Fall Identification through Human Activity Recognition

Darren (A0216214U),
Daryl (A0190259Y),
Jiaxu (A0216256H),
Kelvin (A0218236H),
Sarah (A0222470R),
Zhi Hao (A0217500W)

Table of contents

1. Problem Statement
2. Research
3. Data Pre-processing
4. Model Training
5. Results
6. Final Model
7. Future Improvements

PROBLEM STATEMENT

01

How can we use Human Activity Recognition to identify falls?

[1]: https://www.cdc.gov/falls/data/fall-deaths.html
[2]: https://www.who.int/news-room/fact-sheets/detail/falls

684,000

Estimated yearly global deaths due to falls

[1]: https://www.who.int/news-room/fact-sheets/detail/falls

Fall detection through postural transitions

- Research shows that postural transitions are correlated with fall events
- Specifically, transitions to lying on the ground [1][2] can be used to detect fall events
- Dataset: Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set [3]
- In the Dataset provided: SIT_TO_LIE, STAND_TO_LIE
- Target demographic: Elderly, at-risk people

[1]: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8321307/
[2]: https://www.researchgate.net/publication/221313120
[3]:https://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions

RESEARCH

02

2.1 Past Works

Past Works

Author(s)	Methods (All on pre-processed dataset)	Accuracy (Note: These are ALL validation accuracies)
Jillani Soft Tech[1]	Support Vector Machines (SVM) Random Forest Classifier (RFC)	SVM: 0.988 RFC: 0.981
Essam Mohamed, Abdelrahman Raslan, Youssef Ahmed[2]	Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbors Classifier, Gaussian Naive Bayes (Gaussian NB), Decision Tree (DT), Random Forest Classifier (RFC), XGBoost	LR, LDA, XGBoost: >= 98.5% KNN (k=30, euclidean): 95.4% Gaussian NB: 66.8% DT/RFC: 94.3% & 97.6% (but with 100% training acc)

Sources:
[1] https://www.kaggle.com/code/jillanisofttech/human-activity-recognition-with-smartphones-99-acc
[2] https://www.kaggle.com/code/essammohamed4320/human-activity-recognition-scientific-prespective

Learning points:

While there are many different models being tested:

- The authors did a train_test_split on the given training data set, trained on the split and tested against what is the validation set.
- The model would thus have seen the same users in the validation set
  - The high final model accuracy results are thus inflated

The models are also not generalizable:

- No attempts at regularization in these notebooks
- Some models had 100% training accuracy (DT/RFC) and a validation accuracy that is lower (and we expect test accuracy to be much worse)

Past Works

Author(s)	Methods (All on pre-processed dataset)	Accuracy
Abeer Elmorshedy[3]	K-Nearest Neighbors Classifier (KNN), Random Forest Classifier (RFC), XGBoost, Gaussian Naive Bayes (Gaussian NB), Logistic Regression (LR), Support Vector Machines (SVM)	(Note: These are on the given test dataset) KNN: 92.2%, RFC: 91.4% XGBoost: 93.5% Naive Bayes: 87.5% LR: 96.3%, SVM: 95.8%
Fahad Mehfooz[4]	Deep Artificial Neural Networks (ANN)	(Note: The test dataset was used as validation for the ANN, then tested against test dataset again. Contamination) ANN: 95.2%

Sources:
[3] https://www.kaggle.com/code/abeerelmorshedy/human-activity-recognition
[4] https://www.kaggle.com/code/fahadmehfoooz/human-activity-recognition-with-neural-networks

Learning points:

We now see that when the model is tested against users it has not seen before (in test set):

- The model accuracy suffers quite significantly despite similar methodologies in training the models, compared to the previously seen works
- This affirms our suspicions that different users may have different patterns in the data.

There has not been many deep learning approaches, but the most prominent one validated the ANN on the test set, then tested against the test set to yield the final accuracy:

- Obvious signs of contamination (The model has seen users in the test data)

However, there is also still another unaddressed issue that we learned about after looking through some reference papers on this subject

2.2 Research Papers

Research Paper(s)

Main article:
A CNN-LSTM Approach to Human Activity Recognition by Ronald Mutegeki & Dong Seog Han [1]

Learning Points:

- The research paper refers specifically to the dataset we are using.
- The processed dataset is obtained from splitting the raw data by time window deriving additional statistics through heavy feature extraction.
- Performance on processed dataset tends to be great during modelling, however, it tends to be bad when used on real-world data.
- This is due to the temporal features lost when aggregated in feature engineering.
- CNN-LSTM was thus suggested to be used on the raw data and formed the basis for what would be our final model.

DATA PRE-PROCESSING

03

3.1 DATA EXPLORATION

Why Data Exploration?

- Exploratory Data Analysis (EDA) allows for visual and statistical analysis of our dataset.
- Check for any potential problems within the dataset (eg: Imbalanced Dataset, non-separable dataset etc.), resolved through **feature engineering**.
- Project goal largely revolves around a classification problem, **separable** dataset is ideal.

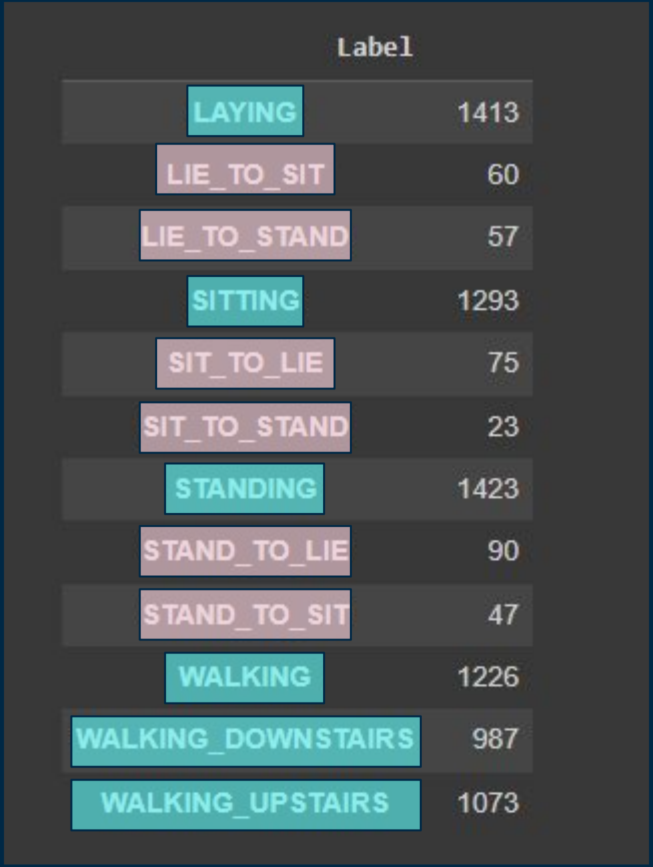
Class Imbalances

Analysis:

There are more instances of non-transitory activities (eg: SITTING, STANDING, WALKING etc.) compared to transitive activities (eg: LIE_TO_SIT, LIE_TO_STAND etc.).

Dataset is skewed towards non-transitory class.

Upsampling of the minority transitive class to increase the number of occurrences.



Label	
LAYING	1413
LIE_TO_SIT	60
LIE_TO_STAND	57
SITTING	1293
SIT_TO_LIE	75
SIT_TO_STAND	23
STANDING	1423
STAND_TO_LIE	90
STAND_TO_SIT	47
WALKING	1226
WALKING_DOWNSTAIRS	987
WALKING_UPSTAIRS	1073

Data Visualisation

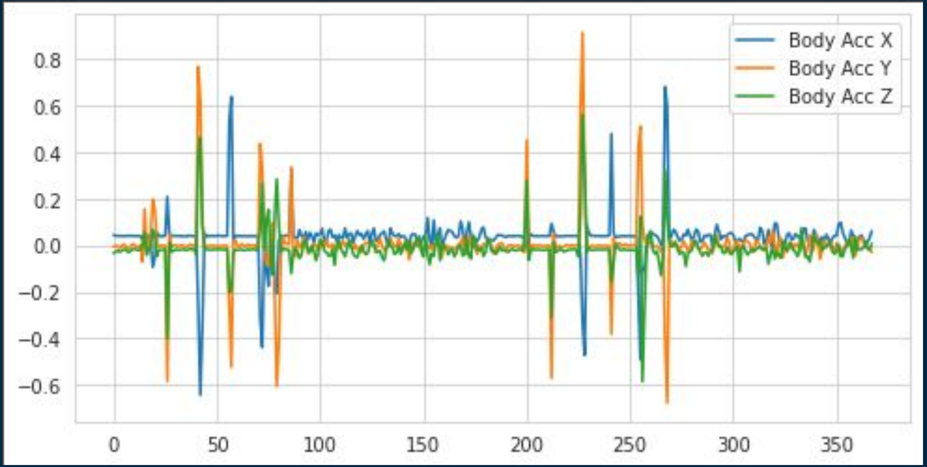
Analysis:

The mean gravitational acceleration of the body with time is measured in 3 dimensions (x,y,z dimension).

We can see that for a given timeframe, we can observe 2 periods in the signals recorded.

Each period contains all 12 classes of movements.

Therefore, as a time series, there is a pattern in the signals that should correspond to a respective class of movement. We can consider the usage of Neural Networks (NNs) to analyse the raw time series data.



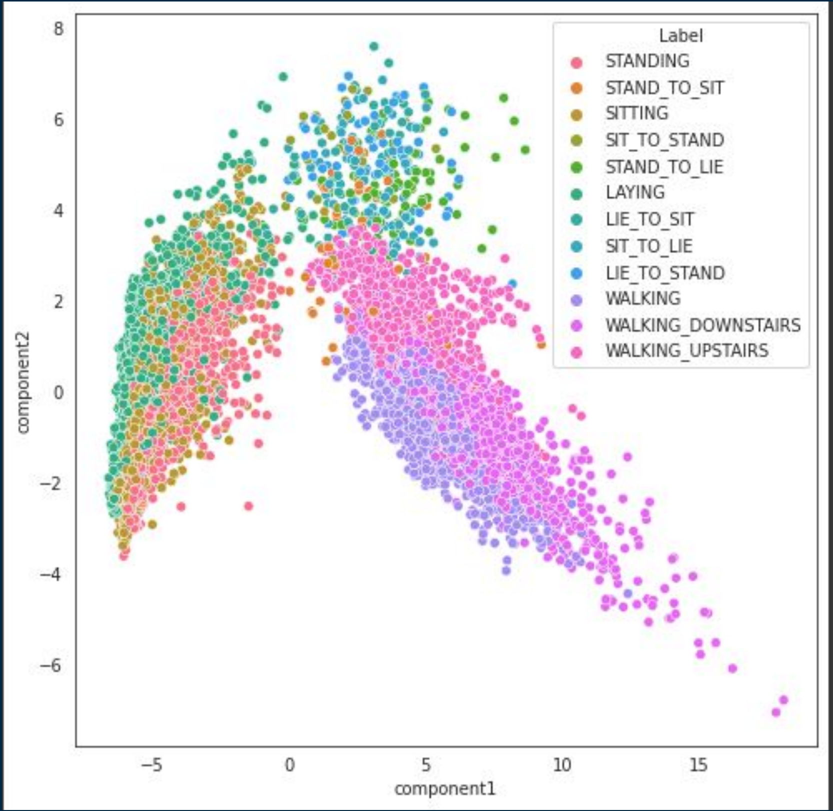
Principal Component Analysis

Analysis:

Clear distinguishment between
STATIC movements (STANDING, SITTING, LAYING),
DYNAMIC movements (WALKING, WALKING_UPSTAIRS),
and **TRANSITIVE** (STAND_TO_SIT, SIT_TO_STAND).

