were numerical continuous. The training set comprised of 28 unique users while test set had the remaining 8 users. The (training) data set was highly unbalanced in regards to each activity (Fig. 1). This might likely be due to the type of activity conducted. Walking and Jogging might need more data (more signals recorded) due to more movement whereas less for sitting and standing.

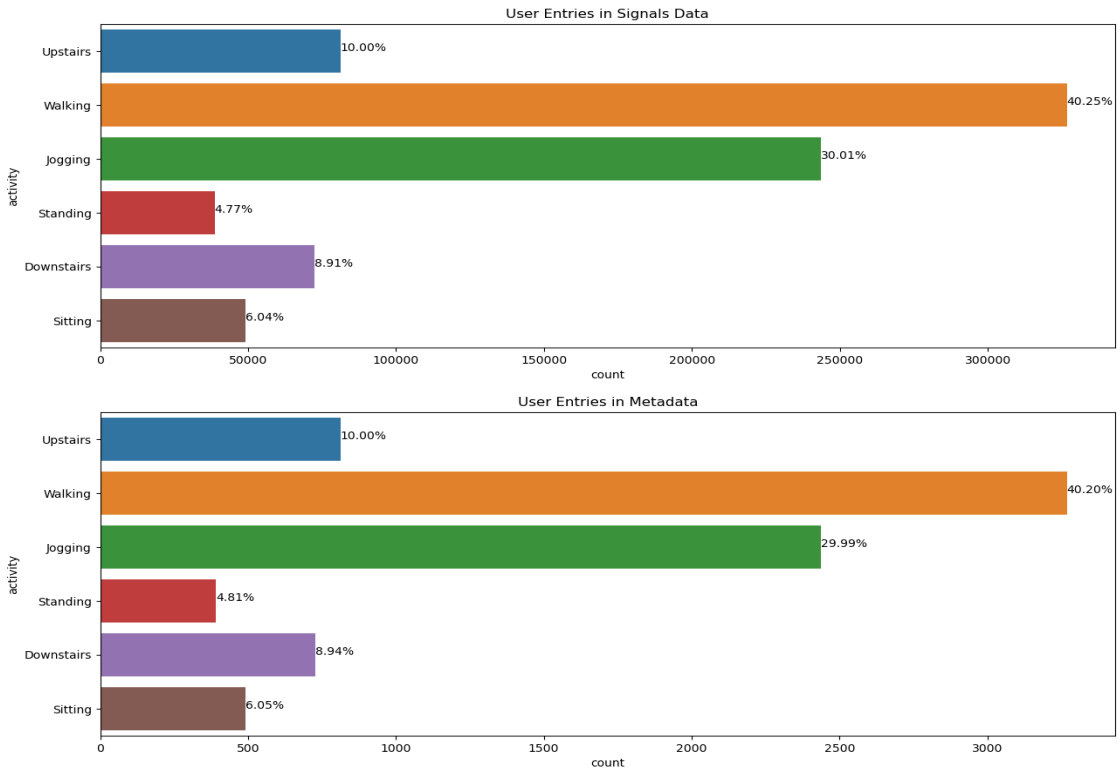

Figure 1. Number of User records (percentage) for each activity

Upon further exploration on the timeseries data, it was observed that the data distribution for acceleration along x-axis and z-axis was closer to a normal distribution, while skewness was observed for acceleration along the y-axis (Fig. 2).

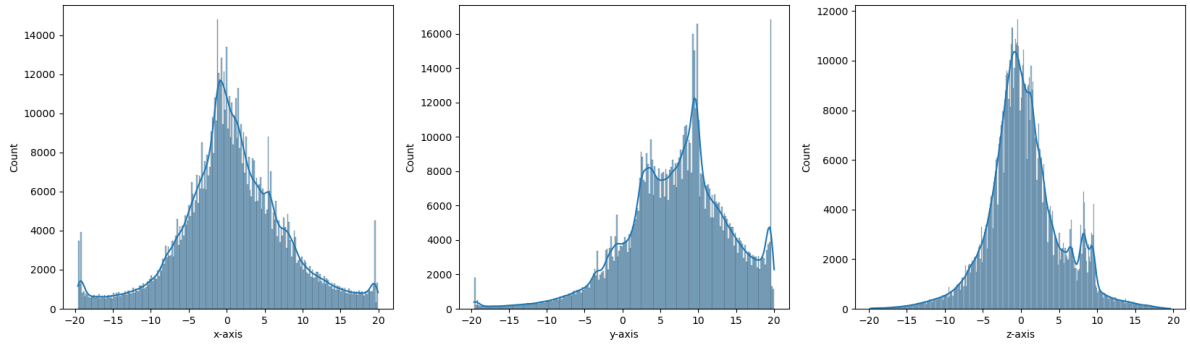

Figure 2. Time series data distribution

A pair-plot was used to check if there is any relationship between the features and the target variable but since data being in large amount, not much was observed, however some distinctions could be seen e.g. Walking, downstairs and jogging are seen to be separable if we used just two features in acceleration along x-axis and y-axis (Fig. 3).

