Deep Learning – Assignment 2 Question 1-2 Report

Submitted by: Itay Bouganim and Ido Rom

PAMAP2 Physical Activity Monitoring dataset

Problem Statement

The PAMAP2 Physical Activity Monitoring dataset contains data of 18 different physical activities (such as walking, cycling, playing soccer, etc.), performed by 9 subjects wearing 3 inertial measurement units and a heart rate monitor. The dataset can be used for activity recognition and intensity estimation, while developing and applying algorithms of data processing, segmentation, feature extraction and classification.

Task Description

The PAMAP2 Physical Activity Monitoring dataset contains data of 18 different physical activities, performed by 9 subjects wearing 3 inertial measurement units and a heart rate monitor. Our goal is to classify timeseries window measurements from individual activities to type of activity the measurements were taken from.

1. **Exploratory Data Analysis**

The following sections will refer to tasks 1.a. and 1.b.

Data collection protocol

Each of the subjects had to follow a protocol, containing 12 different activities. The folder **'Protocol'**. contains these recordings by subject.
Furthermore, some of the subjects also performed a few optional activities. The folder **'Optional'**. contains these recordings by subject.

Data files

Raw sensory data can be found in space-separated text-files (.dat), 1 data file per subject per session (protocol or optional). Missing values are indicated with NaN.
One line in the data files correspond to one timestamped and labeled instance of sensory data.
The data files contain 54 columns: each line consists of a timestamp, an activity label (the ground truth) and 52 attributes of raw sensory data.

Dataset activities (Protocol and Optional) analysis

```
{0: 'transient', 1: 'lying', 2: 'sitting', 3: 'standing', 4: 'walking',
 5: 'running', 6: 'cycling', 7: 'nordic_walking', 9: 'watch_tv',
10: 'computer_work', 11: 'car_driving', 12: 'asc_stairs',
13: 'desc_stairs', 16: 'vaccum', 17: 'ironing', 18: 'folding_laundry',
 19: 'house_cleaning', 20: 'soccer', 24: 'rope_jump'}
```

Protocol data unique activities:

```
 {0: 'transient', 1: 'lying', 2: 'sitting', 3: 'standing', 17: 'ironing', 1
6: 'vaccum', 12: 'asc_stairs', 13: 'desc_stairs', 4: 'walking', 7: 'nordic_
walking', 6: 'cycling', 5: 'running', 24: 'rope_jump'}
```

Optional data unique activities:

```
{0: 'transient', 11: 'car_driving', 9: 'watch_tv', 19: 'house_cleaning', 18
: 'folding_laundry', 10: 'computer_work', 20: 'soccer'}
```

```
Protocol and Optional intersected activites:
{0: 'transient'}
```

Protocol and Optional activity id union:

```
[ 0  1  2  3  4  5  6  7  9 10 11 12 13 16 17 18 19 20 24]
```

number of activities in union: 19

Total activity count (including transient): 19

Timestamp ranges:

```
Min timestamp in protocol data: 5.64
Max timestamp in protocol data: 4475.63
Min timestamp in optional data: 5.66
Max timestamp in optional data: 3203.54
```

II. Data format

II.1. Synchronized and labeled raw data from all the sensors (3 IMUs and the HR-monitor) is merged into 1 data file per subject per session (protocol or optional), available as text-files (.dat).

Each of the data-files contains 54 columns per row, the columns contain the following data:

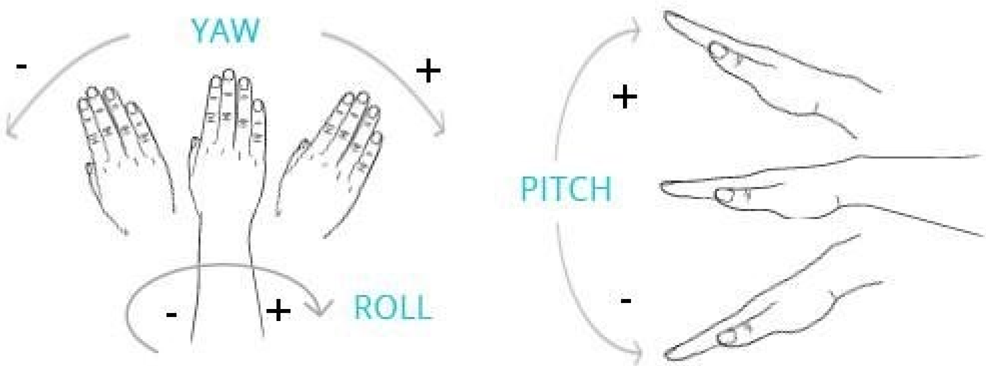
- 1 timestamp (s)
- 2 activityID (see II.2. for the mapping to the activities)
- 3 heart rate (bpm)
- 4-20 IMU hand
- 21-37 IMU chest
- 38-54 IMU ankle

The IMU sensory data contains the following columns:

- 1 temperature (°C)
- 2-4 3D-acceleration data (ms^{-2}), scale: ±16g, resolution: 13-bit
- 5-7 3D-acceleration data (ms^{-2}), scale: ±6g, resolution: 13-bit*
- 8-10 3D-gyroscope data (rad/s)
- 11-13 3D-magnetometer data (μT)
- 14-17 orientation (invalid in this data collection)

Missing sensory data due to wireless data dropping: missing values are indicated with NaN.
Since data is given every 0.01s (due to the fact, that the IMUs have a sampling frequency of 100Hz), and the sampling frequency of the HR-monitor was only approximately 9Hz, the missing HR-values are also indicated with NaN in the data-files.

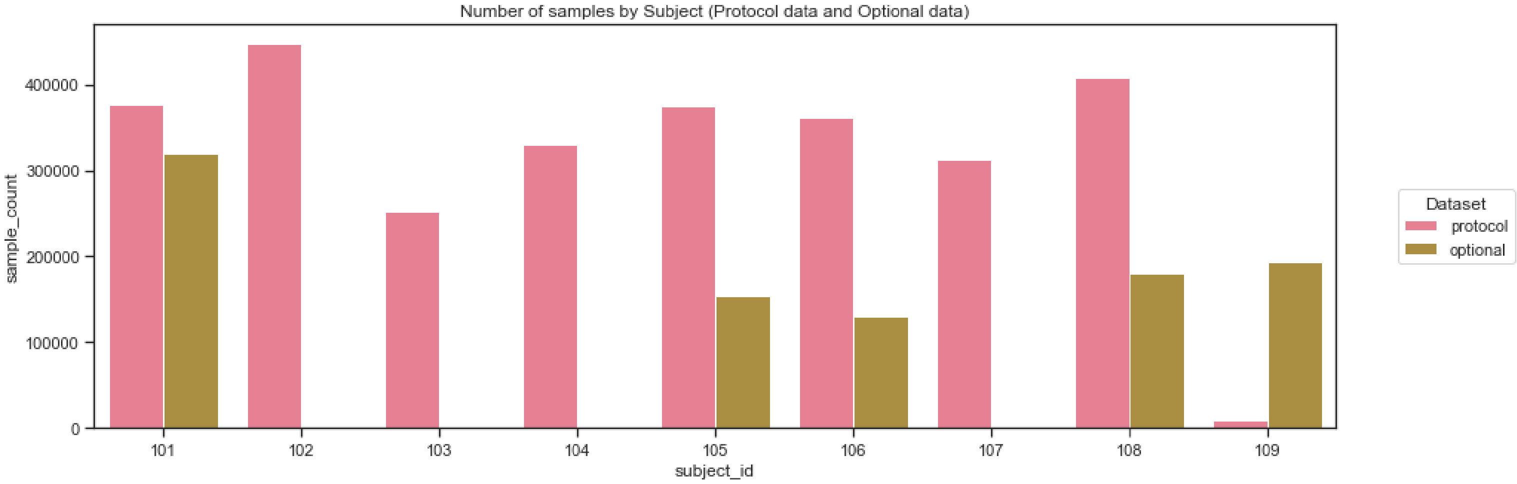
3D sensory data can be seen as (x, y, z) that translates into (pitch, roll and yaw)



Subject Analysis:

Our dataset contains samples of activities from 9 different subjects.

`Subject IDS: [101, 102, 103, 104, 105, 106, 107, 108, 109]`



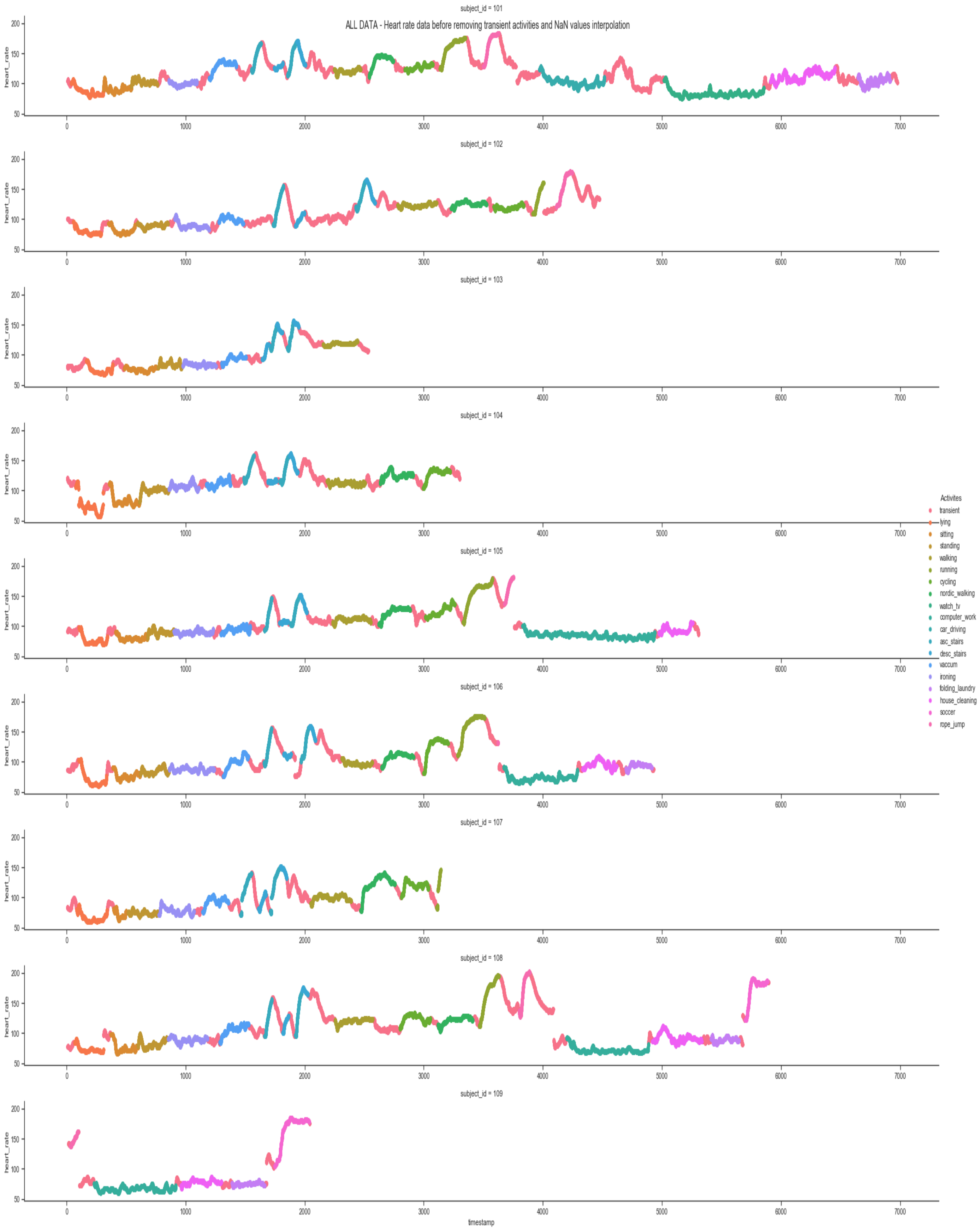
Subject Information

Subject ID	Sex	Age (years)	Height (cm)	Weight (kg)	Resting HR (bpm)	Max HR (bpm)	Dominant hand
101	Male	27	182	83	75	193	right
102	Female	25	169	78	74	195	right
103	Male	31	187	92	68	189	right
104	Male	24	194	95	58	196	right
105	Male	26	180	73	70	194	right
106	Male	26	183	69	60	194	right
107	Male	23	173	86	60	197	right
108	Male	32	179	87	66	188	left
109	Male	31	168	65	54	189	right