Classification between these movements is feasible.

However, specific classification within **STATIC** and **DYNAMIC** movements is not yet visibly separable, due to high correlation between the classes.

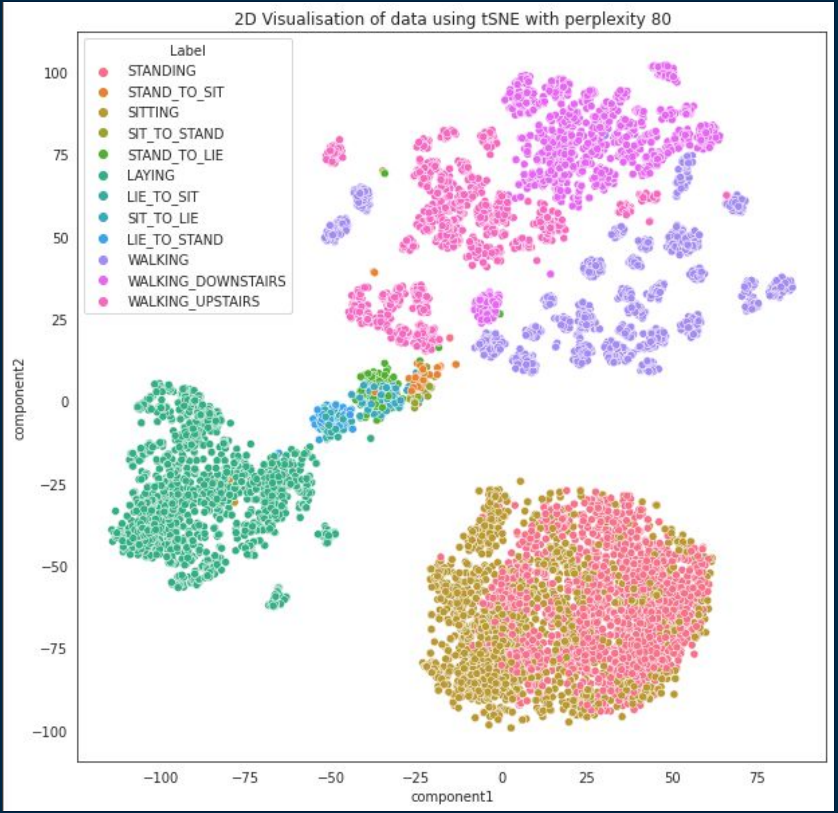


t-Stochastic Neighbour Embedding (tSNE)

Analysis:

Clear separation between **STATIC** movements.
In particular, it is able to distinguish between the "LAYING" cluster and "SITTING"/"STANDING" cluster.

Separability between **TRANSITIONAL** movements (eg: STAND_TO_LIE, STAND_TO_SIT etc.) and **DYNAMIC** movements (eg: WALKING, WALKING_DOWNSTAIRS etc.)



t-Stochastic Neighbour Embedding (tSNE)

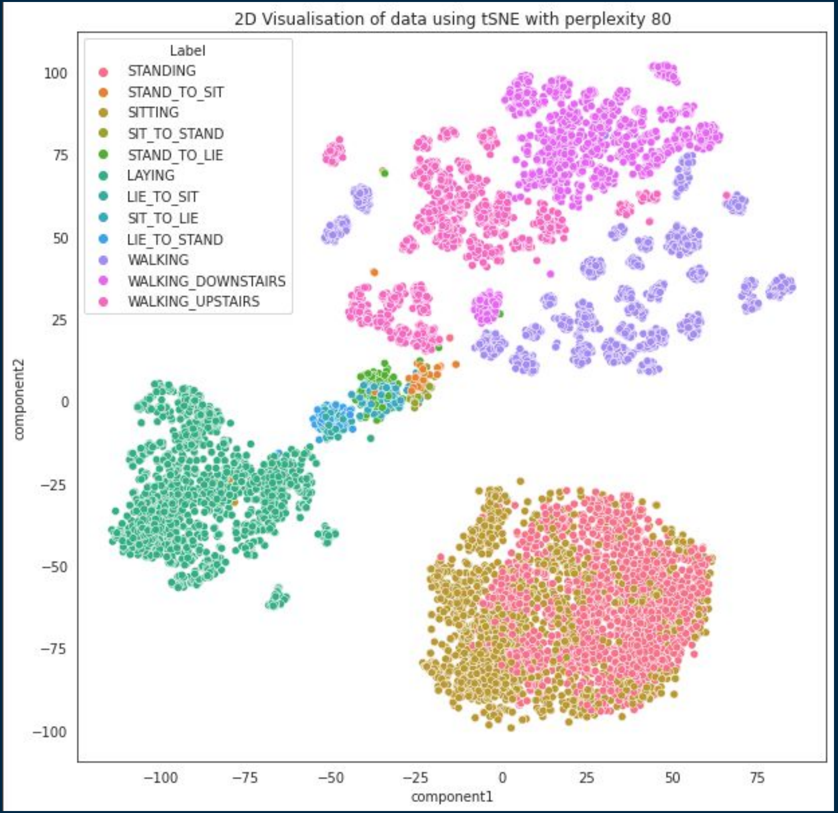
Analysis:

However, it is unable to distinguish between classes with high similarity clearly.

TRANSITIVE movement bidirectional pairs (eg: STAND_TO_LIE and LIE_TO_STAND) are very similar. "STANDING" and "SITTING" classes inseparable using tSNE, due to these 2 activities being highly stationary and similar.

For **DYNAMIC** classes, there is high variation between each class. Nevertheless, there is distinct separation between the 3 **DYNAMIC** classes.

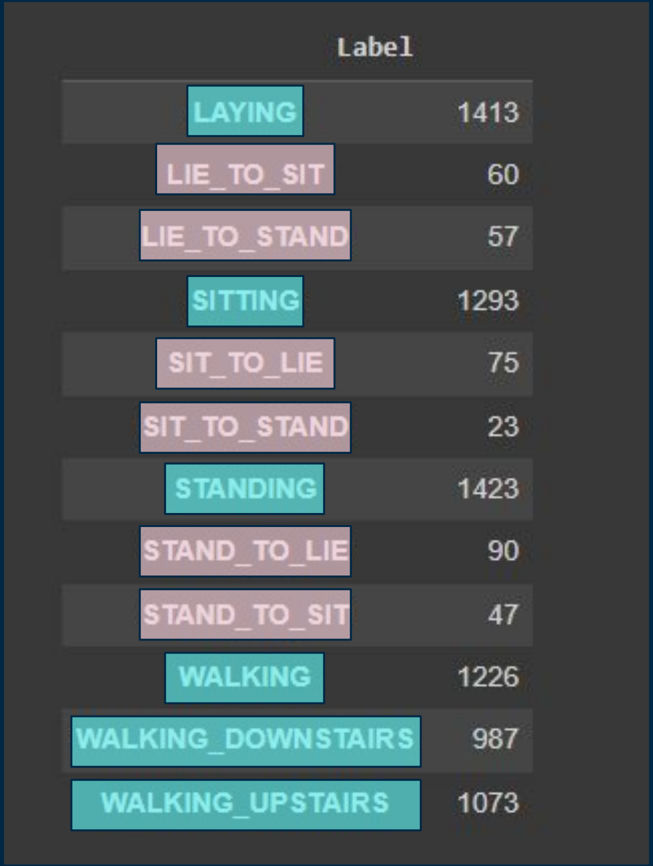
Nevertheless, as separation amongst all 12 classes is relatively good, it is feasible to consider the use of **SVMs** or a **Logistic Regression Classifier**



3.2 DATA PREPARATION

Resolving Class Imbalances

- Using pre-processed data
- Very low number of datapoints for transitional labels as compared to static labels.
- Upsampling was performed on the minority classes.



	Label
LAYING	1413
LIE_TO_SIT	60
LIE_TO_STAND	57
SITTING	1293
SIT_TO_LIE	75
SIT_TO_STAND	23
STANDING	1423
STAND_TO_LIE	90
STAND_TO_SIT	47
WALKING	1226
WALKING_DOWNSTAIRS	987
WALKING_UPSTAIRS	1073

Method-based Validation Split

- Based on experiment method[1]:
    - Some subjects generated training data, some generated test data
    - We followed this method, and further split validation data from training:
        1. Train-Val split of 0.7:0.3
        2. Separated the feature-extracted data by subject
        3. Separated the raw accelerometer data by subject
    - Prevents data snooping

[1]: https://doi.org/10.1016/j.neucom.2015.07.085

MODEL TRAINING

04

4.1 LOGISTIC REGRESSION CLASSIFIER

IMPLEMENTATION

1. Split the dataset into a 70-30 training/validation set, using train_test_split

2. Using RandomizedSearchCV, tune hyperparameters **C**, **penalty** and **solver**
   - C=1
   - Penalty = L1
   - Solver = SAGA

3. **5-fold** Cross Validation to check if model is overfitting
   - CV scores are consistent across all 5 folds, so there should not be major overfitting

PERFORMANCE OF MODEL

Metric	Score
(Micro) Accuracy	0.941
(Micro) Precision	0.942
(Micro) Recall	0.941
(Micro) F1-Score	0.940

Analysis:

We consider the micro-average to account for the imbalanced dataset.