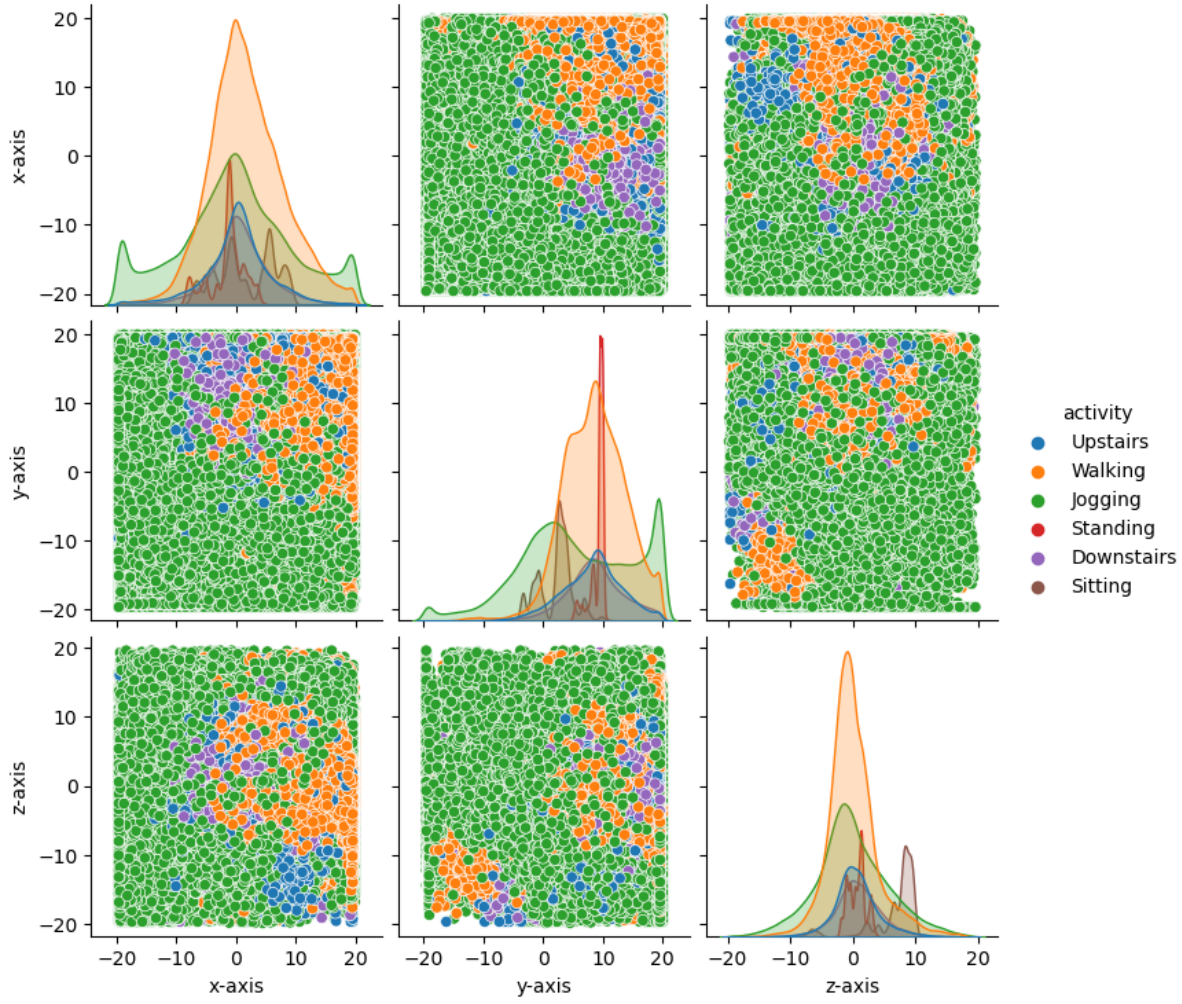

Figure 3. Signals data pair plot

The acceleration values for a sample of 1000 entries were plotted over time for each activity to check if we can observe clear distinction between the activities w.r.t the accelerations. It was observed that the variations in acceleration in the 3 direction for activities like Standing and sitting can easily be distinguished compared to other activities. Activity like walking and jogging show similarities or slight overlap in acceleration values in direction of y and z axis.

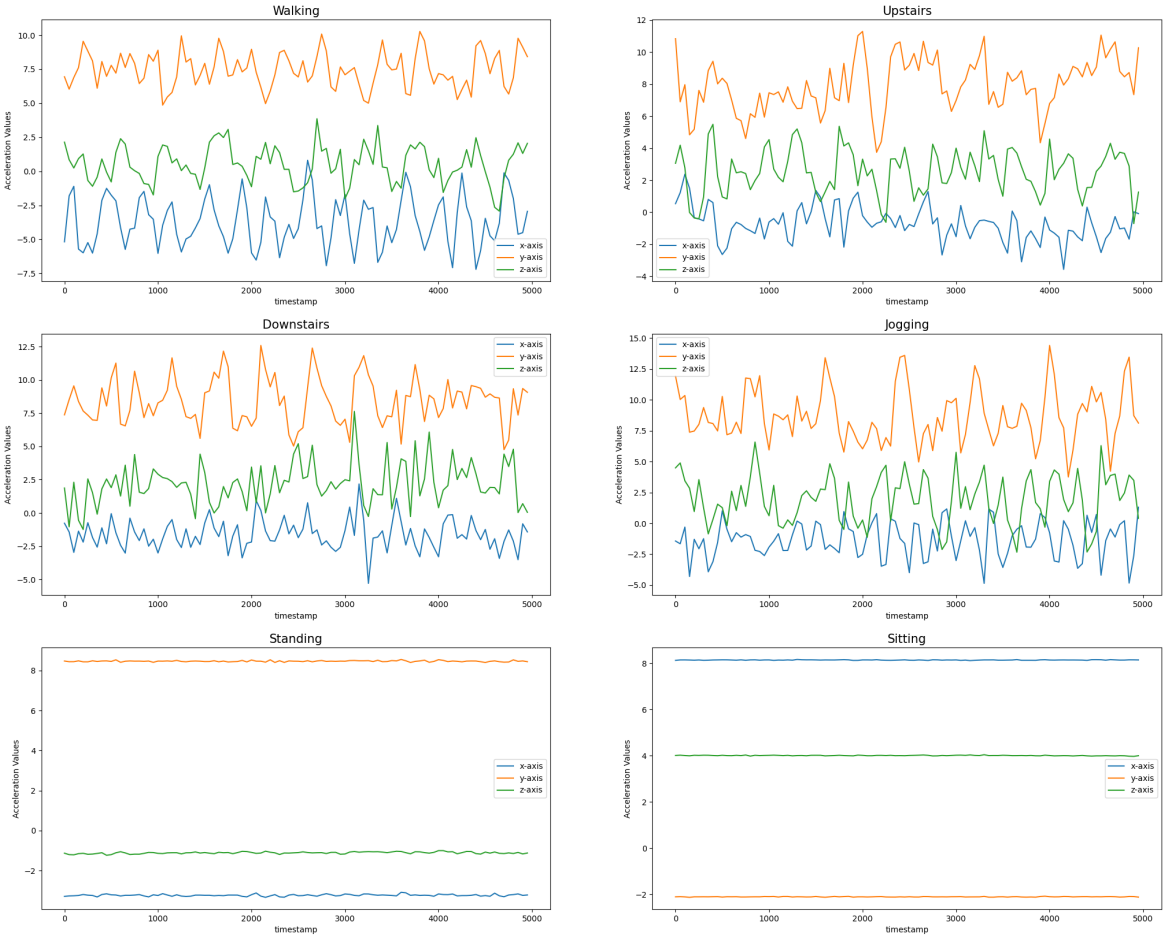

Figure 4. Acceleration values over time for each activity

Outliers were observed in accelerations for all the categories (Fig. 5) but due to large amount of sample with each being necessary for our modelling purpose, outliers were not neglected.

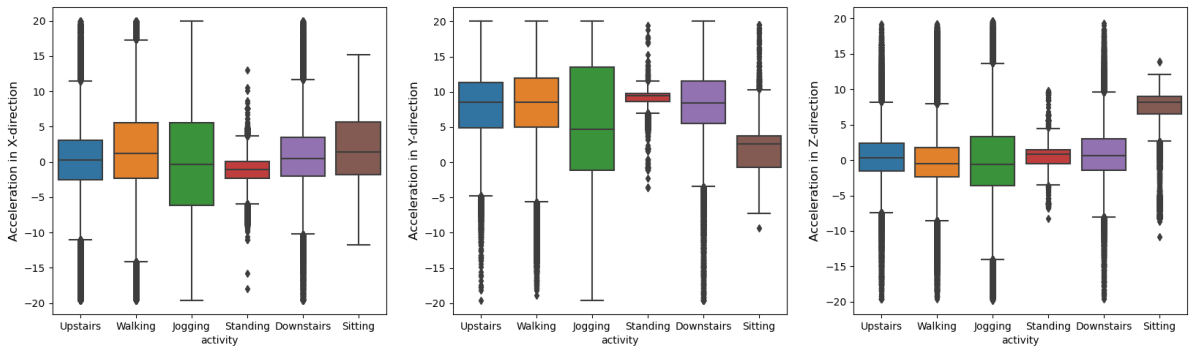

Figure 5. Acceleration distribution (box plot) across all activities

Another key observation w.r.t time series data was that there is no user (train or test set) who has acceleration of 0 across the three axis together, at any point of time. Also, the user snippets were 5 seconds each (0-5000 milliseconds), but we could see that there were few snippets that are less than 5 seconds.

- Total Unique User Snippets (id):
    - Train set - 8129
    - test set - 2621
- Snippets with less than 100 entries (for each id):
    - train set - 28
    - test set - 8

To use this time series data for neural network models like CNN and LSTM, it was essential to impute the data or complete the time series so that each of the (8129, 2621) user snippet had 100 samples/entries which was 5 seconds each. For this purpose, 2 techniques were used.