Activities performed by subjects (in seconds)

	subject101	subject102	subject103	subject104	subject105	subject106	subject107	subject108	subject109	Sum	Nr. of subjects
1 – lying	271.86	234.29	220.43	230.46	236.98	233.39	256.1	241.64	0	1925.15	8
2 – sitting	234.79	223.44	287.6	254.91	268.63	230.4	122.81	229.22	0	1851.8	8
3 – standing	217.16	255.75	205.32	247.05	221.31	243.55	257.5	251.59	0	1899.23	8
4 – walking	222.52	325.32	290.35	319.31	320.32	257.2	337.19	315.32	0	2387.53	8
5 – running	212.64	92.37	0	0	246.45	228.24	36.91	165.31	0	981.92	6
6 – cycling	235.74	251.07	0	226.98	245.76	204.85	226.79	254.74	0	1645.93	7
7 – Nordic walking	202.64	297.38	0	275.32	262.7	266.85	287.24	288.87	0	1881	7
9 – watching TV	836.45	0	0	0	0	0	0	0	0	836.45	1
10 – computer work	0	0	0	0	1108.82	617.76	0	687.24	685.49	3099.31	4
11 – car driving	545.18	0	0	0	0	0	0	0	0	545.18	1
12 – ascending stairs	158.88	173.4	103.87	166.92	142.79	132.89	176.44	116.81	0	1172	8
13 – descending stairs	148.97	152.11	152.72	142.83	127.25	112.7	116.16	96.53	0	1049.27	8
16 – vacuum cleaning	229.4	206.82	203.24	200.36	244.44	210.77	215.51	242.91	0	1753.45	8
17 – ironing	235.72	288.79	279.74	249.94	330.33	377.43	294.98	329.89	0	2386.82	8
18 – folding laundry	271.13	0	0	0	0	217.85	0	236.49	273.27	998.74	4
19 – house cleaning	540.88	0	0	0	284.87	287.13	0	416.9	342.05	1871.83	5
20 – playing soccer	0	0	0	0	0	0	0	181.24	287.88	469.12	2
24 – rope jumping	129.11	132.61	0	0	77.32	2.55	0	88.05	63.9	493.54	6
Labeled total	4693.07	2633.35	1743.27	2314.08	4117.97	3623.56	2327.63	4142.75	1652.59	27248.27	
Total	6957.67	4469.99	2528.32	3295.75	5295.54	4917.78	3135.98	5884.41	2019.47	38504.91	

Heart-rate monitor overview across activities (Optional and Protocol)



Few notices on the Protocol and Optional data

- We can see that the only common activity to the protocol data and optional data of the PAMAP2 dataset is activity 0 which is an transient activity and therfore will be removed.
- We want our data to contain labeled data fro both the protocol activities and optional activities.
- To do so we will concatenate our protocol and optional data. **A problem that may occur from such concatenation is distortion of the timestamp data**
  To overcome this problem, and since we wish to classify each activity individually and not the relation between activities, we will set all the optional data timestamp to be after the protocol timestamps.

Observe missing sensory data due to sensor frequency difference

IMU sensors frequency is 100Hz meaning 100 samples per second.
HR sensor frequency is 9Hz meaning 9 samples per second.

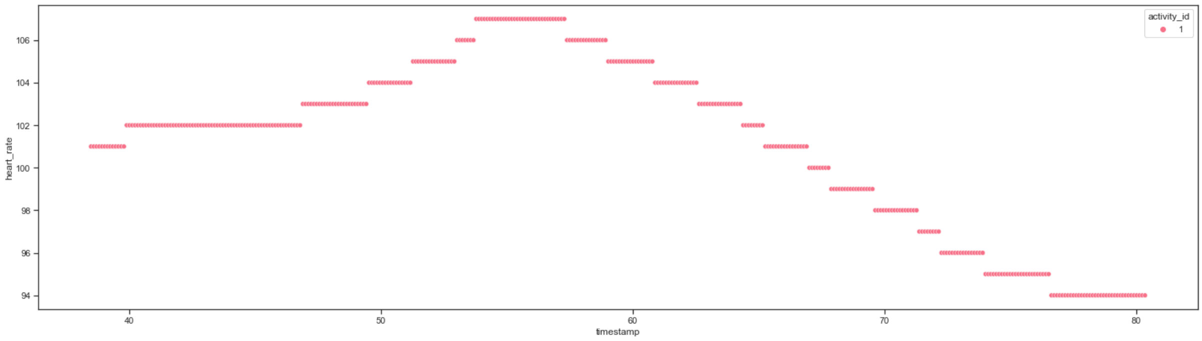
We expect that for every 9 HR samples we will have 100 IMU samples so the ratio will be approx. 100/9 = ~11.11

- The data contains missing values (NaN) due to wireless data dropping and due to the HR and IMU sensor frequency diff
- We can see that the data includes transient activities between other activities (index 0) that needs to be removed
- We can also note that all subject performed the activities in the same order as mentioned in the protocol.
- Not all subject performed all activites (i.e. subject 109 performed only one activity)
- Not all 18 activites has representation in the data

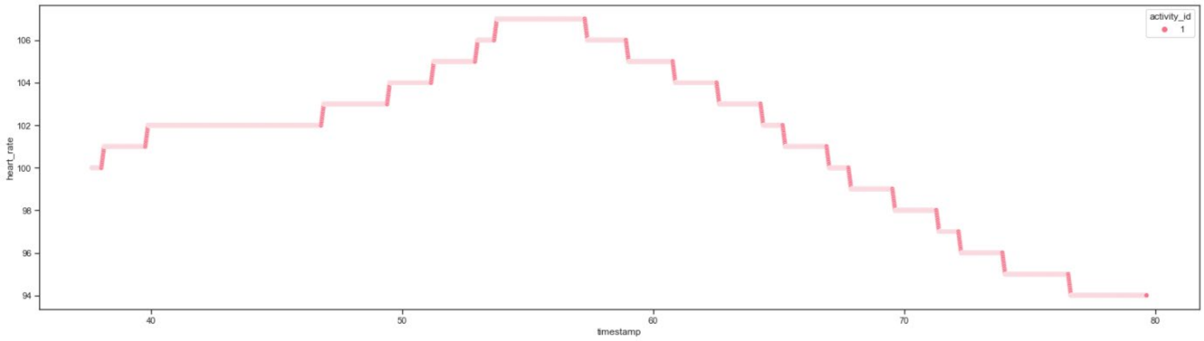
Interpolation of missing data

We can see that the data contains transient activities (activity_id = 0) and that some measurments contain NaN values.
As mentioned earlier, transient activities should be discarded.

We will solve the NaN values by using interpolation to estimate the missing value according to the previous and next closest values.



Before Interpolation of NaN



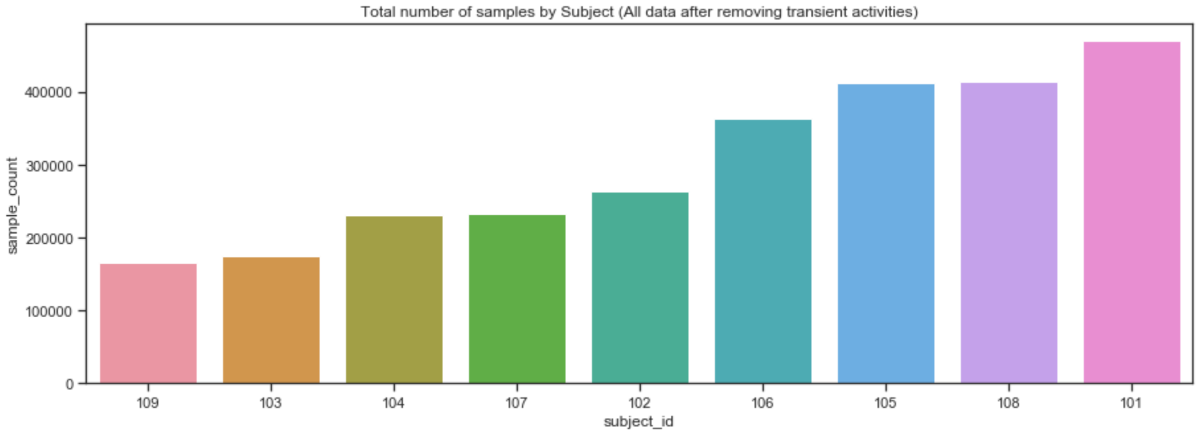
After Interpolation of NaN

After interpolation process and removing transient activities from out data we can conclude the following sample count:

```
Total number of samples (including transient activities): 211777775
Total number of samples (discarding transient activities): 149872415
Number of columns in the tabular data: 55
Unique activites in the data: 18
Number of subjects in the data: 9
```