The model performs unexpectedly well, given that the task is complex

Due to the features having been engineered (based on domain knowledge), we are able to produce a model that performs substantially well

However, the model is unable to generalize to other settings (since it uses statistics of raw signals recorded from this particular experiment) *(https://ieeexplore-ieee-org.libproxy1.nus.edu.sg/document/9065078)*

4.2 SUPPORT VECTOR MACHINE

SVM DATA

Data

Used balanced data with 561 features

Train-Test Split

Split by user id with 70% of users in train set and 30% of users in test set

Features

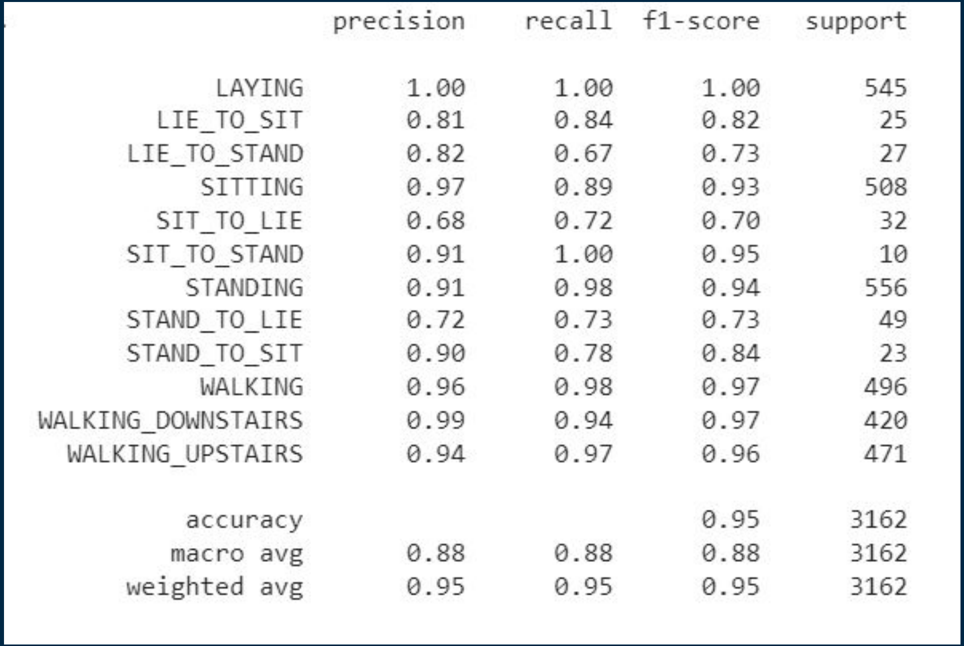
Used all 561 features

SVM RESULTS & ANALYSIS

- After doing hyper parameter tuning using grid search, the best model obtained is
  C: 100
  Gamma: 0.01

  The larger C value means that the margin is small. The smaller gamma value means that the shape of the peak of the rbf kernel is broader

- The SVM is generally able to classify most classes accurately, with transition classes being slightly lower in accuracy



```
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       545
        LIE_TO_SIT       0.81      0.84      0.82        25
      LIE_TO_STAND       0.82      0.67      0.73        27
           SITTING       0.97      0.89      0.93       508
        SIT_TO_LIE       0.68      0.72      0.70        32
      SIT_TO_STAND       0.91      1.00      0.95        10
          STANDING       0.91      0.98      0.94       556
      STAND_TO_LIE       0.72      0.73      0.73        49
      STAND_TO_SIT       0.90      0.78      0.84        23
           WALKING       0.96      0.98      0.97       496
WALKING_DOWNSTAIRS       0.99      0.94      0.97       420
  WALKING_UPSTAIRS       0.94      0.97      0.96       471

          accuracy                           0.95      3162
         macro avg       0.88      0.88      0.88      3162
      weighted avg       0.95      0.95      0.95      3162
```

4.3 DECISION TREE

DT DATA PRE-PROCESSING

Data

Used pre-processed data with 561 features

Train-Test Split

Split by user id with 70% of users in train set and 30% of users in test set

Features

Used a variety of combinations of features:

- All 561 features
- Top 75 features
- All the "max", "mean", etc features

DT RESULTS & ANALYSIS

- Using gridsearch with cross validation to tune hyperparameters, unlimited depth for both classifiers performed the best.
- The DT is generally able to classify most classes accurately, with transition classes being much lower in accuracy
- Random Forest outperformed DT on all fronts, including almost a 10% improvement in both macro and micro accuracy. This is likely because random forests are better able to learn non-linear decision boundaries and overfits to training data less, being more generalizable on test data it hasn't seen before

Decision Tree	Random Forest

Using all features
max_depth=None,
criterion="entropy"
	n_estimators=400,
	max_depth=None,
	max_features="sqrt",
	criterion="entropy"

Accuracy: 84% Precision: 72% Recall: 71% F1: 71%	Accuracy: 92% Precision: 83% Recall: 82% F1: 82%

4.4 ARTIFICIAL NEURAL NETWORK

ANN DATA PRE-PROCESSING

Data

Uses features extracted from time-series data (561 features)

Train-val-test Split

Split train by user id:
30% Validation
70% Training

Standardisation & Framing

Used StandardScaler to remove mean and scale to unit variance

ANN Tuning

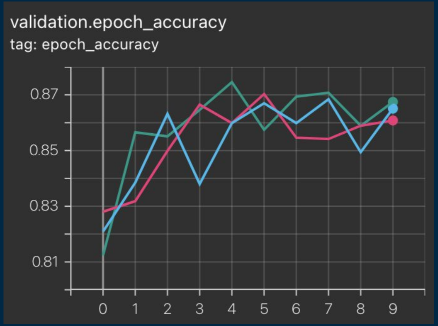
1. Baseline model:
   a. 64-neuron hidden layer, sigmoid activation
2. Perform tuning (n = 3):
   a. Optimizer
   b. Learning Rate
   c. Batch size
   d. Network shape
   e. Regularizers
3. Prioritize: Insights > Performance

[1]: https://stats.stackexchange.com/questions/369104/what-is-a-sensible-order-for-parameter-tuning-in-neural-networks

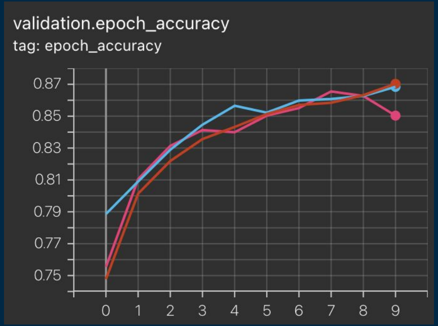
ANN Tuning > Learning Rate

- 0.001 chosen over 0.01 (close score) because manual inspection of graphs show diverging performance on validation → oscillating performance [1]

Rate = 0.01



Rate = 0.001



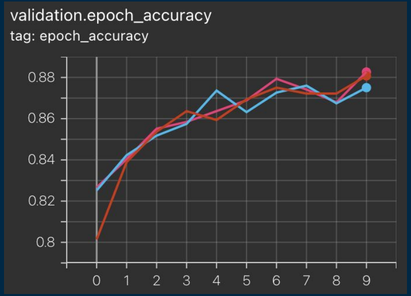
Learning Rate	Accuracy
0.1	0.724
0.01	0.871
0.001	**0.868**
0.0001	0.787
0.00001	0.495

[1]: Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press

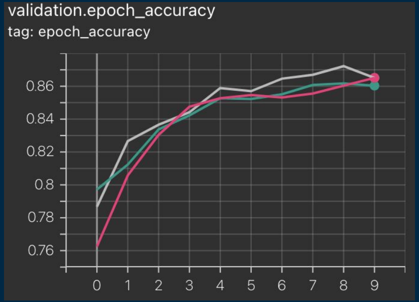
ANN Tuning > Batch Size

- No overfitting observed (due to low epoch count),
- Generally lower batch sizes tend to approach convergence faster, higher batch sizes approach more smoothly but slowly [1]
- choose batch_size of 32 for greater stability in gradient approach (measured by trend in validation graph)

Batch size = 16



Batch size = 32



[1]: https://arxiv.org/abs/1609.04836

Batch Size	Accuracy
2	0.885
4	0.881
8	0.882
16	0.875
32	**0.873**
64	0.858
128	0.846
256	0.813
512	0.787

Network shape

- # hidden layers: [1-5]
- # neurons in each layer: [32-512]
- Activations: [relu, sigmoid]
- Random Search over 100 trials
- Top 10 performing structures had similar results, selected least complex structure to mitigate overfitting to stochastic noise[1]:

Input: 561

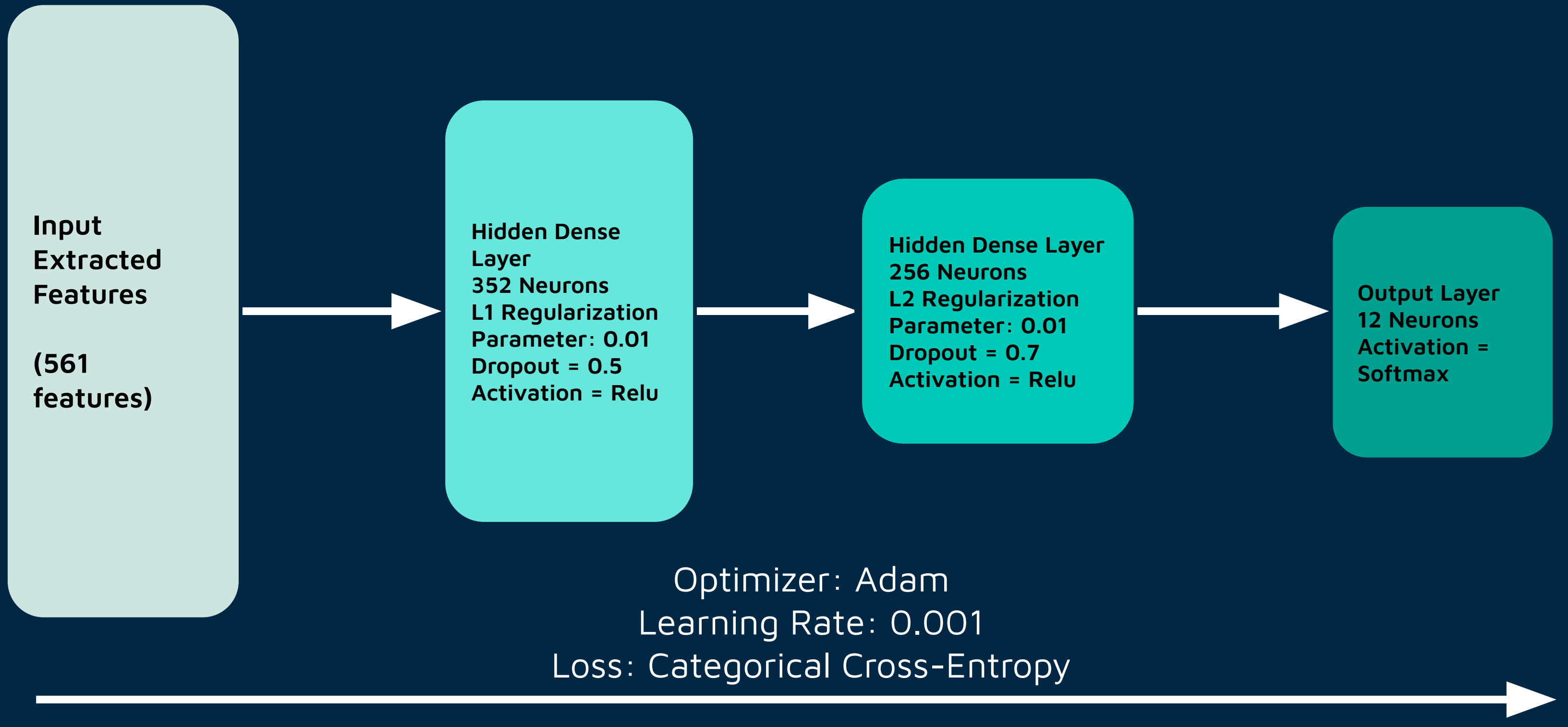
Layer 1: 352 , Relu

Layer 2: 256, Relu

Output: 12 (softmax)