- Forward Filling Imputation: One data set (train + test) was created using forward filling technique to fill the gap (say n_i rows) where in for each user snippet with less than 100 samples, the last sample of each respective snippet (i) was repeated (added) 100- n_i times to complete the time series.
- Zero Filling Imputation: Similarly another data set was created with similar technique but instead of filling the gap with the last sample, 0 value was used.

These two data sets were used to check for the best results. So the final number of entries (rows) were 812900 and 262100 for train and test respectively for each data set.

With target variable being categorical Pearson's correlation coefficient was not used, instead Chi-Square feature selection test to check the features important or relevant to the target variable. When there are three or more levels for the predictor, the degree of association between predictor and outcome can be measured with statistics such as chi-squared tests [11]. Metadata was tested with chi square test as it consists of thirty plus features, though before using this, min-max standardisation (only for this purpose) was performed to normalise the data i.e. to get values between zero to one as chi-square feature selection test required input data to be non-negative. Following the test, 'y-axis_variance' was observed to be the most important while the length features along the three axis were found to be least important (Fig. 6). For modelling purpose however models with and without the length features were tested as a feature which is irrelevant to the target feature can be important when coupled with other features.

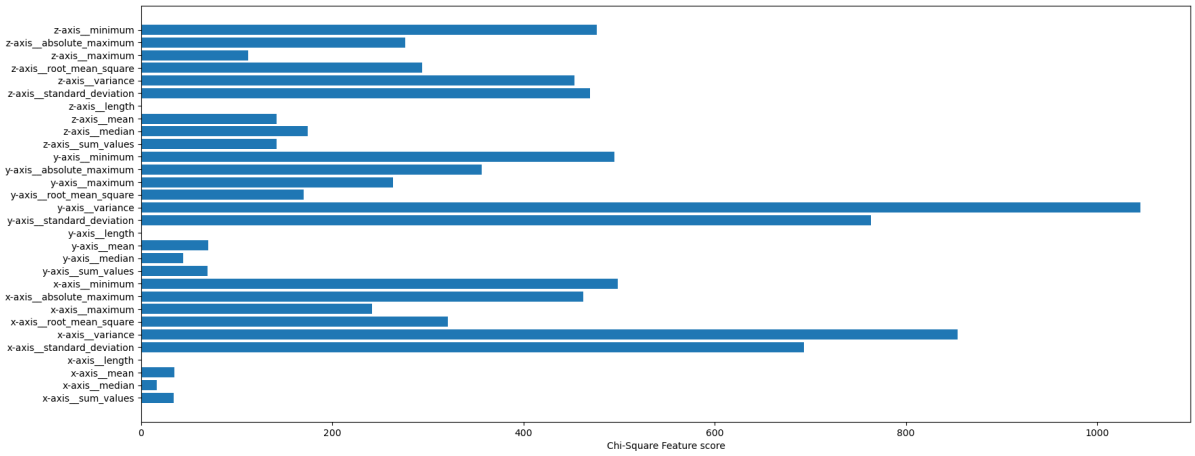

Figure 6. Chi-Square feature importance on Metadata

Before modelling the data (timeseries as well as statistical data), it was necessary to normalise the data (feature scaling), as to make sure each feature is on same scale and, is equally important and to make it easier for ML algorithms to process. This was achieved using 'StandardScaler' so that each feature has zero mean and unit standard deviation.

Train and test data had to prepared accordingly, where independent variables (X) comprised of all the features bar 'user_snippet', target 'activity' and, 'timespan' in case of timeseries data. The dependant feature (Y) was 'activity'. This was prepared for both train and test data although for test data Y was unavailable as the objective was to determine the activity for the test data. Standardization was performed for X whereas label encoding was performed for Y. Label encoding (Changing activity label objects to numerical codes e.g. 'upstairs' - 0, 'downstairs' - 1..) was required as most of the machine learning algorithms were unable to process direct string label objects.

Additionally, a second encoding was required for the initially encoded labels (Y) when using neural network algorithms. Since the problem is multiclass classification, the loss function used in the neural network models was 'categorical_crossentropy', hence categorical encoding was performed to convert the label class vector (Y) to the matrix of binary class. Also, the data was reshaped accordingly based on the algorithm needs before applying the ML algorithms e.g. timeseries data was reshaped to 3 dimensional input before applying Convolutional Neural Networks.

MACHINE LEARNING MODELLING AND RESULTS

Once data was pre-processed, machine learning algorithms were applied to create the models for predicting the human activity. After obtaining the results (predicted output for the test data), inverse transformations were applied to the model outputs to decode the predicted labels so as get the result in desired format which is activity (string object) associated with respective user snippet. The models were compared mainly with the test data prediction accuracy/score. Additionally, Neural network models were compared to each other with convergence of validation loss and accuracy.

DIMENSIONALITY REDUCTION MODELLING FOR VISUALISATION

ML models were created using statistical training data (metadata) with dimensionality reduction algorithms like Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP), for the purpose of visualization in 2 dimensions. One key objective here was to check if the activity clusters formed after reducing the dimensions were separable enough to distinguish the activities so as, to check if it was useful enough for further modelling to use on test data. For t-SNE, perplexity check was also done using KL Divergence metric as perplexity controls the effective number of neighbors that each point considers during the dimensionality reduction process. The divergence stability was observed at 115 perplexity value (Fig. 7).

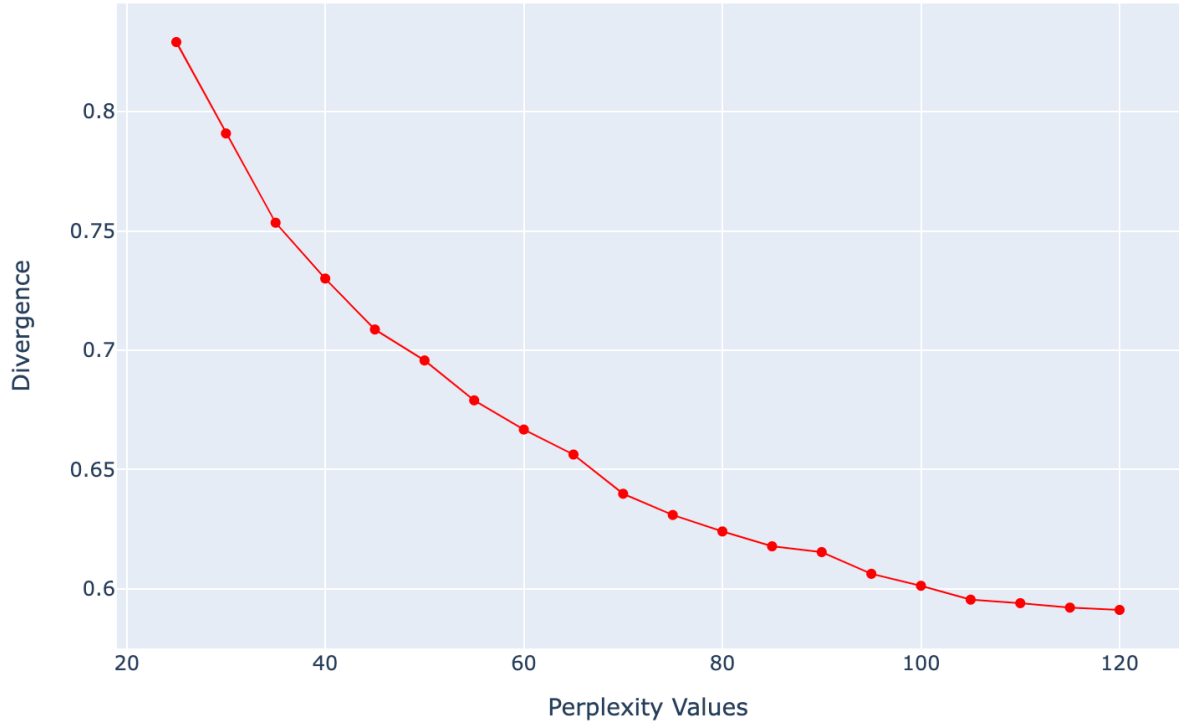

Figure 7. Perplexity vs Divergence

The dimensionality reduction on original statistical features showed t-SNE algorithm performed best. Clusters were visible with PCA though were not clearly separable which may be because PCA maintains linear relationships, but UMAP performed worst where no strong cluster was visible. Although clear clusters were visible with t-SNE, only sitting, standing and jogging data points were observed to be separable with few overlapping outliers, and major overlap observed among the other three in downstairs, upstairs and walking (Fig. 8).