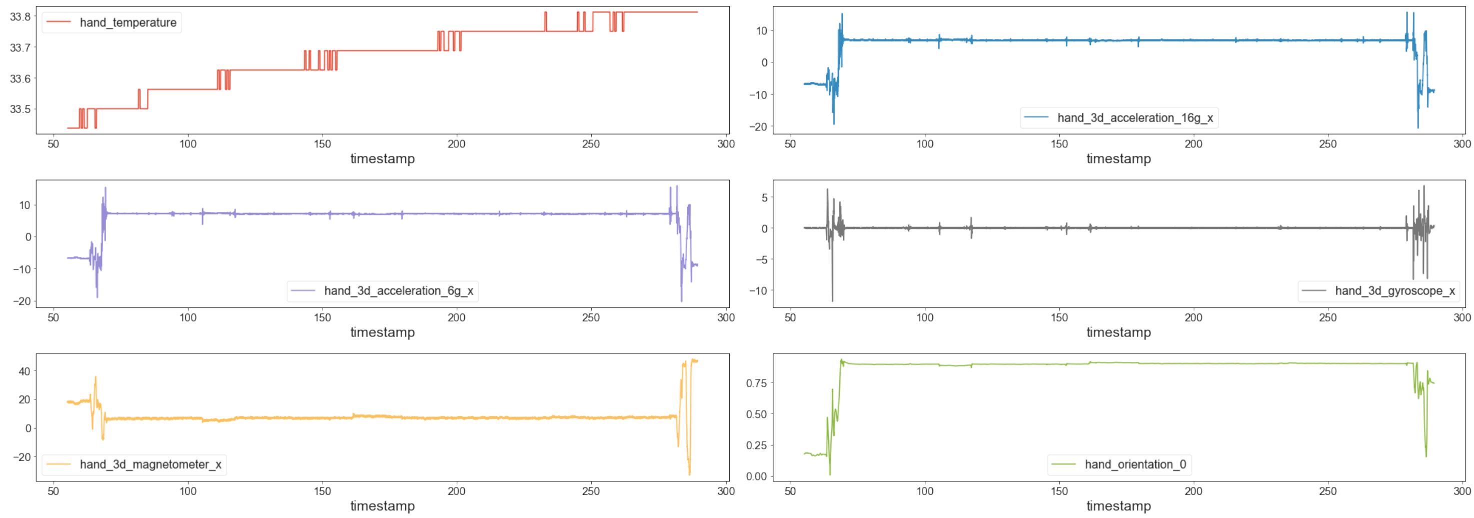
The task at hand

After reviewing out data we can define out task.
The task at hand is a hierarchical classification task.
Given a measurement of the HR-monitor and the IMU sensors we want to predict what is the activity performed by the subject at that particular time frame.
We will classify segments of the time series to one of the valid activities and try to determine the activity the measurements were taken from.

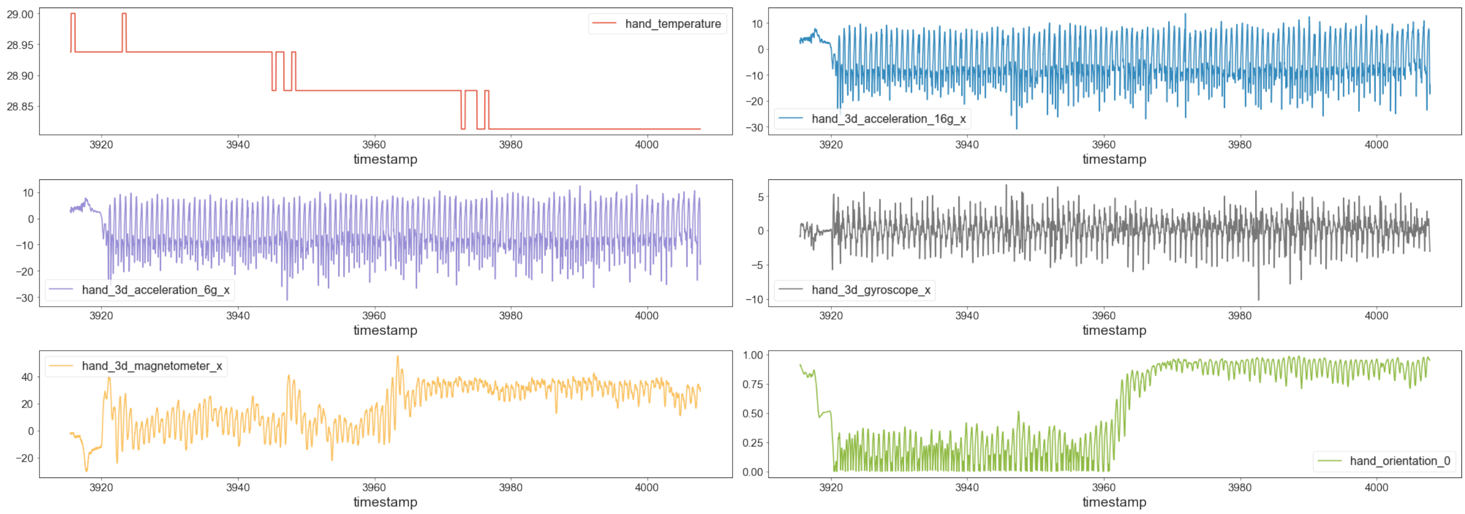


We want to gain further insights about how the different activities can be classified and differentiated according to the measurements taken in a time frame.

We will use subject 102 hand IMU sensors to take some samples from different activities (We will present here 2 different activities. For the full charts and analysis please refer to the notebook attached to this task).

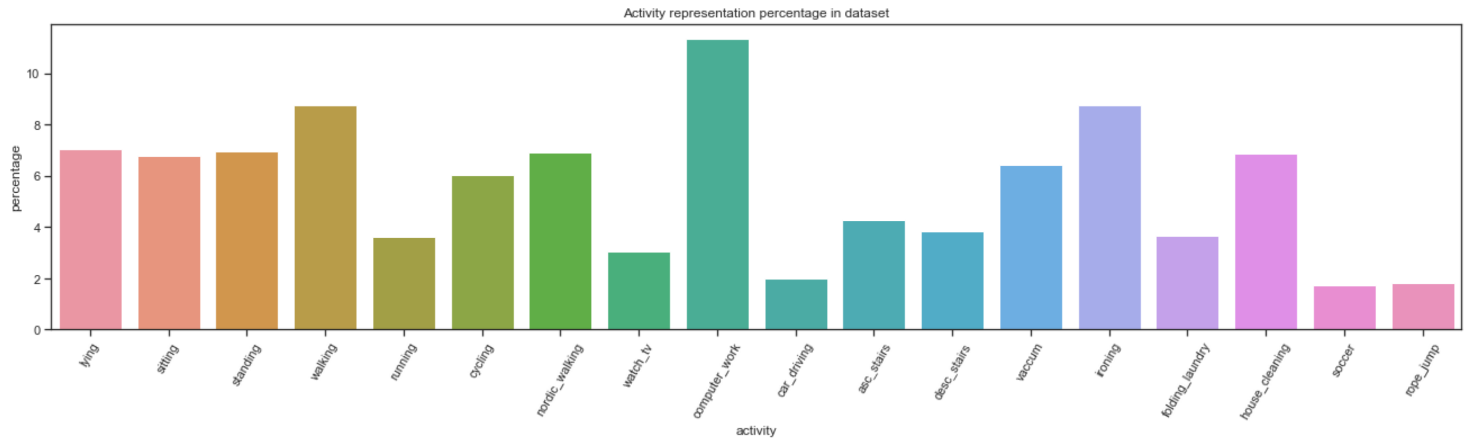


Lying activity



Running activity

Data Balance Analysis



Although we can see from the chart above that we have variation between the representation percentage of the different classes in the dataset, we will use **Shannon Entropy** to calculate the imbalance.
In information theory, **Shannon Entropy** is an estimation of the average amount of information stored in a random variable.
For example, an unbiased coin flip contains 1 bit of information with each result. The entropy of a random variable can be estimated from a series of results.

The entropy value meaning is:

- 0 when there is one single class. In other words, it tends to 0 when your data set is very unbalanced
- Log(k) when all your classes are balanced of the same size nk

Therefore, dividing the entropy by log(k), where k is the number of classes will give us a measurement for balance where:

- 0 for a unbalanced data set
- 1 for a balanced data set

$$H = -\sum_{i=1}^{k} \frac{c_i}{n} \log \frac{c_i}{n}. \qquad \text{Balance} = \frac{H}{\log k} = \frac{-\sum_{i=1}^{k} \frac{c_i}{n} \log \frac{c_i}{n}.}{\log k}$$

```
Entropy value H: 2.773562
Balance value: 0.959586
Balance percentage: 95.9586%
```

We can see that the data is generally balanced, no further action needed

1.c. Self-Supervised tasks suggestions

We can perform multiple self-suprvised task on the data to gain further insights.

Self-supervised task suggestions:

- Forecasting the next heart rate (HR) measurement according to the full measurements taken in a timeframe of x measurements samples.
- Predicting the median measurements according to past measurements time frame and future measurements time frame
- Predicting past measurement according to future measurements time frame

Classification Problem

The problem we wish to address is the classification of consecutive time windows to the activity that was performed in this time segment.

We will use the Sliding window technique with a window look back value of 200.

Since data was measured and interpolated to fit the measurment rate of 100Hz, meaining measurment every 0.01 second,

all window segments will contain 2 seconds of data measurments.

For each of the 2 seconds HR-monitor and IMU sensors measurments we wish to determine what was the activity that produced those measurments (undependent of the performing subject).

hirerchal classification - activities - windows where there is one action - we want to predict the current activity 100hz we want 150-200 timepoint window - maybe interpolate + describe dataframe

2.a. Preprocessing steps and Validation Strategy

Preprocessing steps

- We will segment our data into individual activities while ignoring the subject ID that performed that activity.
  This operation will result in lists of timeframes each one belong to specific activity.
  We will segments all the activities from each subject.
- For every subject we will:
  - Ignore the following columns: subject_id, activity_id, timestamp.
    We want to ignore those columns from the following reasons:
    subject_id - The subject ID that performed that activity will not be relevant to out model and we want it to generalize for every subject.
    activity_id - This is the perdiction we want out model to eventually conclude on.
    timestamp - Since all our data is given in interval of 100Hz and contains all values after interpolation, the specific time the activity was performed in the sequence is not relevant.
    We only need to account for the order of the measurments within each activity and not for the time the activity was performed relative to other activities (We only perdict individual activites with no activity overlap)
  - Scale the specific subject data values after removel of irelevant columns to range [0,1] using the scaler we fitted earlier on the entire data
- We will one-hot encode our activity_id values(y)
- For every activity timeframe recorded in the previous step we will produce sliding windows with length 200 (2 seconds since the measurments are taken in 100Hz)
- **Preprocessing output:** after all the steps we will get X values corresponding with timeframe windows of length 200 from a specific activity and y values that are the one-hot encoded activity if for every timeframe windows

Validation Strategy

As a validation stategy we will use **stratisfied train-test split** in order to preserve the samples representation and include all classes in the training and validation data.
So our validation data will be a statisfied subset of the training windows data taken from the same range of activities.
Additionaly, we will shuffle our training, validation and testing data after divided to windows (since the internal order of the activity timeframe windows matter but not the order between different windows.