[1]: https://link.springer.com/chapter/10.1007/978-3-642-10690-3_3

Optimized ANN



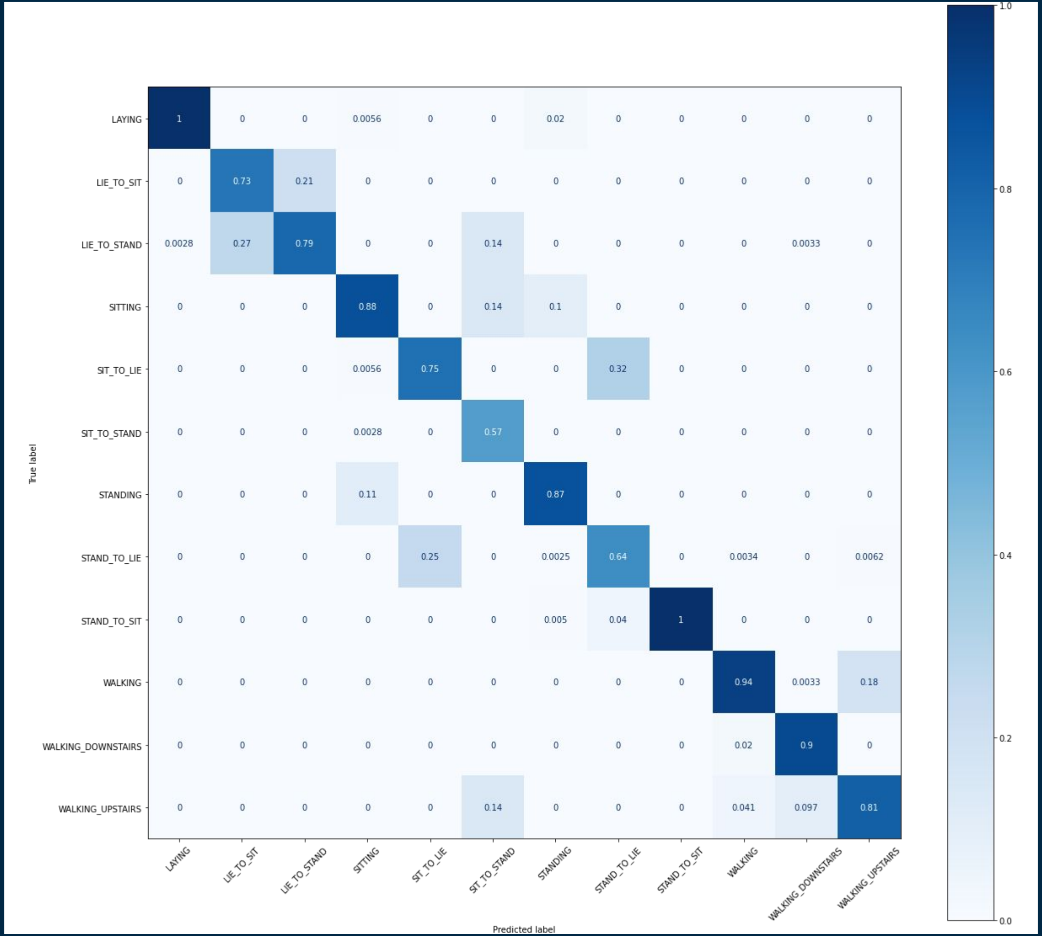
Optimizer: Adam
Learning Rate: 0.001
Loss: Categorical Cross-Entropy

Analysis

Accuracy: 0.802

Model has most trouble distinguishing transition postures

Sample imbalance



	LAYING	LIE_TO_SIT	LIE_TO_STAND	SITTING	SIT_TO_LIE	SIT_TO_STAND	STANDING	STAND_TO_LIE	STAND_TO_SIT	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	1	0	0	0.0056	0	0	0.02	0	0	0	0	0
LIE_TO_SIT	0	0.73	0.21	0	0	0	0	0	0	0	0	0
LIE_TO_STAND	0.0028	0.27	0.79	0	0	0.14	0	0	0	0	0.0033	0
SITTING	0	0	0	0.88	0	0.14	0.1	0	0	0	0	0
SIT_TO_LIE	0	0	0	0.0056	0.75	0	0	0.32	0	0	0	0
SIT_TO_STAND	0	0	0	0.0028	0	0.57	0	0	0	0	0	0
STANDING	0	0	0	0.11	0	0	0.87	0	0	0	0	0
STAND_TO_LIE	0	0	0	0	0.25	0	0.0025	0.64	0	0.0034	0	0.0062
STAND_TO_SIT	0	0	0	0	0	0	0.005	0.04	1	0	0	0
WALKING	0	0	0	0	0	0	0	0	0	0.94	0.0033	0.18
WALKING_DOWNSTAIRS	0	0	0	0	0	0	0	0	0	0.02	0.9	0
WALKING_UPSTAIRS	0	0	0	0	0	0.14	0	0	0	0.041	0.097	0.81

4.5 RECURRENT NEURAL NETWORK

RNN DATA PRE-PROCESSING

Data

Used raw signal data instead of preprocessed features because RNNs will do automatic feature engineering

Train-Test Split

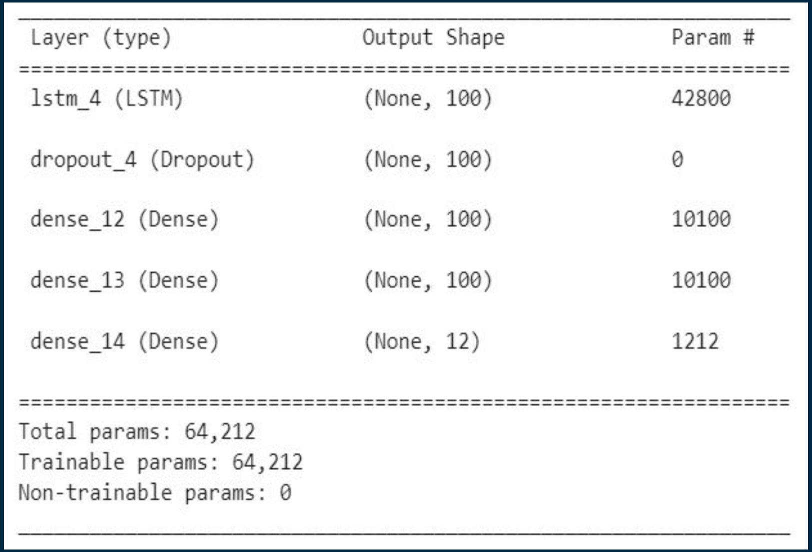
Split by user id with 70% of users in train set and 30% of users in test set.

Standardisation & Framing

Used StandardScaler to center values about zero as extreme values would affect the performance of the RNN

LSTM SLIDING WINDOW

- Reshape input data into 3D tensor using sliding window
- Raw data is first split by user before applying window
- Window size is 250 data points and window is moving by 50 data points. This window size is chosen to obtain enough data points and also to ensure there is not too much overlap between the windows because actions typically do not change too quickly.
- Activity labels are generated based on the mode activity in the window

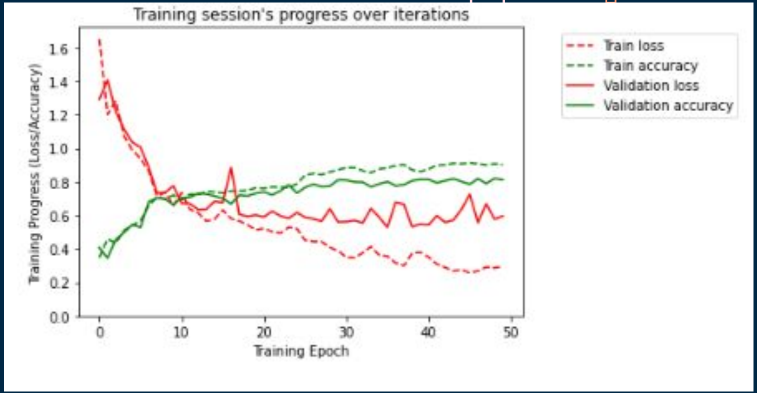


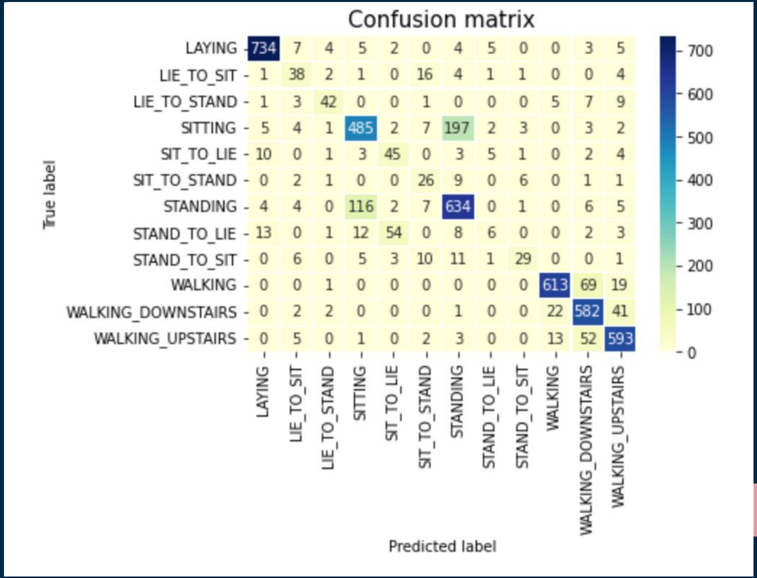
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 100)               42800