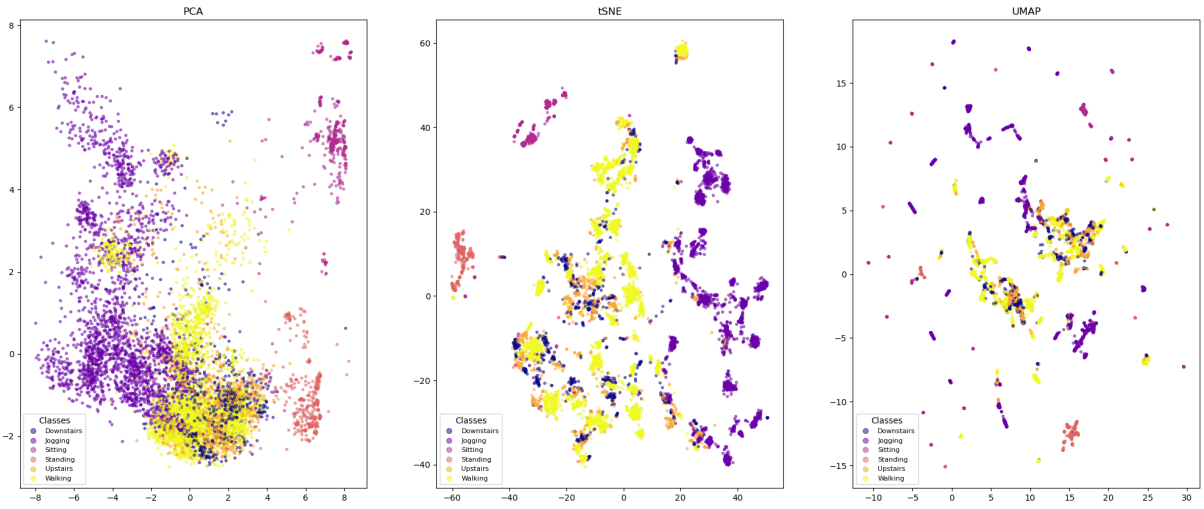

Figure 8. Dimensionality Reduction on original statistical feature data

Zero-Crossing features for time series data like kurtosis which is measure of the combined weight of a distribution's tails relative to the centre of the distribution curve, and skewness which is measure of the asymmetry of a distribution can act as optimal features for HAR [12]. Hence skewness and kurtosis were extracted from the time series and added to the existing statistical features data set. On performing dimensionality reduction on this new data, number of clusters remained same, but the data points were observed to be closely coupled with clear distinction and UMAP improved a lot wherein the cluster visibility was evident. However overlap was observed among three of the remaining activities as same as before.

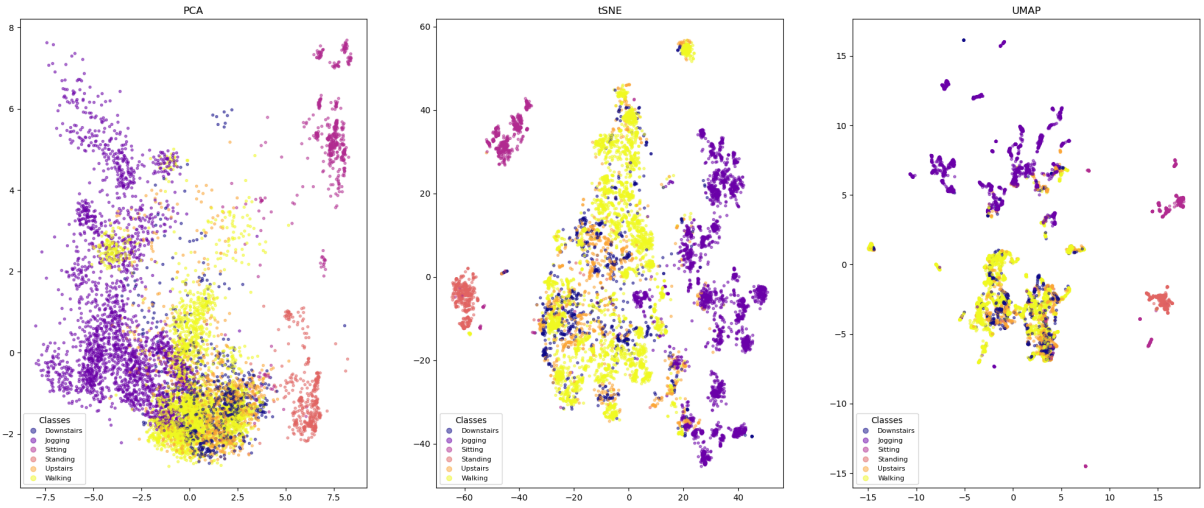

Figure 9. Dimensionality Reduction on statistical features data with addition of Kurtosis and Skewness

Visualising this data after reducing the dimensions was enough to suggest that statistical features alone may not be the best to separate all the 6 activities hence not used was testing purpose. Adding additional features can be helpful but not enough. Using the time series data directly may provide the better results.

LOGISTIC REGRESSION (LR)

First classification algorithm used was logistic regression with cross validation. Cross validation helps in evaluating machine learning models by training several models on subsets of the available input data and evaluating them on the remaining data and thus help to assess how a prediction model performs with an unknown dataset and reducing overfitting.

Cross-Validated version of logistic regression - 'LogisticRegressionCV' was used for model creation purpose as it allows better hyperparameter tuning along with stratified k-fold cross validation. To prevent overfitting ridge penalization was used, and with data being large amount, 'saga' was used for optimization. The penalization factor (inverse of regularization strength) was set as 6 to test 6 values between $1e^{-4}$ to $1e^{4}$ using 5 fold cross validation. Since logistic regression is a binary classification algorithm, ROC-AUC (Receiver Operating Characteristic Area Under the Curve) was used as scoring metric. ROC being the probability curve and AUC being the degree or measure of separability, allows us to compare the performance of different models trained on the same dataset, and to select the model with the best performance with LogisticRegressionCV. Since there were 6 classes and roc_auc is restricted to binary classification task, one vs rest multiclassification parameter was used.

When the model was trained with the statistical data (metadata), the model performance score [AUC] which was the mean of best scores over all folds for each class was observed to be 0.93217. When used with the test data to predict accuracy, the result produced accuracy score of **0.79125 (public: 0.79698, private: 0.78549)** with some overfitting. To improve the accuracy, the length feature for all three accelerations were dropped (least important as seen in chi-square test) and the model performance score was improved to *0.93455* and, also gave better prediction results with best accuracy of **0.81838 (public: 0.81886, private: 0.81790)** for Logistic Regression with minimal overfitting. The penalization factor was changed to 10, with same data as in previous test, to check if any improvement, but the prediction result accuracy did not improve – **0.81572 (public: 0.81509, private: 0.81635)** which was closer to the previous accuracy, and this can be evident by the fact here the model performance score was *0.93451*.