2.b. Naïve baseline solution

As a naive baseline solution, we will use stratified Dummy Classifier.
This stratified dummy classifier will produce dummy predictions so that the predicted class will be the according to the relative probability from the entire train data and the class distribution for each category.
For each row (individual measurement datapoint) in the data the classifier will classify it to class x int the probability of x to be chosen randomly from the data.

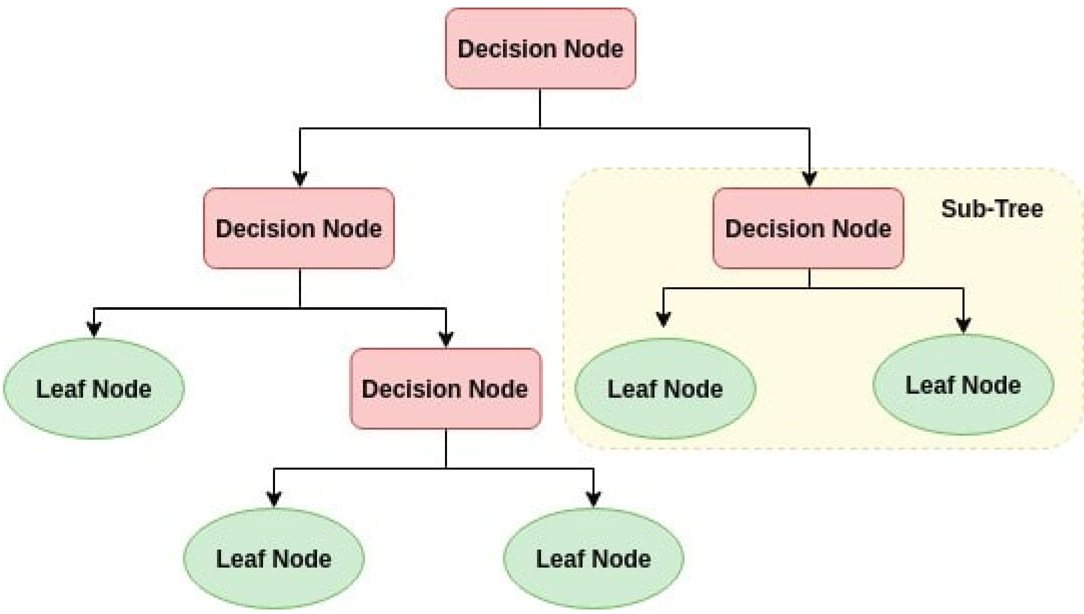
Solution Results:
```
Train Accuracy: 6.74%
Test Accuracy: 6.80%
```

2.c. Classical ML algorithms solid benchmark

We will fit our data (rows from measurments from each activity) into two ML algorithms in order to get a solid benchmark for the future LSTM model performance.
Each algorithm will get individual datapoints as X values (rows from activities) and the expected outcome will be the activity the measurment was taken from as y value.
We will use: **Decision Tree and Random Forest as an extension**

Decision Tree Classification Baseline

A **Decision Tree** is a flowchart-like tree structure where an internal node represents feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome.
The topmost node in a decision tree is known as the root node.
It learns to partition on the basis of the attribute value.
It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making.

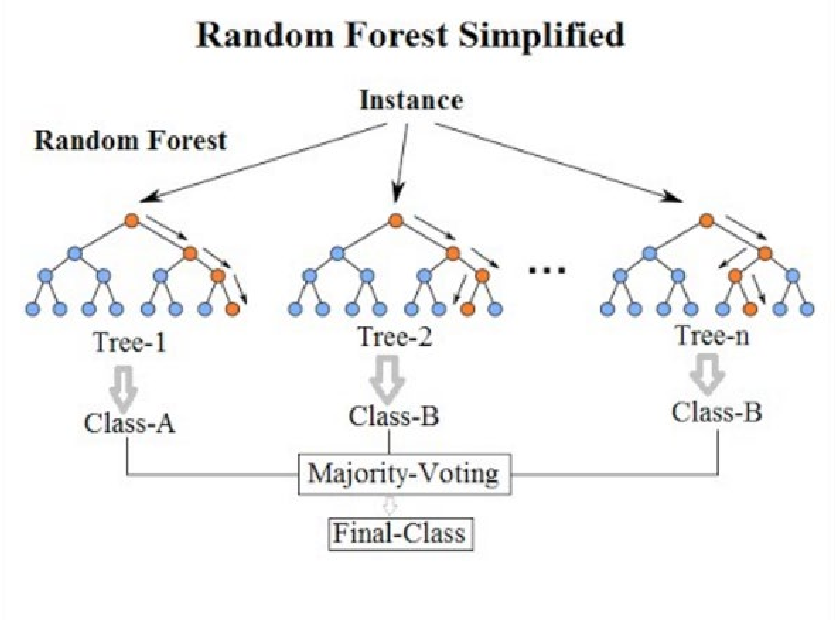


Solution Results:
```
Train Accuracy: 100.00%
Test Accuracy: 24.55%
```

Random Forest Classification Baseline

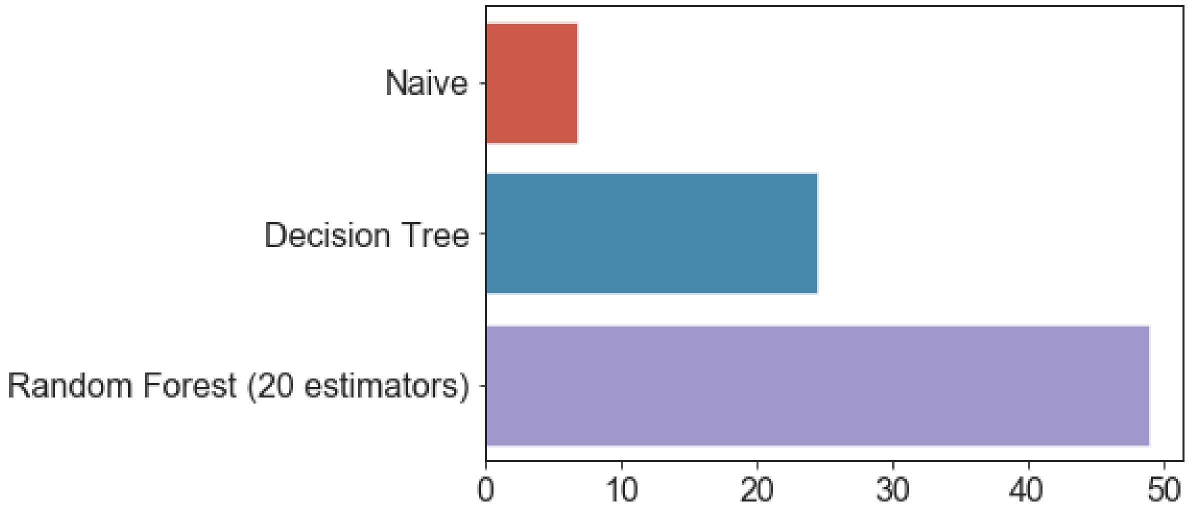
We will try to achieve better baseline results by using multiple decision trees in the form of Random Forest classification ML algorithm.

Random forests or **random decision forests** are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.
Random decision forests correct for decision trees' habit of overfitting to their training set.
Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.



Solution Results:
```
Train Accuracy: 100.00%
Test Accuracy: 48.95%
```

Baseline Models accuracy comparison

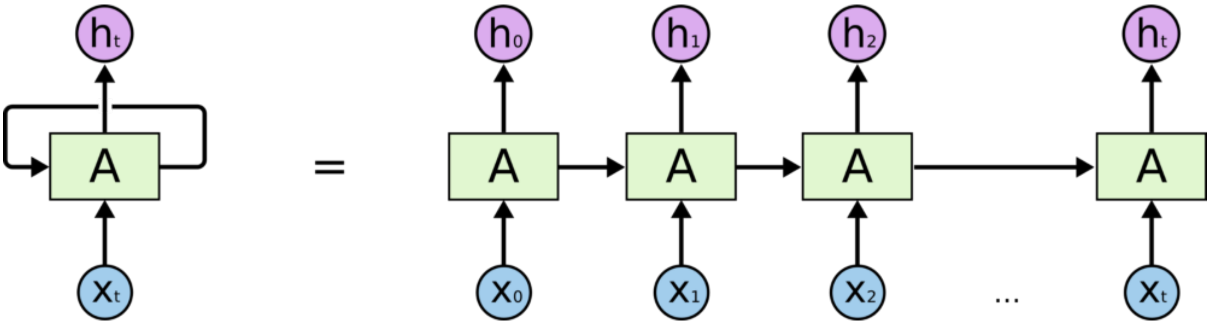


2.d. Classification using LSTM NN

In this section we will use Neural Networks in order to solve our classification problem.
We will use LSTM layer to account for our measurements data window sequences.

About RNNs

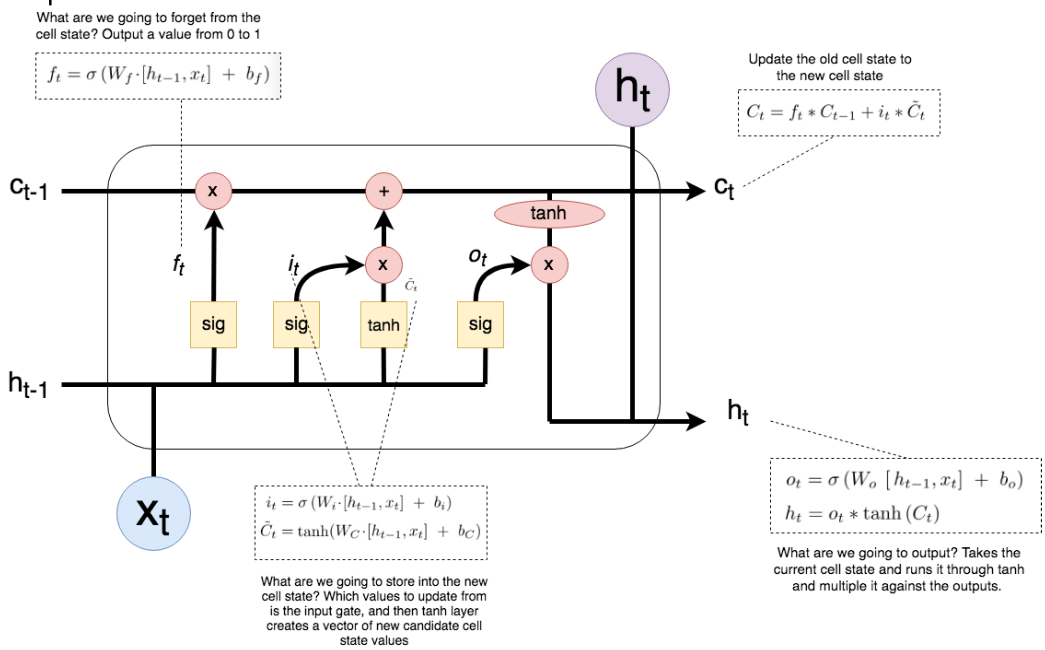
RNN's map an input sequence to an output sequence. RNNs can be used in interesting sequencing tasks. Ranging from machine language translation to time series forecasting.
With a simple RNN, there is the problem of "vanishing gradients", meaning the farther back the loss is propagated,
the rate approaches zero meaning that information about the sequence properties are not saved.
This is solved using Long Short Term Memory Units (LSTMs).



About LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture.
Unlike standard feedforward neural networks, LSTM has feedback connections.
It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).
For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

In this task we will use LSTM in order to process our measurements windows sequence of datapoints.



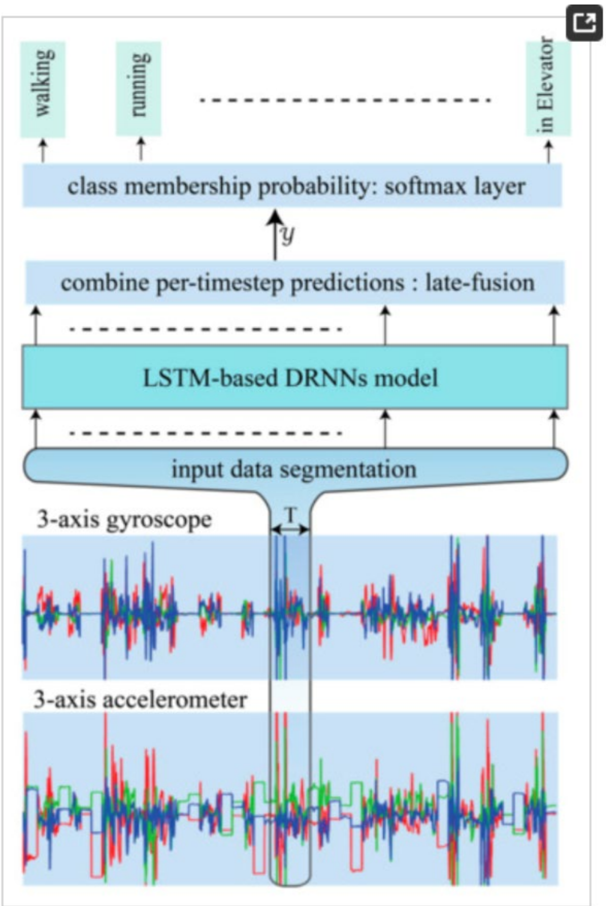
Using LSTMs in an activity classification model

The main idea is to feed the model with time frame of measurements data to try and predict the activity from the pattern of the data by using an LSTM network to remember past measurements

```
Train data input shape: (1647351, 200, 52)
Train data output shape: (1647351, 18)


Validation data input shape: (411838, 200, 52
)
Validation data output shape: (411838, 18)


Test data input shape: (640839, 200, 52)
Test data output shape: (640839, 18)
```

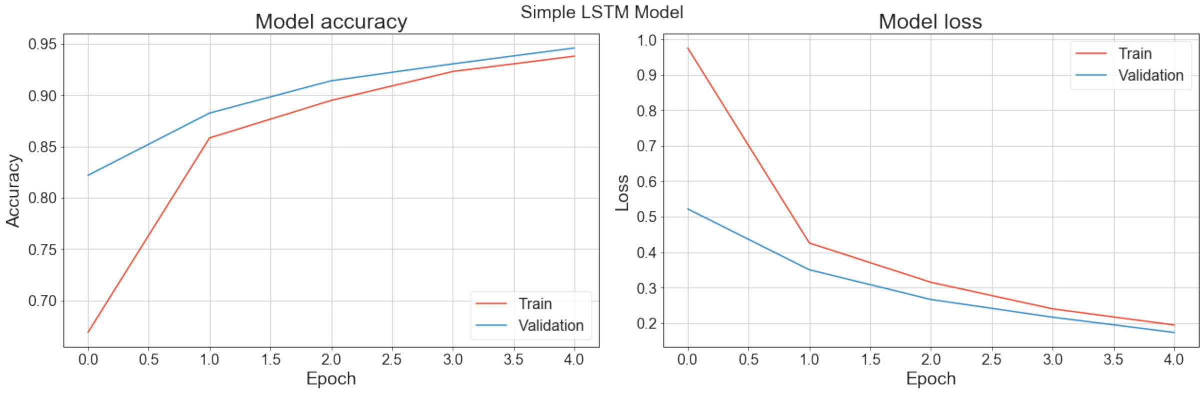


Activity classification by timed measurments data illustration

Initial LSTM model

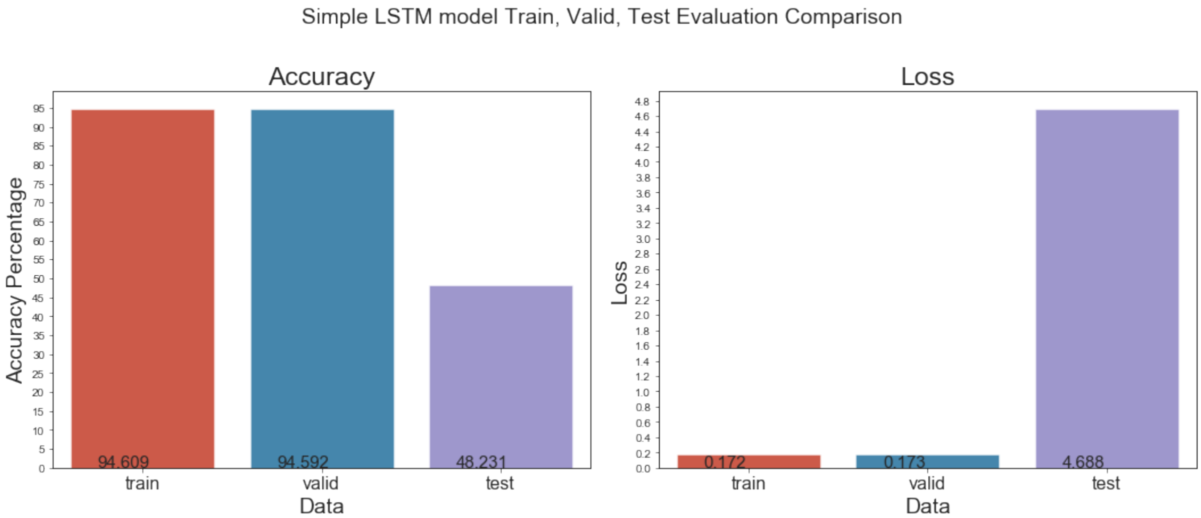
```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 200, 52)]         0
_________________________________________________________________
lstm (LSTM)                  (None, 6)                 1416
_________________________________________________________________
dense (Dense)                (None, 32)                224
_________________________________________________________________
dense_1 (Dense)              (None, 18)                594
=================================================================
Total params: 2,234
Trainable params: 2,234
Non-trainable params: 0
```

Training the model:



Solution Results:

```
Evaulate train loss: 0.1722 / Evaluate train accuracy: 94.609%
Evaulate validation loss: 0.1731 / Evaluate validation accuracy: 94.592%
Evaulate test loss: 4.6883 / Evaluate Test accuracy: 48.231%
```



Looking at classification samples:

Good classifications

0	computer_work	computer_work	99.999642
1	cycling	cycling	99.997652
2	walking	walking	99.994278
3	house_cleaning	house_cleaning	99.975330

We can see that the best prediction have very high classification percentage from the start.
Let's take a look at some of the worst classifications:

Worst classifications

0	computer_work	lying	99.998152
1	sitting	lying	99.994898
2	nordic_walking	running	99.722618
3	asc_stairs	desc_stairs	94.872826

It is clearly visible the model struggles with classifying similar passive activities and similar active activities.

Overall, We can see that the LSTM model performed relatively well, yet did not pass the benchmark score we got by using the Random Forest ML classifier

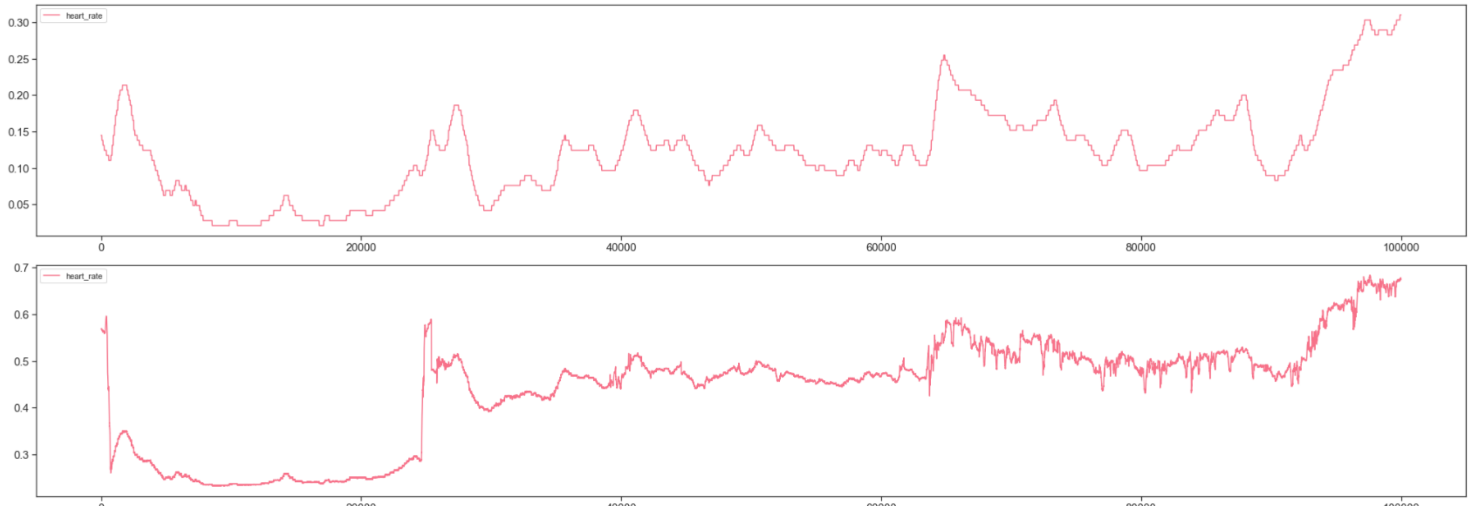
2.e. Pretrain model for forecasting task and fine-tune for classification

We are going to pretrain our model to the previously suggested task of forecasting the next time step heart rate according to the previous 200 measurements samples.
After the model is trained for the heart rate forecasting task we will fine-tune it to conform to the classification task we are after and see if we can get better results.