dropout_4 (Dropout)          (None, 100)               0

dense_12 (Dense)             (None, 100)               10100

dense_13 (Dense)             (None, 100)               10100

dense_14 (Dense)             (None, 12)                1212

=================================================================
Total params: 64,212
Trainable params: 64,212
Non-trainable params: 0
_________________________________________________________________
```

MODEL ARCHITECTURE FOR RNN

LSTM RESULTS & ANALYSIS

- Test Accuracy: 0.814
- The model finds it difficult to differentiate between standing and sitting because both actions are rather stationary, making it hard to tell the difference between the two
- This does not come as a surprise as seen from the tSNE map previously





	LAYING	LIE_TO_SIT	LIE_TO_STAND	SITTING	SIT_TO_LIE	SIT_TO_STAND	STANDING	STAND_TO_LIE	STAND_TO_SIT	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
LAYING	734	7	4	5	2	0	4	5	0	0	3	5
LIE_TO_SIT	1	38	2	1	0	16	4	1	1	0	0	4
LIE_TO_STAND	1	3	42	0	0	1	0	0	0	5	7	9
SITTING	5	4	1	485	2	7	197	2	3	0	3	2
SIT_TO_LIE	10	0	1	3	45	0	3	5	1	0	2	4
SIT_TO_STAND	0	2	1	0	0	26	9	0	6	0	1	1
STANDING	4	4	0	116	2	7	634	0	1	0	6	5
STAND_TO_LIE	13	0	1	12	54	0	8	6	0	0	2	3
STAND_TO_SIT	0	6	0	5	3	10	11	1	29	0	0	1
WALKING	0	0	1	0	0	0	0	0	0	613	69	19
WALKING_DOWNSTAIRS	0	2	2	0	0	0	1	0	0	22	582	41
WALKING_UPSTAIRS	0	5	0	1	0	2	3	0	0	13	52	593

4.6 CONVOLUTIONAL NEURAL NETWORK

CNN DATA PRE-PROCESSING

Data

Raw Signal Data

Train-Test Split

Split by user id with 70% of users in train set and 30% of users in test set

Standardisation & Framing

Used StandardScaler to center values about zero

CNN HYPERPARAMETERS, RESULTS & ANALYSIS

- The hyperparameters tuned are:
  - Number of layers (convolutional/ dense)
  - Number of filters/ neurons
  - Dropout layers and dropout rate

- Through model tuning, and comparing against a base model
  - The tuned model performed better in all aspects.

Base model	Tuned model

Conv2D(16, (2,2), ReLU)
Flatten()
Dense(12)
	Conv2D(**16**, (2,2), ReLU)
	Dropout(**0.1**)
	Conv2D(**8**, (2,2), ReLU)
	Dropout(**0.5**)
	Dense(**64**)
	Dropout(**0.2**)
	Flatten()
	Dense(12)

Accuracy: 84% Precision: 79% Recall: 75% F1: 76%	Accuracy: 88% Precision: 81% Recall: 79% F1: 79%

EVALUATION OF RESULTS

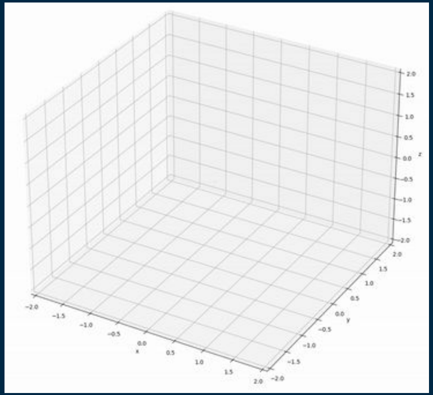
05

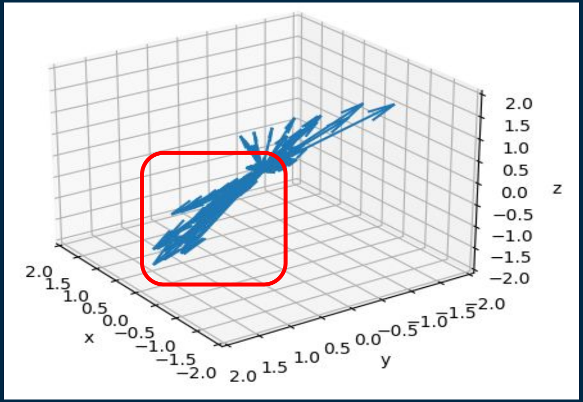
RESULTS OVERVIEW

Models	**Results (Accuracy)**	**Recall**	**Precision**	**F1**
Logistic Regression	0.94	0.83	0.87	0.85
SVM	0.95	0.88	0.88	0.88
Decision Tree/Random Forests	0.92	0.82	0.82	0.83
ANN	0.90	0.79	0.82	0.80
CNN	0.88	0.79	0.81	0.79
RNN	0.81	0.75	0.78	0.76

NN Visualize transitional actions

- Allows for comparison of classified states
- NN is able to compare peaks in movement in order and rate of change
- This shows the rapid change in gradient recognised by the NN





FINAL MODEL

06

6.1 CNN-LSTM NETWORK

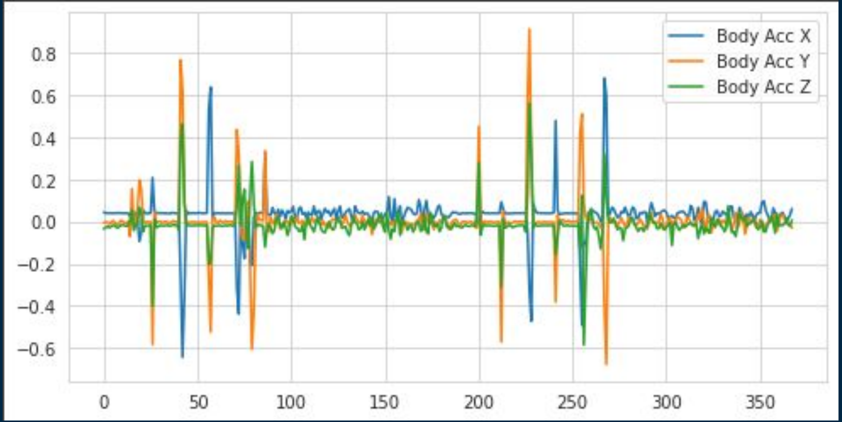
INSPIRATION

Key points in processing on raw signals:

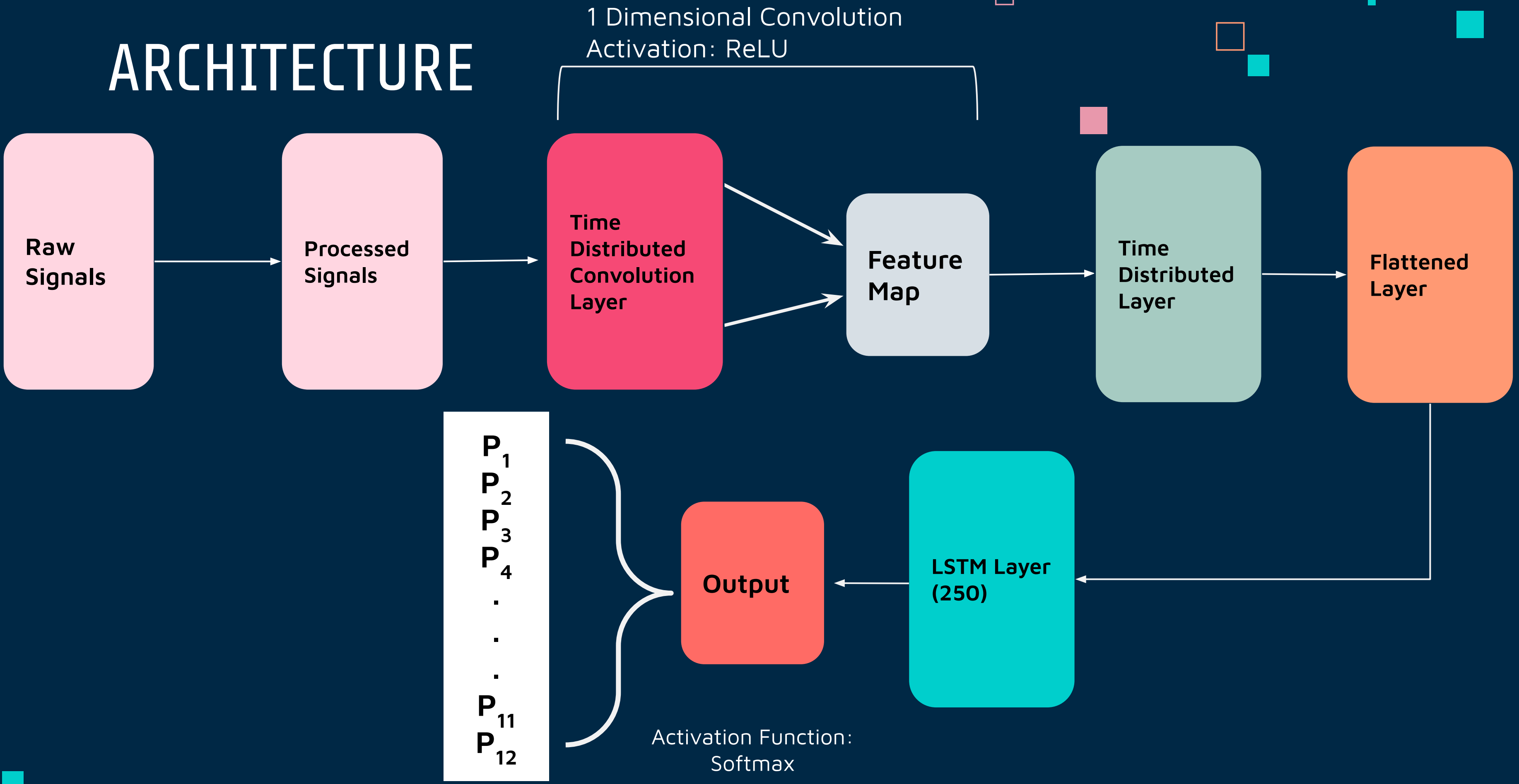
1. Temporal nature of signals
2. The spatial nature and patterns formed by the waveform of signals.

Convolutional layers can capture the spatial information of the data, and LSTM layers helps maintain the temporal integrity of the data.

Utilize both features to capture the most information within the raw signals.



ARCHITECTURE



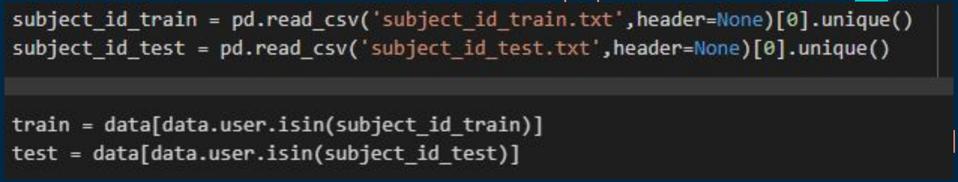
DESIGN OF MODEL

Data Preprocessing:

To facilitate a form of comparison with previous models:
Split the original raw data into train and test groups as how the experimenter initially assigned experiment users.

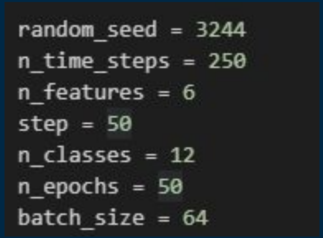
Preprocessing of Inputs to NN:

For each group, the data is scaled, normalized, and framed into windows as input for the CNN-LSTM model. Each window is assigned the modal label of the frames.

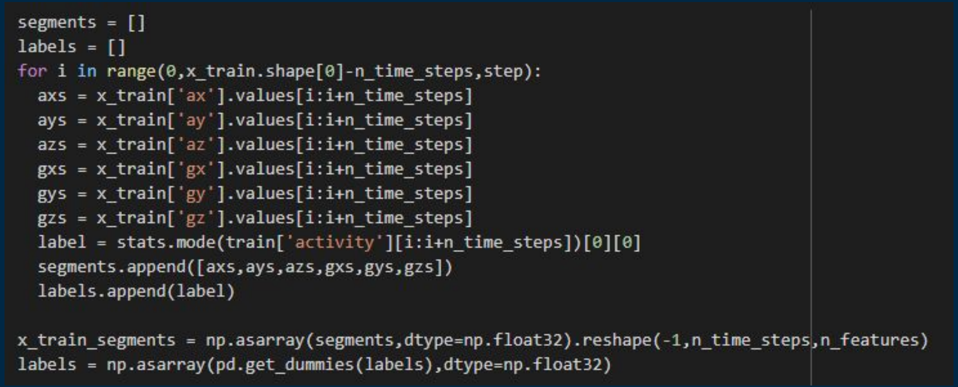