Further parameter changes were applied with different cross validation values (k-fold), penalization factors and sequential feature selection also being tried but were insufficient to improve the best accuracy of **0.81838** for Logistic Regression. Many factors could be responsible for this, starting with insufficient features in data or need for more data samples. Also, since logistic regression assumes linear relationships between predictors and the response variable, which means it can only model simple decision boundaries. If the decision boundary is non-linear, logistic regression may not be able to capture it accurately. Also it is susceptible to outliers which were present in our data.

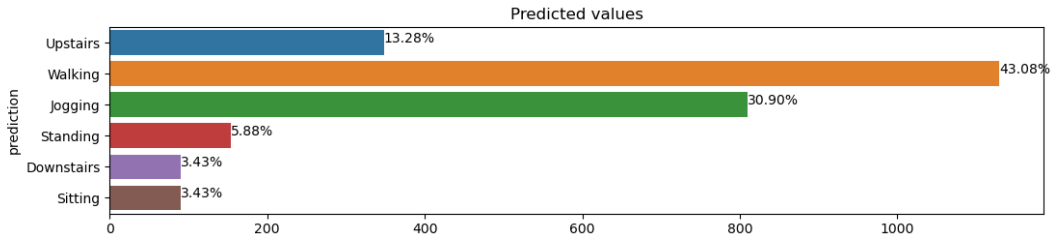

Figure 10. Logistic Regression best prediction activities count

SUPPORT VECTOR MACHINE (SVM)

Due to non-linearity in the data, SVM was used next due to its added advantage of better performance over high-dimensional non-linearly separable data. It is also more robust to outliers compared to logistic regression and provide flexibility over choosing kernel functions thus better customization of models.

Hyperparameter tuning and 5-fold cross validation was performed using 'GridSearchCV'. 'GridSearchCV' performs a search over all possible combinations of hyperparameters and evaluates the performance of each combination using cross-validation and best performance is selected based on the scoring metric.

The parameters were tested for range a of values to find the best fit, avoid overfitting and improve accuracy. Major parameter include –

- the penalty parameter of error term 'C' which controls the trade-off between achieving a low training error and a low testing error,
- the kernel parameter to specify the kernel functions which can be radial basis function, liner or polynomial,
- kernel specific parameters like gamma parameter for rbf to control the distance of the influence of a single training point, degree parameters to specify the degree of polynomial functions.

5-fold cross validation with 'roc_auc_ovr' scoring metric was applied for reducing overfitting and better multi-class classification results. The prediction score did improve to approximately **0.82**, however more tuning seemed essential for better results along with additional features added to the metadata. Another reason for low accuracy could be due to SVM being not suitable for large datasets. Also, SVM does not perform well when noise is present i.e. target classes are overlapping, which can be seen in our data.

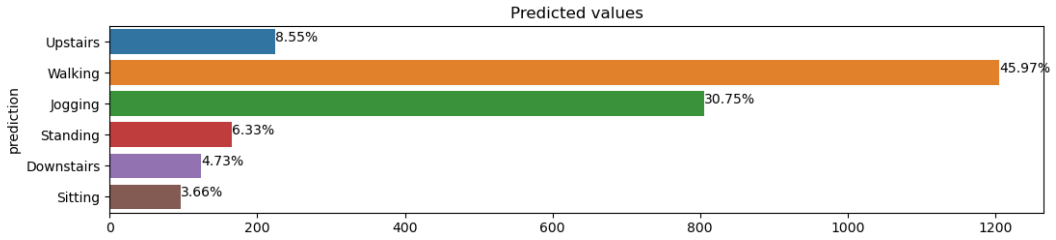

Figure 11. SVM Model Prediction Values

RANDOM FOREST CLASSIFIER

Another popular classification algorithms used was the Random Forest classifier to improve the existing prediction accuracy.

The parameters were tested for range a of values to find the best fit, avoid overfitting and improve accuracy. Parameters for the model included –

- ('n_estimators') Number of trees in the forest were tested for ranges from 20 to 30 with increments of 5.
- ('max_depth') Maximum depth of the tree was tested for ranges from 20 to 35 with increments of 5. The deeper the tree, the more complex it is and the more it can fit the training data.
- ('min_samples_leaf') The minimum number of samples required to be at a leaf node were tested for leaf sizes from 2 to 18 with increments of 2. A smaller leaf size can make the model more flexible, but too small of a leaf size can lead to overfitting.
- ('max_features') Maximum number of features that are considered for splitting at each node were tested for values of 0.1, 0.3, 0.5, 0.7, 0.9, and 1. The higher the number, the more likely the algorithm is to find a high-quality split, but also the more likely it is to overfit hence need lower values.
- ('bootstrap') Boolean value that determines whether bootstrap samples are used when building trees was tested for both True and False. If True, bootstrap samples are used, which means that some samples will be repeated in different trees. This can help improve the accuracy of the model, but it also increases the variance. If False, the entire dataset is used to build each tree, which can lead to lower accuracy but lower variance.

5-fold cross validation was used and 'roc_auc_ovr' scoring metric was applied since it was multiclass classification.

The model once trained and optimal parameters attained, the (grid search) best model performance score was observed to *0.94*. However, although while predicting too much overfitting was observed. The reason behind this result may well be because random forest struggle with imbalanced datasets, where one class has significantly more examples than the other which can be seen in the HAR data set. The model can be biased towards the majority class, leading to poor performance on the minority class. Also, random forest can be sensitive to noise and outliers in the data, which can affect the splitting process and lead to poor performance.

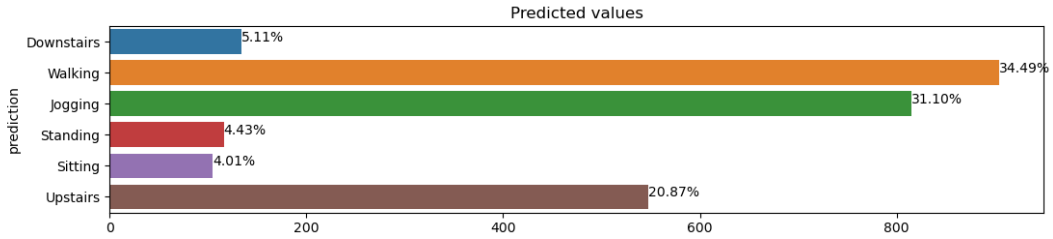

Figure 12. Random Forrest Prediction Values

DEEP FULLY CONNECTED NEURAL NETWORKS (DENSE)

In search of better prediction accuracy, the last model trained and tested with the statistical features used the deep fully connected neural network (dense) layers, where neurons of the layers are connected to the predicting layer with hidden layers allowing network to learn more abstract and complex features from the input data.

The model included first 2 layers with 128 and 64 neurons each with ReLU activation. One advantage of ReLU is that it does not active all neurons at same time and helps to prevent exponential growth in computational power required to operate the network. The final layer consisted of the 6 neurons with softmax activation for output. Dropouts were introduced to reduce overfitting.

The model was trained with batch size of 128 with 20% validation split, for 20 epochs. Lower epochs were used because the statistical data had lesser samples compared to the time series data. The model produced a very high validation loss which indicated better tuning the model. When additional layers with additional neurons were introduced, more loss and less accuracy than the previous model was observed thereby low prediction accuracy on test data.