Model architecture before fine tuning:

```
Model: "functional_91"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_31 (InputLayer)        [(None, 200, 52)]         0
_________________________________________________________________
lstm_48 (LSTM)               (None, 6)                 1416
_________________________________________________________________
dropout_31 (Dropout)         (None, 6)                 0
_________________________________________________________________
dense_76 (Dense)             (None, 32)                224
_________________________________________________________________
dense_77 (Dense)             (None, 1)                 33
=================================================================
Total params: 1,673
Trainable params: 1,673
Non-trainable params: 0
_________________________________________________________________
```

Forecasted HR values (first 100,000):



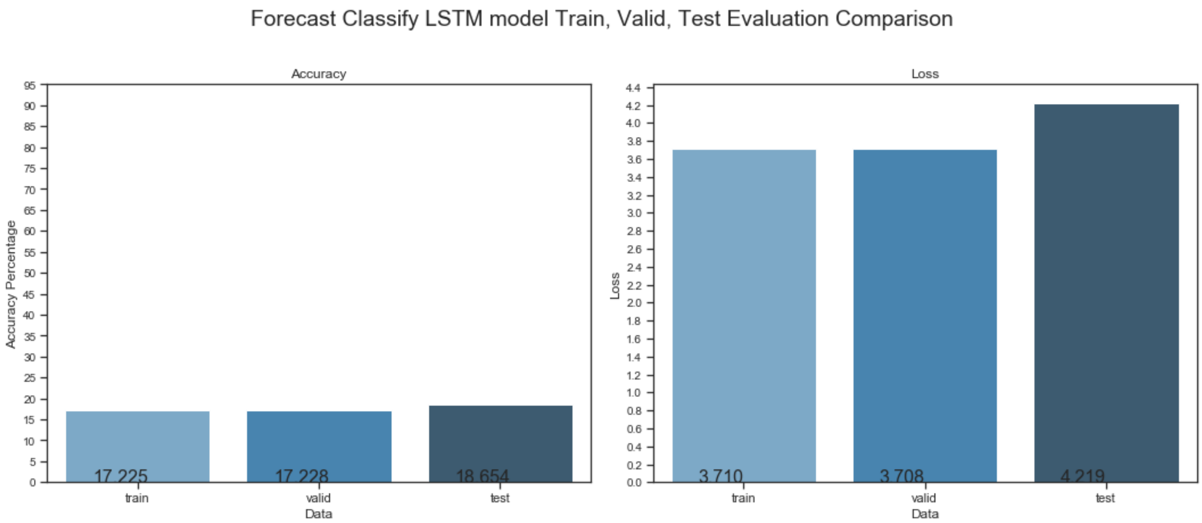
True values (Top) vs. Prediction values (Bottom)

Fine tuning the model to the classification task:

```
After changing last layer to classification:
Model: "functional_97"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_33 (InputLayer)        [(None, 200, 52)]         0
_________________________________________________________________
lstm_50 (LSTM)               (None, 6)                 1416
_________________________________________________________________
dropout_33 (Dropout)         (None, 6)                 0
_________________________________________________________________
dense_80 (Dense)             (None, 32)                224
_________________________________________________________________
batch_normalization_7 (Batch (None, 32)                128
_________________________________________________________________
dropout_34 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_82 (Dense)             (None, 100)               3300
_________________________________________________________________
dense_83 (Dense)             (None, 18)                1818
=================================================================
Total params: 6,886
Trainable params: 5,182
Non-trainable params: 1,704
_________________________________________________________________
```

Solution Results:

```
Evaulate train loss: 3.7101 / Evaluate train accuracy: 17.225%
Evaulate validation loss: 3.7083 / Evaluate validation accuracy: 17.228%
Evaulate test loss: 4.2188 / Evaluate Test accuracy: 18.654%
```



We can see that we achieved worse results than our basic LSTM model and our baselines. We will continue trying to improve the previous simple LSTM model.

2.f. Improvement suggestions and first LSTM iteration summary

Why is the model fitting well on training and validation data but less for test data?

In our opinion the 4 main reasons for the good accuracy and loss ratings of the model on the validation data versus the accuracy and loss rating of the testing data are:

1. The validation data and training data both contain timeframes from the same exact activities and were both taken from the same subset of subjects (101, 102, 103, 104, 105, 106, 109)
2. The validation data and training data both contain samples from each class and since we use stratified train-test split to produce the validation data, the activity representation percentage in the training data and validation data is similar.
   The testing data was taken from different subjects.
3. The testing data does not contain any samples for two of the classes the model was trained on (No samples for watching TV and Car driving activities).
   The training/validation data contain samples from all classes available to us.
4. The testing data is from a whole different subject subset (107, 108) and therefore does not contain timeframes that intersect with the timeframes the model was trained on.

Improvements suggestions going forward

- Increasing node count for LSTM layer to increase look back memory
- Add stacked LSTM layers to increase depth of the model (First LSTM output will be the second LSTM input, with return sequences flag for the first LSTM)
- Add 1D Time Distributed Convolutional layer before LSTM layers to increase the dominant and average features capturing for before passing to to the LSTM layers
- Add more dense layers after the LSTM layers with dropout to add model complexity and try better distinct similar features

2.e. Improving the model

We will implement the following improvements:

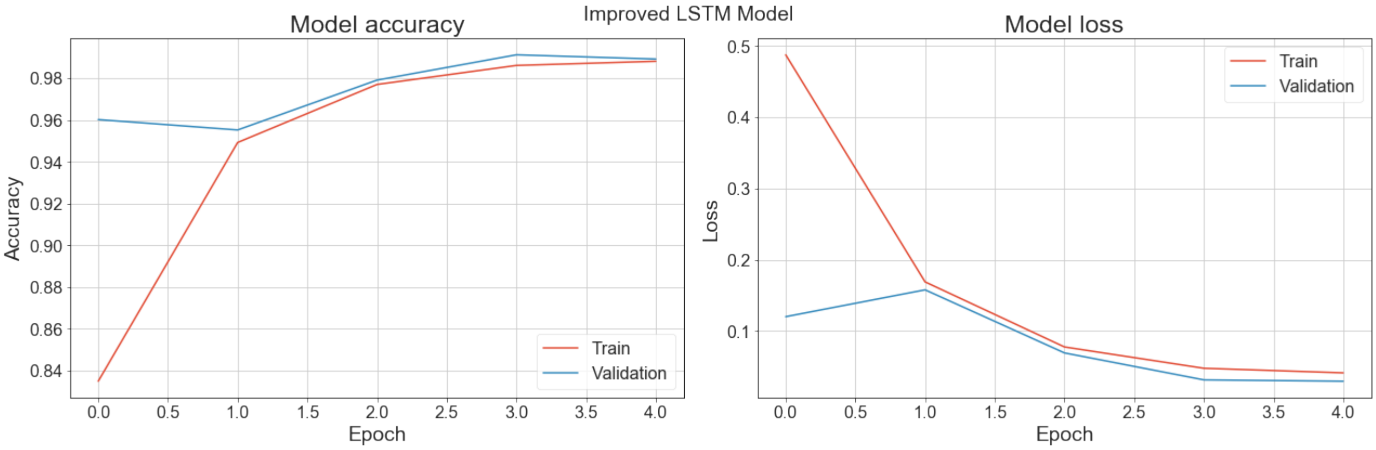
1. Increasing node count for LSTM layer to increase look back memory
2. Add 1D Time Distributed Convolutional layer before LSTM layers to increase the dominant and average features capturing for before passing to to the LSTM layers

For both of those improvements we will additionally add more dense layer to try and capture more complex features.

First LSTM model improvement (Adding LSTM nodes)

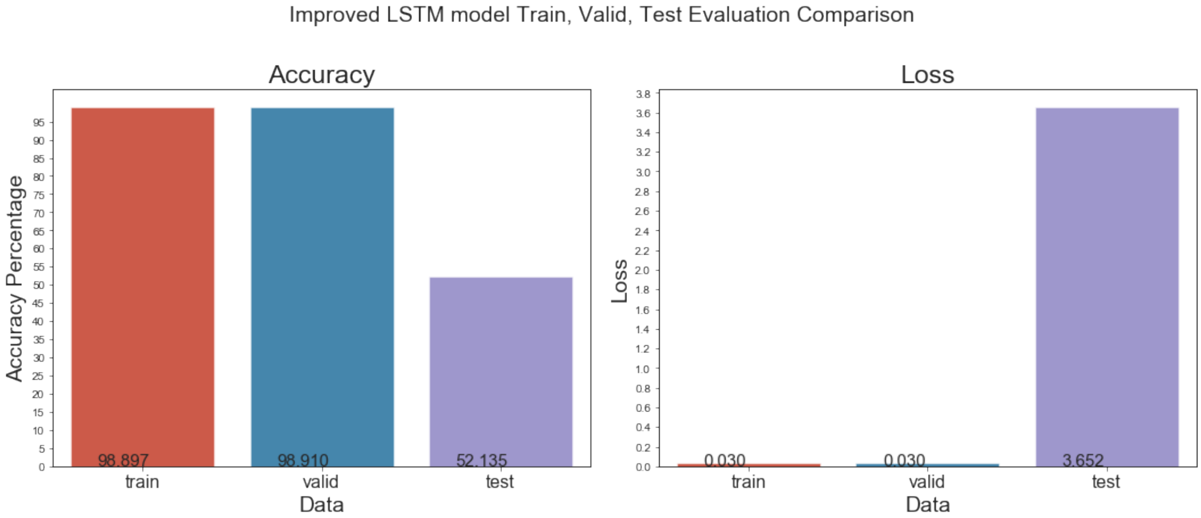
```
Model: "functional_25"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_13 (InputLayer)        [(None, 200, 52)]         0         
_________________________________________________________________
lstm_12 (LSTM)               (None, 100)               61200     
_________________________________________________________________
dropout (Dropout)            (None, 100)               0         
_________________________________________________________________
dense_24 (Dense)             (None, 100)               10100     
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0         
_________________________________________________________________
dense_25 (Dense)             (None, 100)               10100     
_________________________________________________________________
dense_26 (Dense)             (None, 18)                1818      
=================================================================
Total params: 83,218
Trainable params: 83,218
Non-trainable params: 0
_________________________________________________________________
```

Training the model:



Solution Results:

```
Evaulate train loss: 0.0296 / Evaluate train accuracy: 98.897%
Evaulate validation loss: 0.0296 / Evaluate validation accuracy: 98.910%
Evaulate test loss: 3.6519 / Evaluate Test accuracy: 52.135%
```



Looking at classification samples:

We can see that the model prediction percentage is higher than the previous iteration.
We can also observe that the worst prediction still mainly involves active vs. passive activities as points where the model mostly struggles.