```
subject_id_train = pd.read_csv('subject_id_train.txt',header=None)[0].unique()
subject_id_test = pd.read_csv('subject_id_test.txt',header=None)[0].unique()
```

```
train = data[data.user.isin(subject_id_train)]
test = data[data.user.isin(subject_id_test)]
```



```
random_seed = 3244
n_time_steps = 250
n_features = 6
step = 50
n_classes = 12
n_epochs = 50
batch_size = 64
```



```
segments = []
labels = []
for i in range(0,x_train.shape[0]-n_time_steps,step):
  axs = x_train['ax'].values[i:i+n_time_steps]
  ays = x_train['ay'].values[i:i+n_time_steps]
  azs = x_train['az'].values[i:i+n_time_steps]
  gxs = x_train['gx'].values[i:i+n_time_steps]
  gys = x_train['gy'].values[i:i+n_time_steps]
  gzs = x_train['gz'].values[i:i+n_time_steps]
  label = stats.mode(train['activity'][i:i+n_time_steps])[0][0]
  segments.append([axs,ays,azs,gxs,gys,gzs])
  labels.append(label)

x_train_segments = np.asarray(segments,dtype=np.float32).reshape(-1,n_time_steps,n_features)
labels = np.asarray(pd.get_dummies(labels),dtype=np.float32)
```

DESIGN OF MODEL

Layer Implementation

Convolutional Layer:
1-Dimensional convolutional layer with 250 filters and kernel size of 3.

Time Distribution Layer:
Wraps around **every layer prior** to an LSTM cell to preserve the temporal aspect within the input frames.

Flatten Layer:
Flattens the feature maps as a single input into the LSTM cells.

LSTM Layer:
Contains 250 units which corresponds to the number of data points that in the input signals. Allows the layer to process data points from the entire time step input into the network.

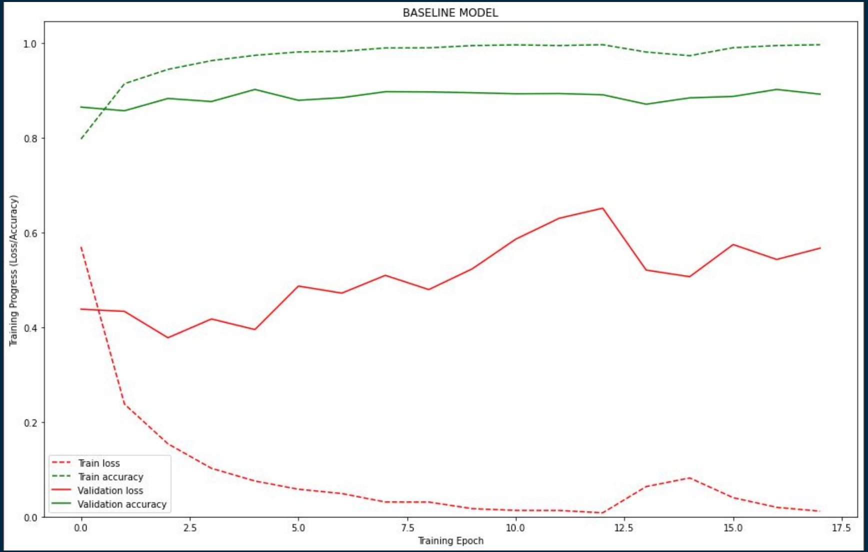
Output Layer:
A Dense layer with 12 neurons with SoftMax activation function to give us the probability of each class.

Parameters	Samples
Training Set	11541
Validation Set	20% of Training
Testing Set	4738
Dropout	0.5 (if any)
Regularization	Early Stoppage, patience = 5

EVALUATION OF MODEL

Model Description

Signs of overfitting:
5th epoch: validation accuracy stops improving
8th epoch: training accuracy reaches 99%
With more epochs, validation loss increases steadily
Low training error and high validation error



Subpar accuracy achieved by this model, indicative of the model not learning the data well enough.

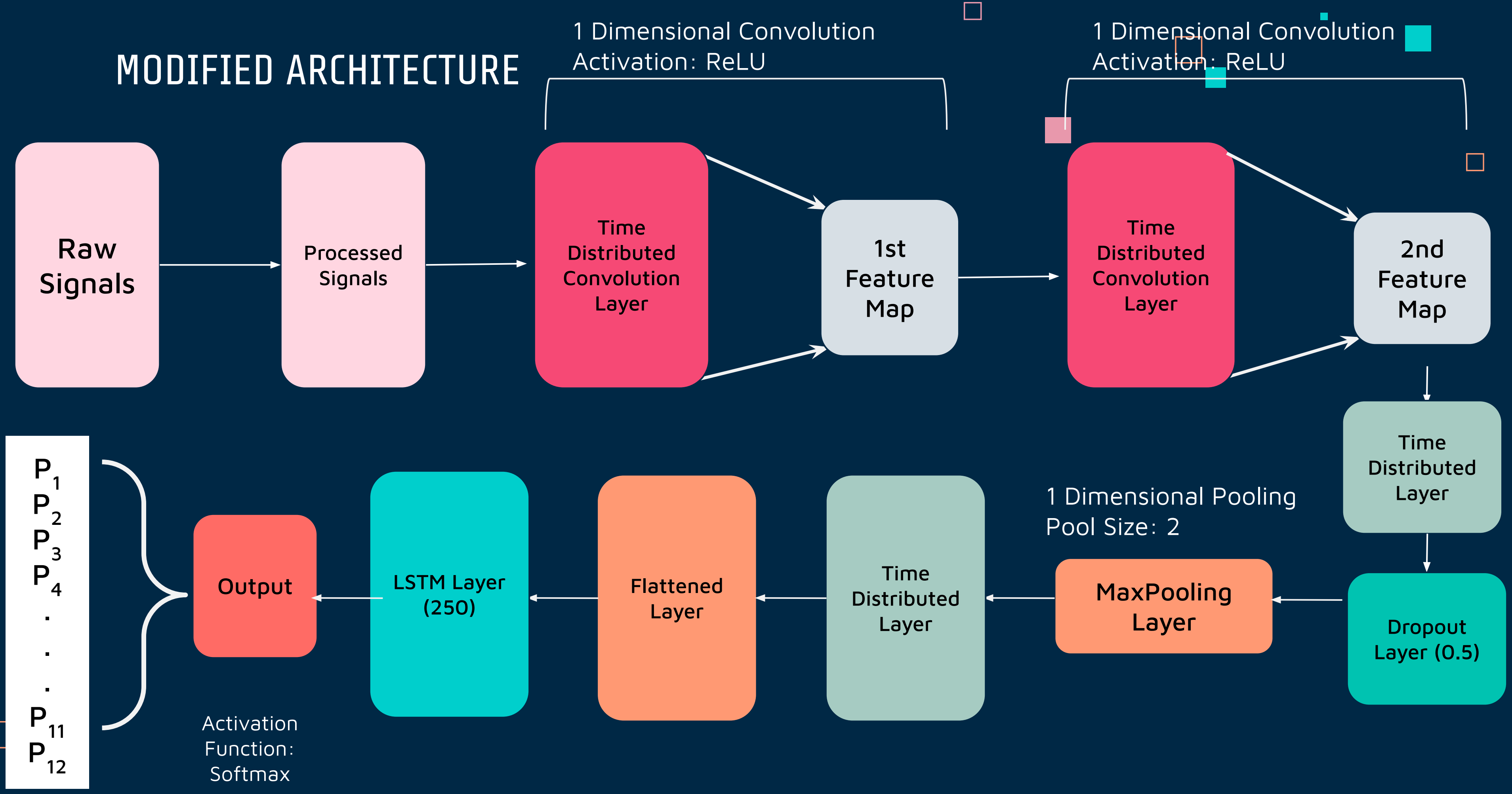
Further modifications to the architecture can be applied to improve learning of higher level features of the training data.

Evaluation	Value
Accuracy	0.889
Precision	0.898
Recall	0.887
AUC	0.980
Cross-Entropy Loss	0.500

ATTEMPTED ADJUSTMENTS

Model	Description	Remarks
2CNN-LSTM	2 Convolutional Layers were used instead of 1 only	Overfitting became more drastic. Accuracy improved marginally.
2CNN-LSTM with Pooling	MaxPooling was applied to capture the larger picture of the waveforms, and to reduce dimensionality	Overfitting became very drastic. Accuracy improved greatly.
Dense 2CNN-LSTM	A Dense Layer was connected between the LSTM and output layer.	Overfitting managed to reduce, but is still apparent. Accuracy improved marginally
Dense 2CNN-LSTM with Regularisation	Dropout layers were placed between layers to reduce overfitting. Regularisation was tuned.	Reduction in overfitting was marginal. Accuracy had marginal improvements
2CNN-LSTM with Pooling and Model Tuning	Combinations of Dropout layers were experimented with.	Best reduction in overfitting with highest increase in accuracy

MODIFIED ARCHITECTURE



ADDITIONS TO MODEL

NEW Layers

Convolutional Layers:
An additional convolutional layers.
To compensate for this addition, size of the filter in each convolutional layer is scaled down to 125.

MaxPooling Layer:
Added a pooling layer to reduce dimensionality.
To better capture the general pattern of the signals throughout each input frame.
pool_size set to 2, to obtain the max of each convolved Acceleration and Gravitational Acceleration signals.

Dropout Layer:
Applied a TimeDistributed wrapped Dropout Layer on the 2nd feature map, before applying pooling.
To introduce another layer of regularization, preventing overfitting.

Parameters	Initial	Final
Training Set	11541	11541
Validation Set	20% of Training	20% of Training
Testing Set	4738	4738
Dropout	None	0.5
Regularization	Early Stoppage, patience = 5	Early Stoppage, patience = 5

RATIONALE FOR ADDITIONS

Convolutional Layer

- The model was not learning the data well enough. In particular, as the signals contain many mini-fluctuations (jagged edges), using 1 Convolutional layer might cause our NN to learn low-level features with many perturbations within it.
- Using 2 Convolutional layers allows for hierarchical decomposition, which allows our neural network to learn higher level features within the waveforms. [1]

MaxPooling Layer

- With a deep model, there are much more parameters to train, which could contribute significantly to overfitting. Using a pooling layer reduces the number of dimensions at that layer.
- Pooling also adds **translational invariance** to our feature maps. As frame shifts, there is a translation in the entire waveform with time. Hence, we want the NN to be able to recognize patterns in the waveform, regardless of their position. [2]
- MaxPooling is able to capture the **overall presence of these shapes**, even with the **peaks and crests** within the waveforms in the frames. [3]

[1]: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks
[2]:https://stats.stackexchange.com/questions/208936/what-is-translation-invariance-in-computer-vision-and-convolutional-neural-network
[3]:https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks

RATIONALE FOR ADDITIONS

Dropout Layers and Regularization Values

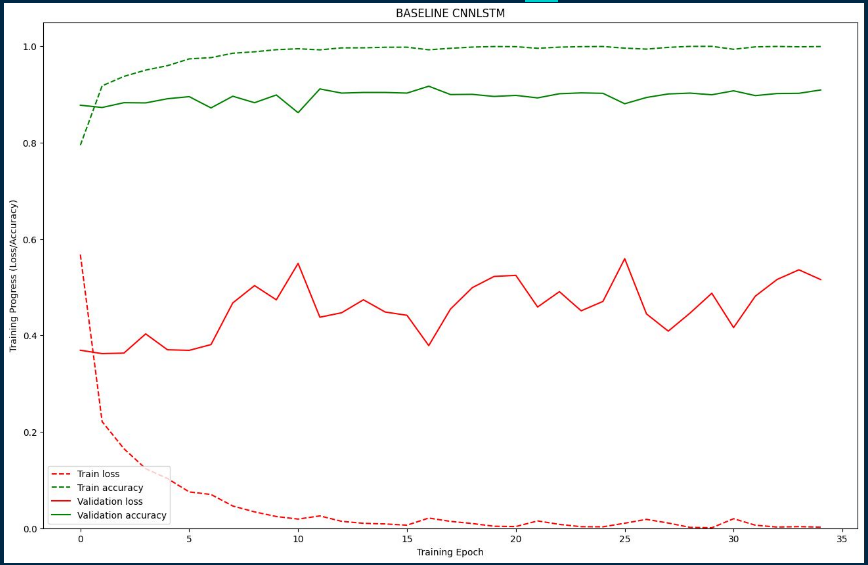
- Performs regularization due to the large number of parameters to be trained (250 units in LSTM cell, 2 Convolutional Layers with 125 filters each).
- Experimented with varying layers of dropouts with different rates at each layer.
    - Varied implementing 0 to 3 Dropout layers. The more dropout layers there were, the lower the accuracy of the model. Likely due to these layers deactivating too many neurons.
    - Varied implementing dropout rates of 0.0 to 0.5. Accuracy was stagnant, but the cross-entropy loss fluctuated greatly. With higher dropout rates in all 3 dropout layers, the loss was lower. However, accuracy decreased, likely due to the layers deactivating too many neurons
    - **Combined Effect**: The improvements in accuracy and decrease in cross-entropy loss was **marginal** in the case of using 2 or 3 Dropout layers.
- To avoid **unnecessary deactivations of neurons**, or **unnecessary additions of Dense layers** only 1 Dropout layer was implemented. Placed between the 2nd Convolutional layer and the MaxPooling layer.

EVALUATION OF MODEL

Model Description

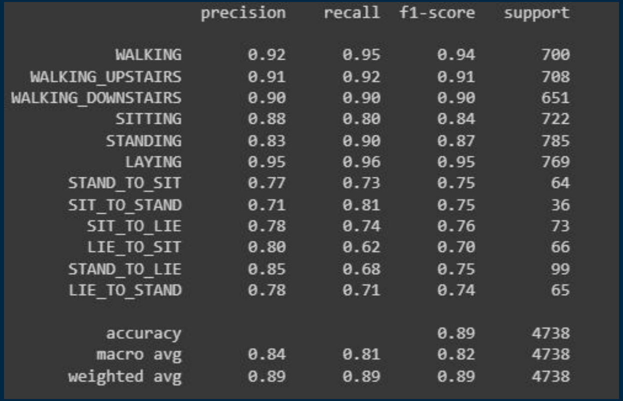
With the implementation of an additional Convolutional layer and MaxPooling, the validation accuracy is generally higher, and more consistent.

With the addition of dropout layers and the pooling layer, the validation loss is reduced to ranges between 0.4 to 0.6, which is a considerable reduction compared to the base model.



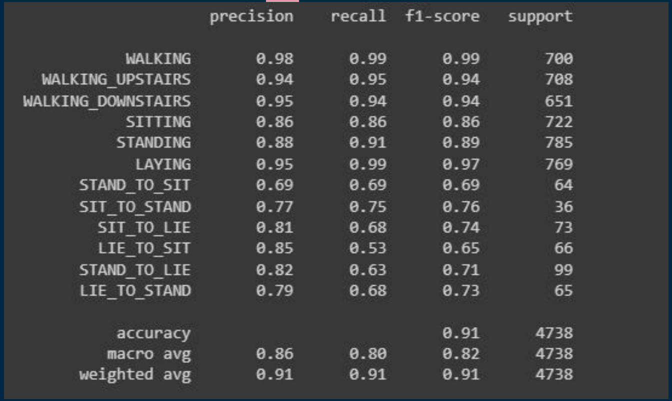
Evaluation	Value
Accuracy	0.914
Precision	0.920
Recall	0.912
AUC	0.982
Cross-Entropy Loss	0.424

COMPARISON OF MODELS