A different model architecture with optimal tuning was essential for better results with Dense however, with several existing studies indicating direct use of time series data instead of statistical extracted data resulting in better prediction accuracy with WISDM HAR, the modelling approach was switched to time series data.

CONVOLUTIONAL NEURAL NETWORKS (CNN)

Best test data prediction score was obtained using a combination of convolutional neural network (CNN) layers and deep fully connected neural network (dense) layers, applied to the time series data (signals data), as creating further models using statistical features data (metadata) deemed unproductive for better test data prediction score. The objective behind using CNN was because CNN has been shown to be effective for processing sequential data and CNN layers are specifically designed to extract local features from input data, and each layer can learn different levels of abstraction. By stacking multiple CNN layers on top of each other, the model can extract more complex and abstract features from the input data, which can lead to better performance. Also, adding multiple dense layers helps the model learn non-linear relationships between features extracted from the input data, learn more complex representations, and improve the ability to discriminate between the different classes (activities).

The data (independent variables (test and train)) was segmented and re-shaped to 3 dimensions with 100 timesteps and the 3 features for each timestep so that the correct input shape could be provided to the first convolutional 1d layer. Also as mentioned earlier, the dependent variable (train) vector was re-shaped to binary matrix for of 6 activity columns.

Once data was prepared, different models with CNN and Dense layers were tested with different hyperparameters.

For the first model, the zero filled imputed time series data was used. Two convolutional (Conv1D) layers were used, first layer consisted of 64 filters while second of 32. Filters decides number of features to be extracted from the input data and were chosen accordingly after trying different filters for the same, as increasing number of filters improves model's accuracy up to a certain point, after which accuracy starts to decrease due to overfitting. Weight regularization [13] L2 regularization was used as it helps to prevent overfitting by adding a penalty term to the loss function that encourages the weights to be small. This can be useful in scenarios where the model has a large number of parameters and there is a risk of overfitting. The regularization of $1e^{-3}$ was applied for kernel and $1e^{-4}$ used for the bias on both the CNN layers. Kernel size helps capture longer-term temporal relationships and was set to 3 for both layers. Smaller kernel size helps capture details in smaller features better. Relu activation was used to introduce nonlinearity into the model.

Dropout of 0.4 was introduced to prevent overfitting as it works by randomly dropping out a proportion of the nodes in a neural network during training and encourages each neuron to be useful on its own, rather than relying on other neurons to correct its errors. Further, Maxpooling 1D with pool size of 2 was used to perform the down sampling operation to extract the most salient features of the input signal while reducing its dimensionality. This can also help to reduce overfitting and improve the efficiency of the subsequent layers in the network. Further Flatten was introduced to convert multidimensional input tensor into a one-dimensional output tensor so that it can be fed to the dense layer.

The first dense layer was added with 512 neurons. Increasing the number of neurons in the dense layer can increase the capacity of the model to learn complex patterns in the data, but it can also lead to overfitting. On the other hand, reducing the number of neurons can simplify the model and reduce the risk of overfitting, but it may result in underfitting and poor performance. Another dropout was introduced, and model architecture ended with the dense layer to get the desired result i.e., classifying the 6 activities.

For compiling the model, the adam optimizer with the default learning rate was used. It affects the speed and stability of the training process. Being a multi-class classification problem, Categorical Crossentropy loss function was used with accuracy metric for evaluating the model performance. The model was fit with the training data for 40 epochs with a batch size of 20 and 20% validation split. Increasing the number of epochs can lead to better model performance, but can also increase the risk of overfitting. Smaller batch size can potentially lead to better

results but training process can be computationally expensive and time consuming. Validation split can affect the model's ability to generalize to new data, as it determines how much of the training data is used for validation.

```
Model: "sequential_64"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_170 (Conv1D)          (None, 98, 64)            640
_________________________________________________________________
conv1d_171 (Conv1D)          (None, 96, 32)            6176
_________________________________________________________________
dropout_22 (Dropout)         (None, 96, 32)            0
_________________________________________________________________
max_pooling1d_114 (MaxPoolin (None, 48, 32)            0
_________________________________________________________________
flatten_37 (Flatten)         (None, 1536)              0
_________________________________________________________________
dense_54 (Dense)             (None, 512)               786944
_________________________________________________________________
dropout_23 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_55 (Dense)             (None, 6)                 3078
=================================================================
Total params: 796,838
Trainable params: 796,838
Non-trainable params: 0
_________________________________________________________________
```

Figure 13. CNN Model 1 Summary

The model architecture is summarised in above figure. Due to multiple CNN layers and the 512-neuron dense layer, the model formed was complex which is evident with the total parameters value *796,838*. The model was checked for validation loss and accuracy with each epoch. The validation accuracy fluctuated approximately between 0.72 to 0.89 during each epoch while validation loss fluctuated between 0.40 to 0.80.

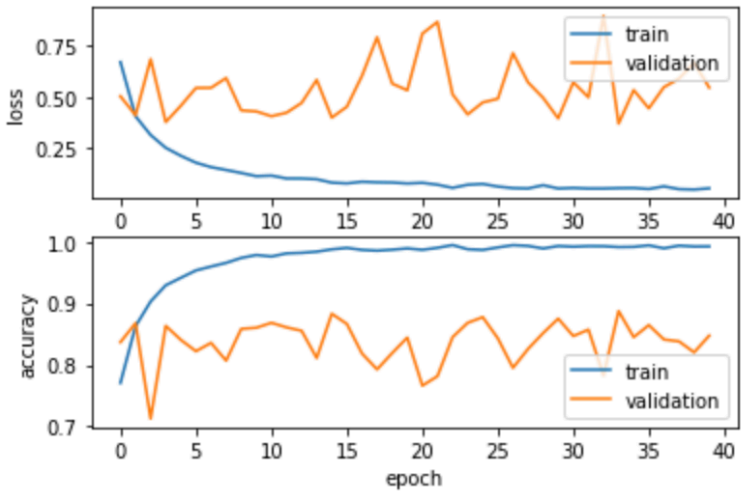

Figure 14. CNN Model 1: (Zero filling imputation) Training-Validation Loss, accuracy curves

When used on the test data, the prediction accuracy considerably increased to **0.89345 (Public: 0.90264, Private: 0.88425)**, however overfitting was observed which can be related due the fluctuations seen in the validation loss when training the data (Fig. 14).

With the same model architecture, the forward filled imputed time series data was trained and tested, and this model provided slightly better prediction accuracy of **0.89691 (Public: 0.90339, Private: 0.89043)** with less overfitting observed which is evident due to lower fluctuations in validation loss and accuracy (Fig. 15). This suggested that forward filled imputed time series data was more useful than the zero filled series data. So further models were tested using forward filled imputed time series (signals) data.