```
                    precision    recall  f1-score   support

           WALKING       0.92      0.95      0.94       700
  WALKING_UPSTAIRS       0.91      0.92      0.91       708
WALKING_DOWNSTAIRS       0.90      0.90      0.90       651
           SITTING       0.88      0.80      0.84       722
          STANDING       0.83      0.90      0.87       785
            LAYING       0.95      0.96      0.95       769
      STAND_TO_SIT       0.77      0.73      0.75        64
      SIT_TO_STAND       0.71      0.81      0.75        36
        SIT_TO_LIE       0.78      0.74      0.76        73
        LIE_TO_SIT       0.80      0.62      0.70        66
      STAND_TO_LIE       0.85      0.68      0.75        99
      LIE_TO_STAND       0.78      0.71      0.74        65

          accuracy                           0.89      4738
         macro avg       0.84      0.81      0.82      4738
      weighted avg       0.89      0.89      0.89      4738
```

Fig 1. Base Model



```
                    precision    recall  f1-score   support

           WALKING       0.98      0.99      0.99       700
  WALKING_UPSTAIRS       0.94      0.95      0.94       708
WALKING_DOWNSTAIRS       0.95      0.94      0.94       651
           SITTING       0.86      0.86      0.86       722
          STANDING       0.88      0.91      0.89       785
            LAYING       0.95      0.99      0.97       769
      STAND_TO_SIT       0.69      0.69      0.69        64
      SIT_TO_STAND       0.77      0.75      0.76        36
        SIT_TO_LIE       0.81      0.68      0.74        73
        LIE_TO_SIT       0.85      0.53      0.65        66
      STAND_TO_LIE       0.82      0.63      0.71        99
      LIE_TO_STAND       0.79      0.68      0.73        65