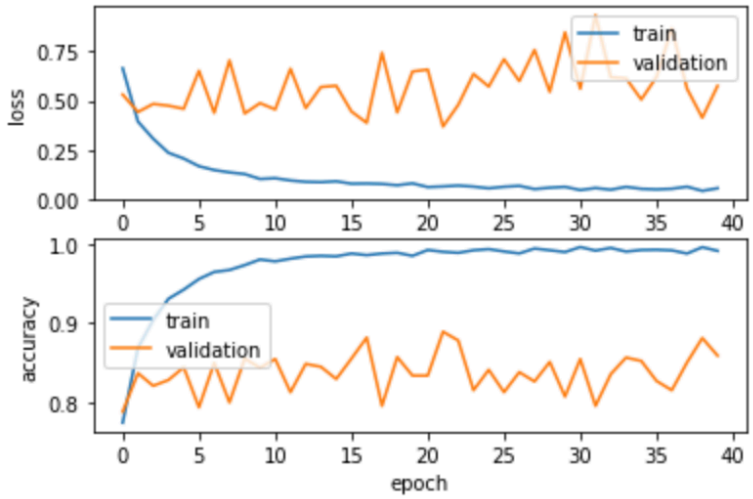

Figure 15. CNN Model 1: (Forward filling imputation) Training-Validation Loss, accuracy curves

To reduce the complexity of the model and improve prediction, further models with different layers and hyperparameters were tested. Better prediction scores were observed when additional convolution layer was added with different kernel size. The first CNN layer was same as previous model, however for the second and third layers, kernel size was set to 5 in attempt to capture longer-range dependencies in the input data. Also, the dense layer neurons was reduced to 256 for further optimisation. The adam optimizer learning rate was set to 5e-4 . By using a smaller learning rate, the optimizer takes smaller steps and is less likely to overshoot the optimal solution. The overall model proved to be more efficient and less complex with total parameters reduced to *189,894* (Fig. 16).

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_34 (Conv1D)           (None, 98, 64)            640

conv1d_35 (Conv1D)           (None, 94, 32)            10272

dropout_34 (Dropout)         (None, 94, 32)            0

max_pooling1d_22 (MaxPoolin  (None, 47, 32)            0
g1D)

conv1d_36 (Conv1D)           (None, 43, 32)            5152

dropout_35 (Dropout)         (None, 43, 32)            0

max_pooling1d_23 (MaxPoolin  (None, 21, 32)            0
g1D)

flatten_12 (Flatten)         (None, 672)               0

dense_24 (Dense)             (None, 256)               172288

dropout_36 (Dropout)         (None, 256)               0

dense_25 (Dense)             (None, 6)                 1542

=================================================================
Total params: 189,894
Trainable params: 189,894
Non-trainable params: 0
```

Figure 16. CNN Model 2 Summary

The model was fit for different batch sizes, and validation splits with 100 epochs and when applied to test data, resulted in consistent above **0.90** prediction scores. The prediction score improved to **0.91775 (Public: 0.92075, Private: 0.91435)** (final Kaggle competition leaderboard score) with validation split of 10% and batch size of 32 with validation loss minimized and accuracy was maximized (Fig. 17).

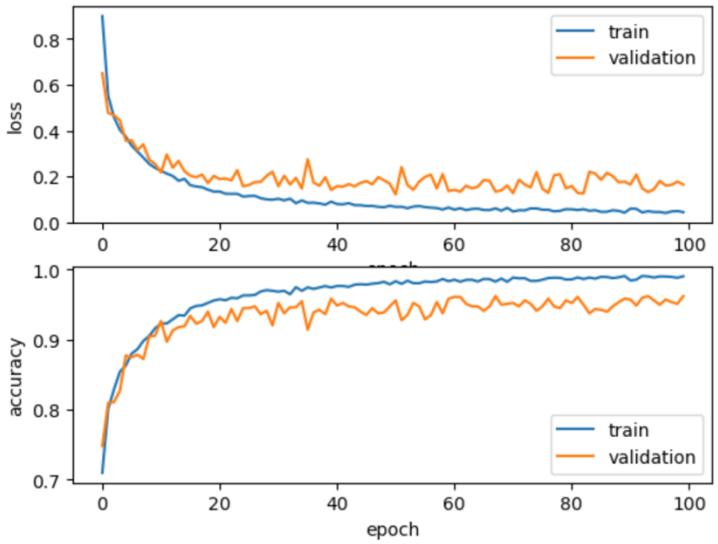

Figure 17. CNN Model 2 (Batch Size 32, Validation 0.10) Training-Validation Loss, accuracy curves

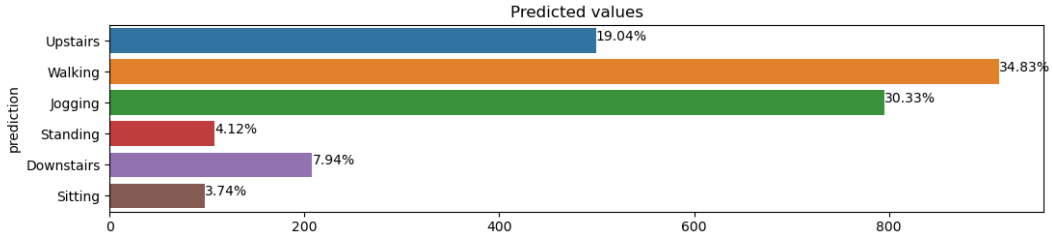

Figure 18. CNN Model 2 (Batch Size 32, Validation 0.10) Prediction Values

However, when multiple iterations were run (re-training for 30 iterations in a loop, each 100 epochs) on the same model with different batch sizes and validation splits starting from a lower (as low as 5, 0.2) to higher values during each iteration, the best fit model was obtained. The reason behind choosing this approach was because of the stochastic nature of the neural networks and possibility of same model getting more opportunities to learn from the training data and adjust its parameters accordingly to capture more complex patterns and make better predictions. The model prediction scores were saved during each iteration (Models could be coded to be saved as well). As, each iteration progressed the training loss started to get minimal and constant - least validation loss and most validation accuracy was observed up to point when the validation splits increased, it stated getting worse. During this process, one of the best fit models with least overfitting observed produced prediction result of **0.92445 (Public: 0.92528, Private: 0.92361).** The best overall prediction score was observed at batch size of 25 and validation of 0.24 with score of **0.929375 (Public: 0.93283, Private: 0.92592)**. The model at that iteration could attain validation accuracy between 0.94-0.98 (Fig. 19). Some overfitting was observed which may be due to training data used being incomplete without actual observed values (completed by forward filled).

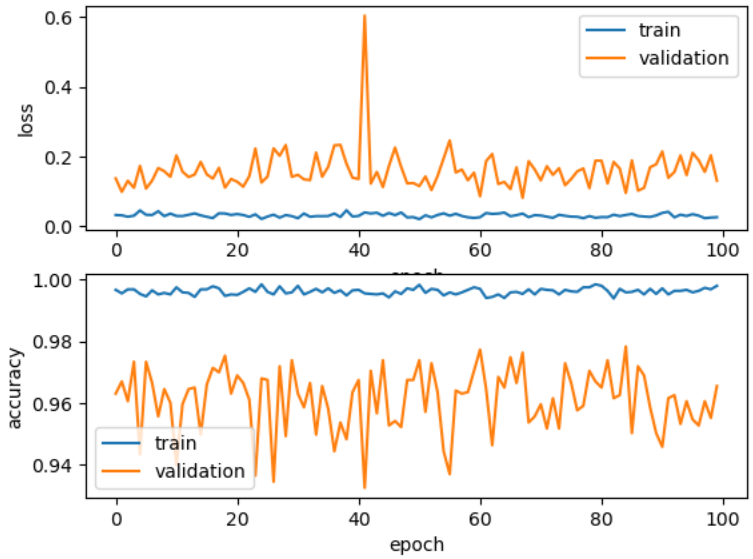

Figure 19. CNN Model 2 (Batch Size 25, Validation 0.24) Training-Validation Loss, accuracy curves

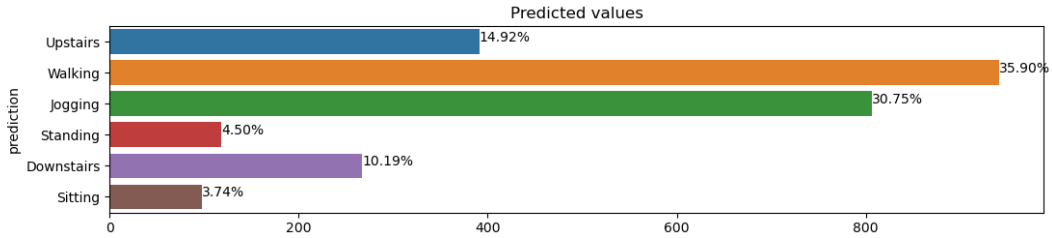

Figure 20. CNN Model 2 (Batch Size 25, Validation 0.24) Prediction Values

Although, best possible score was attained, the approach to find best fit using multiple iterations was time consuming and could have been computationally heavy for certain computing hardware hence has room for improvements. Also due to the randomness involved in neural network training if some additional data is provided to be tested may not guarantee same prediction score and might need retraining.

Some additional algorithm like LSTM was also used in combination with CNN and Dense, however a better accuracy was not attained and needed better hyperparameter tuning.

As this project was part of Kaggle Competition, total of 208 model submissions were carried out and best score was attained after competition closure. As we switched from statistical data to time series data with neural networks, the prediction rate started to increase. The following figure shows our model submission scores improved gradually throughout the competition.

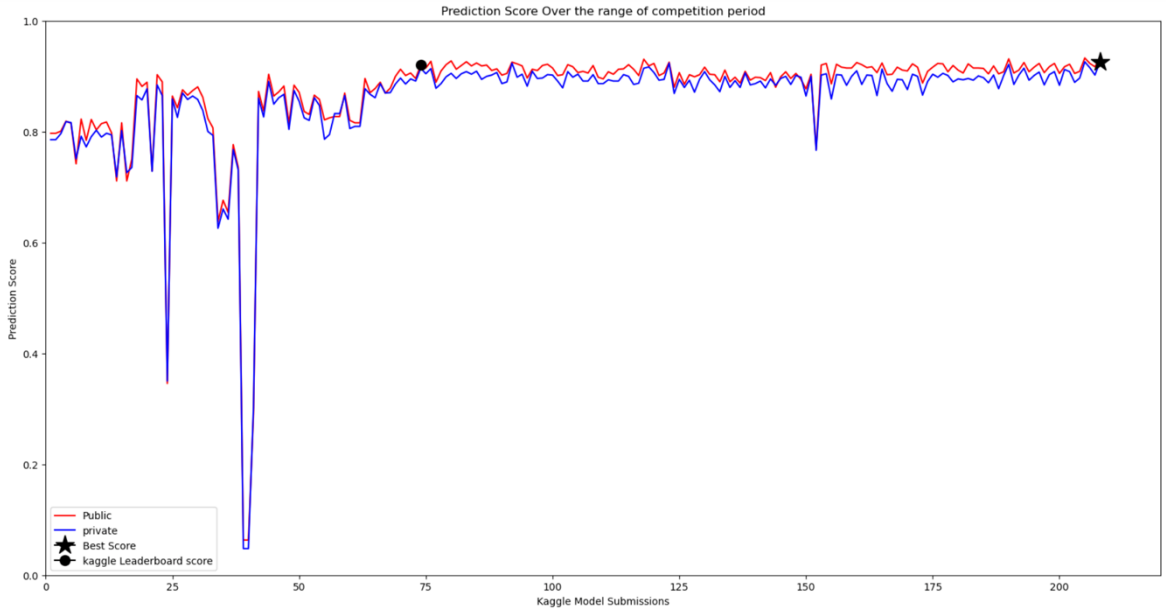

Figure 21. Kaggle Submission Scores throughout the competition

CONCLUSION

The HAR data provided was trained with several models using both statistical data and time series data, with best prediction accuracy of **0.93** seen with Neural network (CNN + Dense) algorithm application on the time series data. However, there is room for significant improvements in modelling. The model can still be explored to reduce the complexity by removing one of the dense layers and replacing it with sufficient CNN layers while maintaining the accuracy. Dropouts can be minimized or removed by replacing with other sufficient regularizations for reducing overfitting. LSTM and CNN has been found to be successful with WISDM time series data [14] so further modelling with combination of these algorithms with appropriate hyperparameter tuning may prove successful. In terms of the available statistical data, using the provided features alone are not sufficient for better accuracy and extracting additional features from the time series becomes necessary. Also, existing studies for HAR show, using more complete data samples along with additional data like gyroscope features can aid to better accuracy of the model for better human activity recognition.

REFERENCES

1. Pham C et al (2020), SensCapsNet: deep neural network for non-obtrusive sensing based human activity recognition. IEEE Access 8:86934–86946. URL: https://doi.org/10.1109/ACCESS.2020.2991731
2. Qi J, Wang Z, Lin X, Li C (2018) Learning complex spatio-temporal configurations of body joints for online activity recognition. IEEE Trans Hum Mach Syst 48(6):637–647. URL: https://doi.org/10.1109/THMS.2018.2850301
3. Zhu R et al (2019) Efficient human activity recognition solving the confusing activities via deep ensemble learning. IEEE Access 7:75490–75499. URL: https://doi.org/10.1109/ACCESS.2019.2922104
4. Wang K, He J, Zhang L (2019a) Attention-based convolutional neural network for weakly labeled human activities' recognition with wearable sensors. IEEE Sens J 19(17):7598–7604. URL: https://doi.org/10.1109/JSEN.2019.2917225
5. Phyo CN, Zin TT, Tin P (2019) Deep learning for recognizing human activities using motions of skeletal joints. IEEE Trans Consum Electron 65(2):243–252. URL: https://doi.org/10.1109/TCE.2019.2908986
6. Du Y, Lim Y, Tan Y (2019) A novel human activity recognition and prediction in smart home based on interaction. Sensors (Switzerland). URL: https://doi.org/10.3390/s19204474
7. Deep S, Zheng X (2019) Leveraging CNN and transfer learning for vision-based human activity recognition. In: 2019 29th international telecommunication networks and application conference ITNAC 2019, pp 35–38, 2019. URL: https://doi.org/10.1109/ITNAC46935.2019.9078016
8. Beddiar DR, Nini B, Sabokrou M, Hadid A (2020) Vision-based human activity recognition: a survey. Multimed Tools Appl 79(41–42):30509–30555. URL: https://doi.org/10.1007/s11042-020-09004-3
9. Ding H et al. (2015) FEMO: a platform for free-weight exercise monitoring with RFIDs. In: SenSys 2015—proceedings of 13th ACM conference on embedded networked sensor systems, pp 141–154. URL: https://doi.org/10.1145/2809695.2809708.
10. WISDM (Wireless Sensor Data Mining), (2013). URL: https://www.cis.fordham.edu/wisdm
11. Jason Brownlee, (2019), Machine Learning Mastery, URL: https://machinelearningmastery.com/feature-selection-with-categorical-data/
12. Maryam Banitalebi Dehkordi, Abolfazl Zaraki, Rossitza Setchi (2021) Optimal Feature Set for Smartphone-based Activity Recognition, Procedia Computer Science: Volume 192, pp 3497-3506: 1877-0509, URL: https://doi.org/10.1016/j.procs.2021.09.123.
13. Jason Brownlee, (2019), Machine Learning Mastery, URL: https://machinelearningmastery.com/how-to-reduce-overfitting-in-deep-learning-with-weight-regularization/
14. Xia, Kun & Huang, Jianguang & Wang, Hanyu. (2020). LSTM-CNN Architecture for Human Activity Recognition. IEEE Access. PP. 1-1., URL: 10.1109/ACCESS.2020.2982225