          accuracy                           0.91      4738
         macro avg       0.86      0.80      0.82      4738
      weighted avg       0.91      0.91      0.91      4738
```

Fig 2. Final Model

- Comparing the precision, recall and F1-score of each class, the final model has shown an improvement in detecting WALKING classes and STATIC classes (SITTING, STANDING, LAYING). (General **increase** in precision, recall and F1-score)
- However, for TRANSITIVE class, the precision, recall and F1-score has little to no increase from our initial to final model. (**No clear improvement/deprovement** in metrics)
- Overall, the micro-average accuracy of the final model has increased **from 89% to 91%**

COMPARISON OF MODELS

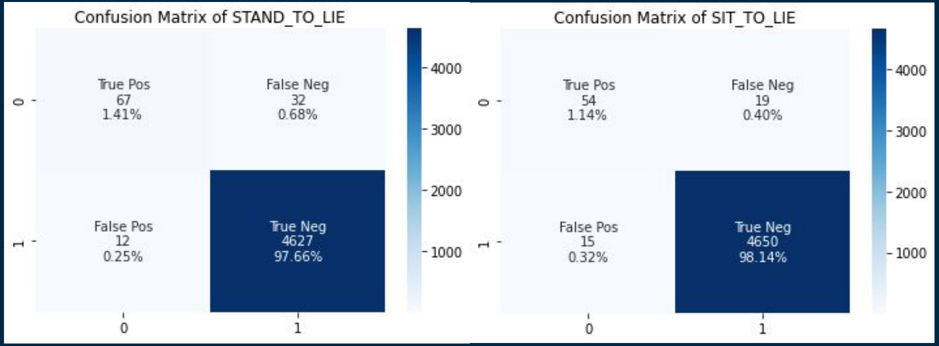


Fig 1. Base Model

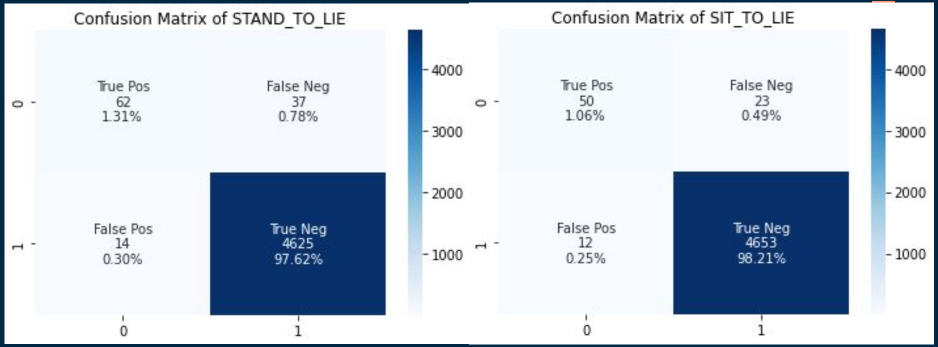


Fig 2. Final Model

- There is a overall deprovement for the STAND_TO_LIE class. The final model predicted less True Positives and more False Positives and False Negatives. Therefore, the final model gives a lower precision **(85% to 82%)** and lower recall **(68% to 63%)**
- There is an overall improvement for the SIT_TO_LIE class. Though it predicted more False Negatives and less True Positives, it was able to predict lesser False Positives and more True Negatives. As such, there is a higher precision **(78% to 81%)**, and lower recall **(74% to 68%)**
- **There is higher False Negatives for both classes from the final model. This incurs a larger cost.**

DID THE FINAL MODEL WORK BETTER?

Benefits:

- Performance increased against the baseline model
  - Adding Convolutional layers and Pooling layers allowed the model to better learn the patterns of the waveforms.
  - Combination of Convolutional layer and LSTM cells proved to be more effective.

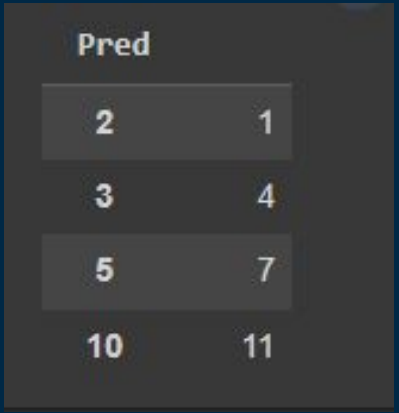
- Tuning of the network reduced overfitting within the model
  - General reduction in Cross-Entropy loss
  - Gradual increase in validation accuracy
  - Validation accuracy does not stray too far from training accuracy

Disadvantages:

- Overfitting is still apparent within our model
  - Validation loss still shows a slight general increasing trend. There is a still a difference in validation loss and training loss, although the difference is much smaller.
  - This is likely due to the implementation of more layers within our network, and hence, more trainable parameters.

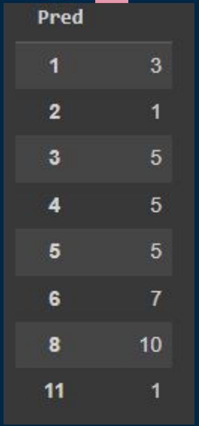
- **Higher False Negatives** in detecting dangerous movements
  - Likely due to combined effect of overfitting + lack of data points with these classes.

BLACKBOX OBSERVATION OF OUTPUT



Pred	
2	1
3	4
5	7
10	11

Fig 1: Distribution of Incorrect Classes for label SIT_TO_LIE



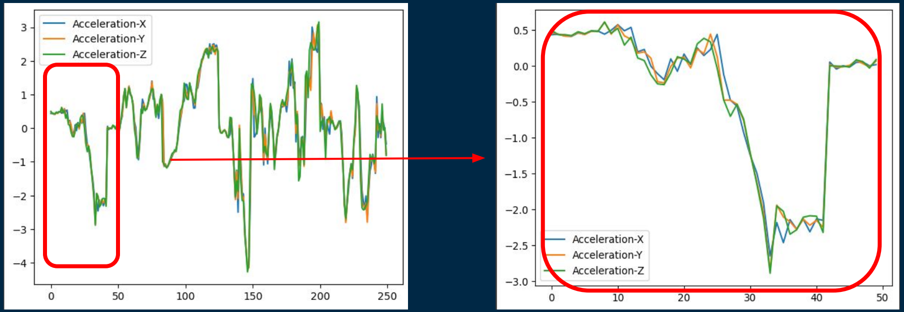
Pred	
1	3
2	1
3	5
4	5
5	5
6	7
8	10
11	1

Fig 2: Distribution of Incorrect Classes for label STAND_TO_LIE

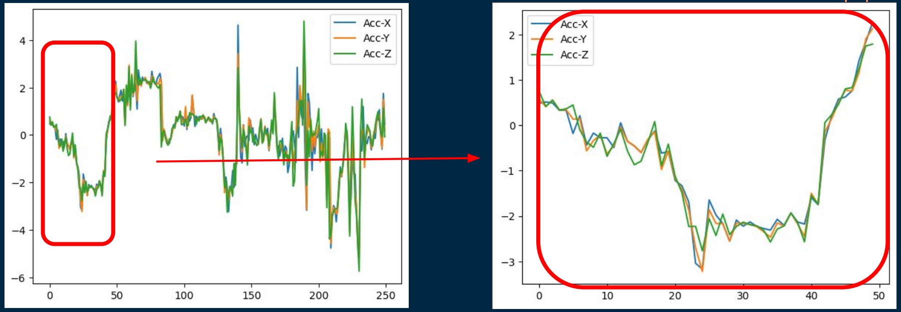
Predictions with the label SIT_TO_LIE, out of all incorrect predictions, the model often predicted SIT_TO_LIE as STAND_TO_LIE (class 10).

Similarly, for predictions with label STAND_TO_LIE, the model often predicted it as SIT_TO_LIE (class 8)

BLACKBOX OBSERVATION OF OUTPUT



Frame 1101
Predicted: STAND_TO_LIE
Actual: SIT_TO_LIE



Frame 1533
Predicted: SIT_TO_LIE
Actual: STAND_TO_LIE

- There are some portions of the waveforms that are highly similar. Consider the first component of the above 2 frames

- We see that there is a general trend of a crest. Since our model is unable to tell apart the nuances in the component waveforms, the model might be confused between the 2 classes.

Future Improvements

07

FUTURE IMPROVEMENTS

- There is a need to have a robust method to upsample the waveforms that belong to our minority classes (i.e TRANSITIVE actions)
  - Given the setting of the experiment, there is an inherent imbalance of TRANSITIVE movements (every 2 STATIC movements, only 1 TRANSITIVE movement generated)

- Consider the idea of performing Convolutions within LSTM cells
  - Currently we perform Convolutions through a Convolution layer
  - This contributes to a deep NN, which brings about the problem of overfitting
  - If we can perform convolutions while reducing the number of trainable parameters, this might reduce the issue of overfitting
  - Consider the usage of ConvLSTM layers [1][2]

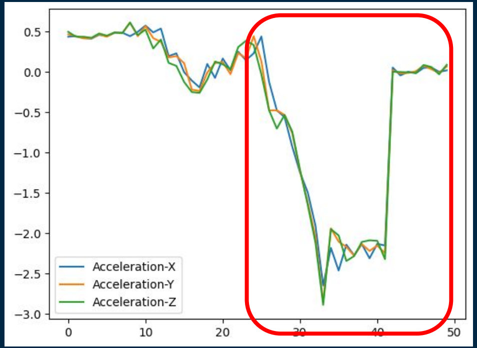
- There is a need to have a more robust definition of dangerous movements.
  - In actuality, dangerous movements are defined by a sequence of class labels (i.e STAND -> STAND_TO_LIE -> LYING). They are also qualified by their duration for each class label. (i.e STAND -> STAND_TO_LIE -> LYING -> LYING -> ... -> LYING would indicate someone has collapsed).
  - We do not have labels that can identify these dangerous movements to the model. The context of which the data was gathered during the experiment is different (subjects were made to perform specific and fixed actions), so it would not be accurate to define dangerous movements purely based off the given data

[1]: https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7
[2]: https://paperswithcode.com/method/convlstm

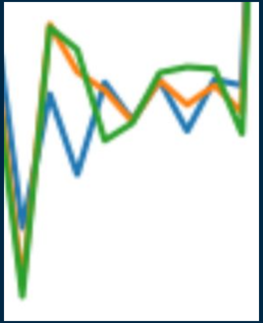
FUTURE IMPROVEMENTS

- Consider adding in noise into our waveforms. These minor perturbations to our waveform ensures that our NN does not purely memorise the position of our waveforms with respect to the timeframe, but rather, learns the general pattern of the waveform
    - Gaussian Distributed Noise can be added into the raw signals [1]
    - Solves Vanishing Gradient Problem

- Perform further tuning on our LSTM cell
    - Want the cell to remember the general pattern of the waveform instead of minor variations.
    - We want to balance the ability to detect nuances in the waveform (mentioned in blackbox output), with the ability to detect general trends equally.
    - **Tuning** has to be done on the **'forget' bias** within the 'forget' gate of the LSTM cell [2]



<- What we want the LSTM cell to memorise



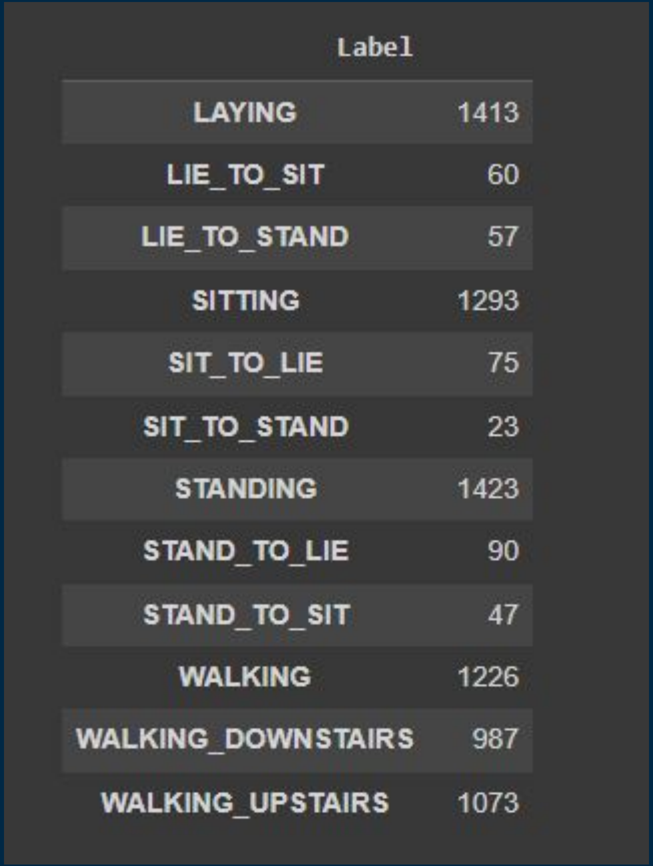
<- What it might actually be memorising

[1]: https://towardsdatascience.com/perturbation-theory-in-deep-neural-network-dnn-training-adb4c20cab1b
[2]: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm

FUTURE IMPROVEMENTS - Data

- Look for/generate data using other devices
    - Noise is especially present in earlier devices like the S2 Galaxy [1]
    - Our model could overfit to the noise/specific device properties

- Reduce the data imbalance by recording more samples with labels as postural transitions
    - ~15 times as many samples for static positions than postural transitions
    - Recording samples in different locations can also help to generalize and improve learning [2]



	Label
LAYING	1413
LIE_TO_SIT	60
LIE_TO_STAND	57
SITTING	1293
SIT_TO_LIE	75
SIT_TO_STAND	23
STANDING	1423
STAND_TO_LIE	90
STAND_TO_SIT	47
WALKING	1226
WALKING_DOWNSTAIRS	987
WALKING_UPSTAIRS	1073

[1]: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4570352/
[2]: https://ieeexplore.ieee.org/document/6063364

CONCLUSION

- Started working with linear models before moving onto more complex models like NNs (CNNs and RNNs)
- We learn that linear models can be good at modelling this HAR problem. But its performance is very dependent on the dataset it was trained on, feature engineering performed, and how the data was collected. It is unable to be deployed to be used in a generalized setting.
- NNs perform well in modelling this complex HAR problem. An ensemble of CNN-LSTM performed the best out of all NNs that were trained
- However, with regards to detecting dangerous activities, our model is not robust enough to tackle this problem. More tuning has to be done before deployment of our model.
- More consideration of the validity of this dataset towards solving our problem statement has to be given. This dataset is not suitable on its own to target the issue of detecting dangerous movements.
- Nevertheless, our model is still competent in differentiating whether a person is sitting, standing, lying, or walking (even to the point of detecting walking direction). It can find use in other settings.

THANK YOU