Top Correct Predictions Samples from each label:

	Predicted Label	True Label	Prediction Percentage
0	computer_work	computer_work	100.000000
1	lying	lying	100.000000
2	house_cleaning	house_cleaning	99.999976
3	cycling	cycling	99.999917
4	walking	walking	99.999821
5	vaccum	vaccum	99.998140
6	asc_stairs	asc_stairs	99.994457
7	nordic_walking	nordic_walking	99.975449
8	standing	standing	99.975091
9	desc_stairs	desc_stairs	99.931300
10	ironing	ironing	99.927312
11	running	running	99.260086
12	sitting	sitting	97.228575
13	soccer	soccer	96.992797
14	rope_jump	rope_jump	96.364003

Top Worst Prediction Samples from each label:

	Predicted Label	True Label	Prediction Percentage
0	computer_work	sitting	100.000000
1	lying	asc_stairs	100.000000
2	house_cleaning	vaccum	99.999952
3	cycling	nordic_walking	99.999809
4	walking	cycling	99.999690
5	desc_stairs	running	99.984944
6	asc_stairs	desc_stairs	99.983215
7	vaccum	ironing	99.977535
8	running	rope_jump	99.977165
9	standing	vaccum	99.966455
10	rope_jump	soccer	99.955338
11	folding_laundry	ironing	99.783081
12	sitting	standing	99.752516
13	nordic_walking	soccer	99.738234
14	soccer	nordic_walking	97.129130
15	ironing	vaccum	96.703279
16	car_driving	asc_stairs	36.507136

We can see that we got an improvement and surpassed our Random Forest benchmark, yet we can further improve by using a more complex model involving CNN.

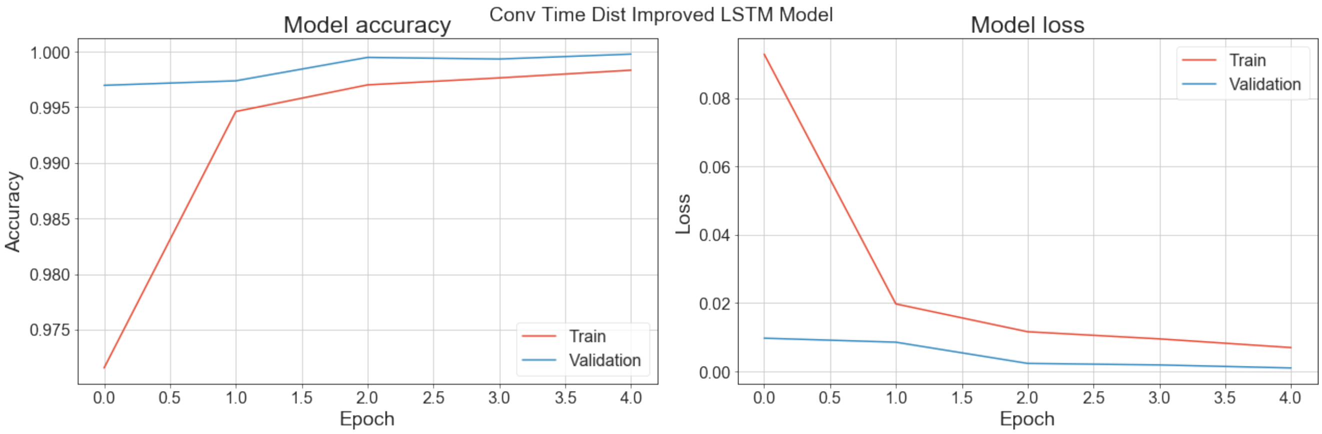
Second LSTM model improvement (Adding CNN and LSTM combination)

Add Time Distributed 1D Convolutional layers

Added 2 1D Convolutional layers with concatenation of the max pooling and average pooling to be the input for the LSTM layer.

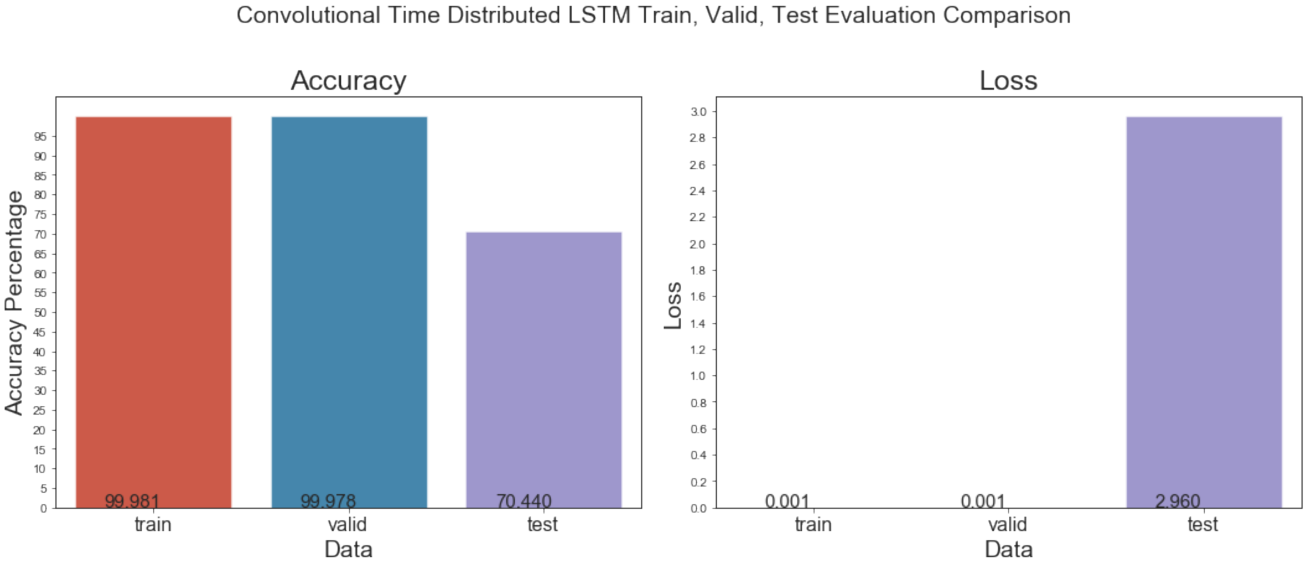
```
Model: "functional_39"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_20 (InputLayer)           [(None, 200, 52, 1)] 0
__________________________________________________________________________________________________
time_distributed_40 (TimeDistri (None, 200, 50, 32)  128         input_20[0][0]
__________________________________________________________________________________________________
time_distributed_41 (TimeDistri (None, 200, 50, 32)  128         time_distributed_4
0[0][0]
__________________________________________________________________________________________________
time_distributed_42 (TimeDistri (None, 200, 48, 32)  3104        time_distributed_4
1[0][0]
__________________________________________________________________________________________________
time_distributed_43 (TimeDistri (None, 200, 48, 32)  128         time_distributed_4
2[0][0]
__________________________________________________________________________________________________
time_distributed_44 (TimeDistri (None, 200, 48, 32)  0           time_distributed_4
3[0][0]
__________________________________________________________________________________________________
time_distributed_45 (TimeDistri (None, 200, 16, 32)  0           time_distributed_4
4[0][0]
__________________________________________________________________________________________________
time_distributed_46 (TimeDistri (None, 200, 16, 32)  0           time_distributed_4
4[0][0]
__________________________________________________________________________________________________
concatenate_5 (Concatenate)     (None, 200, 16, 64)  0           time_distributed_4
5[0][0]
                                                                 time_distributed_4
6[0][0]
__________________________________________________________________________________________________
time_distributed_47 (TimeDistri (None, 200, 1024)    0           concatenate_5[0][0
]
__________________________________________________________________________________________________
lstm_19 (LSTM)                  (None, 100)          450000      time_distributed_4
7[0][0]
__________________________________________________________________________________________________
dropout_15 (Dropout)            (None, 100)          0           lstm_19[0][0]
__________________________________________________________________________________________________
dense_40 (Dense)                (None, 100)          10100       dropout_15[0][0]
__________________________________________________________________________________________________
dense_41 (Dense)                (None, 18)           1818        dense_40[0][0]
==================================================================================================
Total params: 465,406
Trainable params: 465,278
Non-trainable params: 128
```

Training the model:



Solution Results:

```
Evaulate train loss: 0.0009 / Evaluate train accuracy: 99.981%
Evaulate validation loss: 0.0010 / Evaluate validation accuracy: 99.978%
Evaulate test loss: 2.9604 / Evaluate Test accuracy: 70.440%
```



The predicted values are very similar to the values and percentages observed in the previous model iteration.

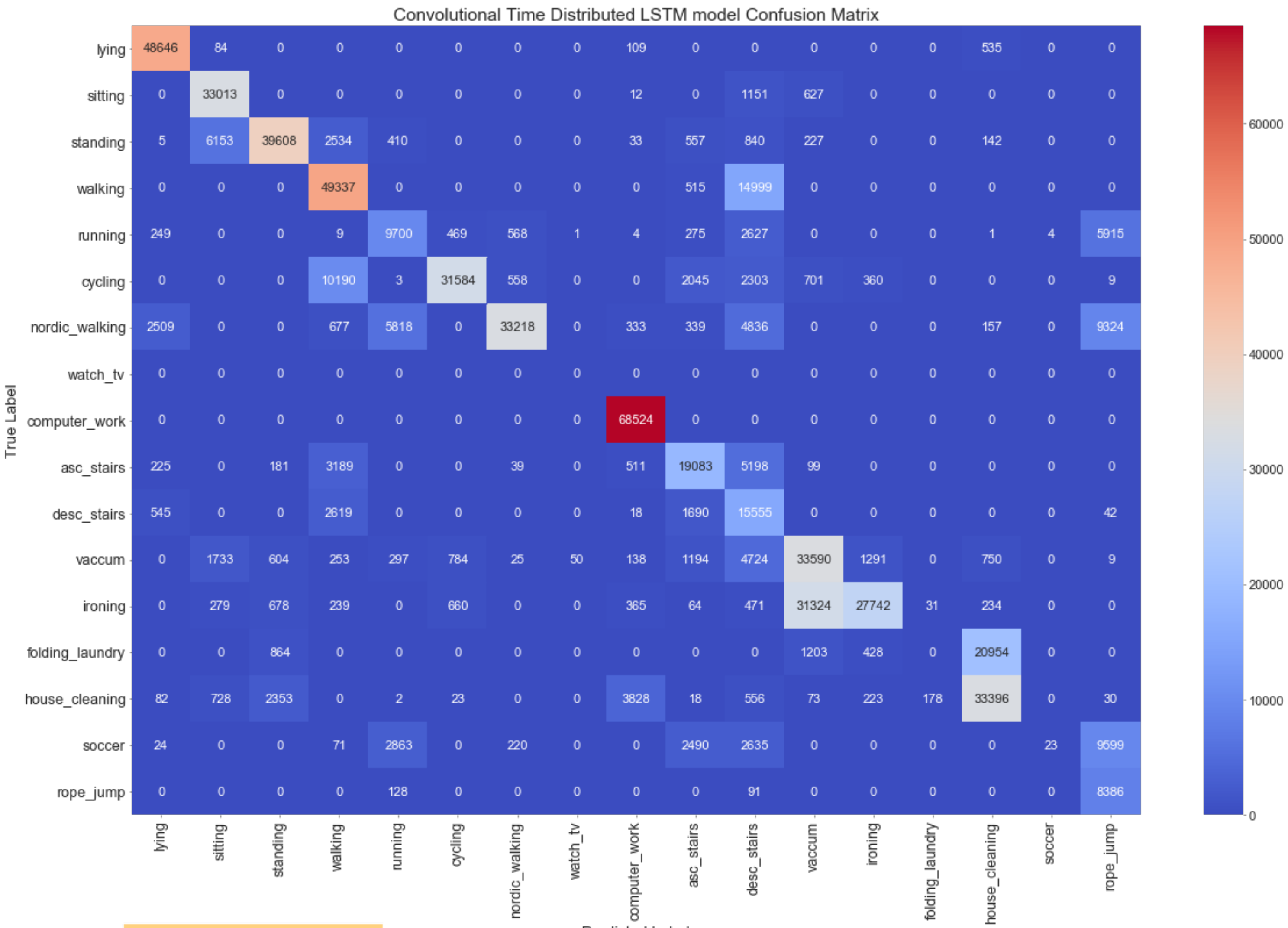
Top Correct Predictions Samples from each label:

	Predicted Label	True Label	Prediction Percentage
0	computer_work	computer_work	100.000000
1	running	running	100.000000
2	nordic_walking	nordic_walking	100.000000
3	lying	lying	100.000000
4	cycling	cycling	100.000000
5	rope_jump	rope_jump	100.000000
6	walking	walking	100.000000
7	house_cleaning	house_cleaning	100.000000
8	sitting	sitting	99.999988
9	ironing	ironing	99.999964
10	desc_stairs	desc_stairs	99.999952
11	vaccum	vaccum	99.999940
12	asc_stairs	asc_stairs	99.999821
13	standing	standing	99.996495
14	soccer	soccer	99.607545

Top Worst Prediction Samples from each label:

	Predicted Label	True Label	Prediction Percentage
0	computer_work	house_cleaning	100.000000
1	running	soccer	100.000000
2	nordic_walking	running	100.000000
3	house_cleaning	folding_laundry	100.000000
4	rope_jump	soccer	100.000000
5	walking	asc_stairs	99.999988
6	desc_stairs	walking	99.999928
7	vaccum	ironing	99.999833
8	sitting	vaccum	99.999750
9	ironing	vaccum	99.999642
10	cycling	vaccum	99.999511
11	lying	desc_stairs	99.999154
12	standing	house_cleaning	99.998939
13	asc_stairs	desc_stairs	99.996495
14	folding_laundry	house_cleaning	96.980876
15	watch_tv	vaccum	95.239538
16	soccer	running	69.355273

CNN + LSTM model confusion matrix



Missing Labels: car_driving

Time Distributed Convolutional Usage Conclusions

We can see that we achieved a big improvement by adding time distributed 1D convolutional layers to our model (~52% - ~70.4%).

We can see that the fact that 2 of the activity types are not represented in out test set hurts affects out testing.

One of the big observations is that the training and validation set (that was taken from the same subset of subjects) almost firs perfectly (higher than 99%).

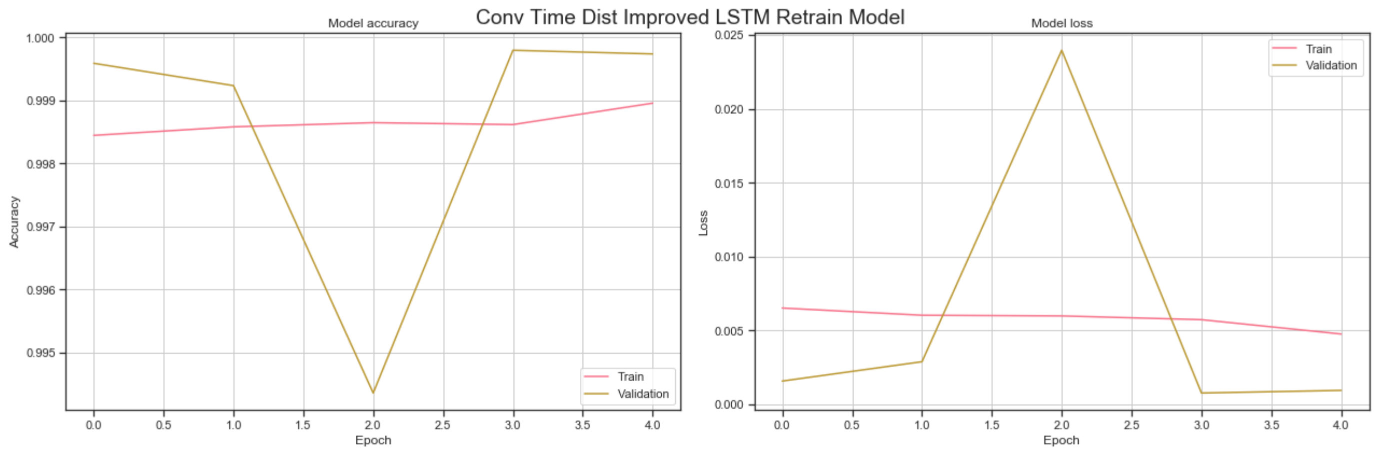
However our test data achieves only 70.4% approx and still have a big loss value.

The reason for it is probably the fact that we chose as our data all the protocol and optional data, that caused inconsistency in the representation of the different activities for different subjects (as not all subjects performed the optional activities).

If we would train the model only for the protocol activities we would not encounter that problem and probably could achieve higher accuracy and lower loss metrics for out test data.

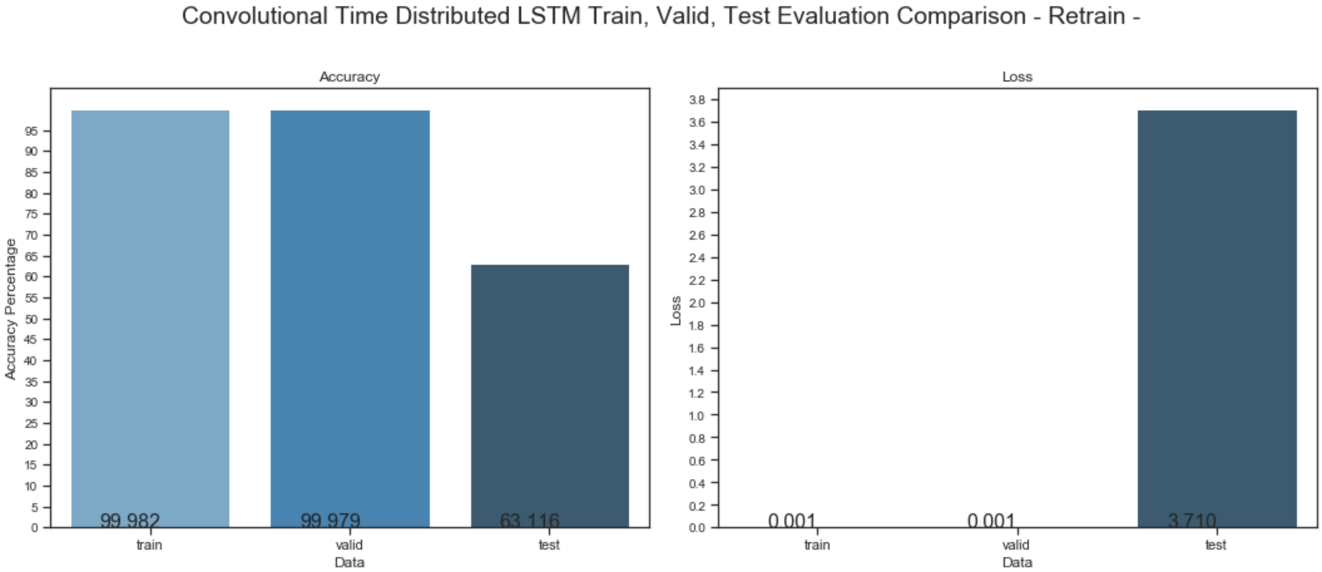
We will try training the same model for 5 more epochs:

Second training of the model:



Solution Results:

```
Evaulate train loss: 0.0007 / Evaluate train accuracy: 99.982%
Evaulate validation loss: 0.0008 / Evaluate validation accuracy: 99.979%
Evaulate test loss: 3.7098 / Evaluate Test accuracy: 63.116%
```



Second train conclusions

We can see that the model increased very slightly when it comes to train/validation metrics. However, we had a big drop in the test data metrics. We can conclude that the model started overfitting at this point and more training will lower our test metrics.

(We cannot really notice the overfitting for the validation data since it was sampled from the same activity time frames as the training data and therefore our model accepts it well as opposed to the training data.

Summary

Results Summary					
Model	Main Features	Batch Size	Epochs	Runtime	Input Shape
Naive Baseline	Stratisfied dummy classifier	-	-	5sec approx.	(None, 52)
Decision Tree Benchmark	Decision Tree Classifier	-	-	8min approx.	(None, 52)
Random Forest Benchmark	Random Forest Classifier: - Decision Tree estimator count: 20	-	-	18min approx.	(None, 52)
Simple LSTM model	Simple model using simple LSTM: - LSTM nodes: 6 - Dense layer with 32 neurons	512	5	23min approx.	(None, 200, 52)
Pretrained heart rate forecast LSTM model	Pretrained model with regression task: - Model was trained to forecast according to 200 time samples (window size 200) the next value of the heart rate. - Loss function for pretrained model: MSE - LSTM nodes: 6 - Dropout: 0.2 - Dense Layer with 32 neurons - Added classification layers: Dropout 0.5 and Dense with 100 neurons	512	5 (forecast) + 5 (classify)	36min approx.	(None, 200, 52)
Improved LSTM model	Improved LSTM model with extra neurons: - LSTM nodes: 100 - 2 Dense Layers 100 neurons each - Dropout between dense layers (0.5, 0.2)	512	5	45min approx.	(None, 200, 52)
Time Distributed Convolutional LSTM model (BEST ACHIEVED)	LSTM model using Time Distributed 1D Convolutional layers and LSTM layer: - 2 x Time Distributed1D Convolutional layers (filters=32, kernel dims=3) - After convolution concatenation of Average and Max Pooling (pool size=3) - Dropout after convolutional layer 0.2 - LSTM nodes: 100 - Dropout 0.5 - Dense Layer with 100 neurons	128	5	7Hr approx.	(None, 200, 52)
Time Distributed Convolutional LSTM model (Second train)	Retrain the previous model for 5 more epochs	128	5	7Hr approx.	(None, 200, 52)

Model	Optimizer	Learning Rate	Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	Loss Function
Naive Baseline	Adam	0.01 (default)	6.74%	-	-	-	6.8%	-	-
Decision Tree Benchmark	Adam	0.01 (default)	100%	-	-	-	25.55%	-	-
Random Forest Benchmark	Adam	0.01 (default)	100%	-	-	-	48.95%	-	-
Simple LSTM model	Adam	0.01 (default)	94.609%	0.1722	94.592%	0.1731	48.231%	4.6883	Categorical Crossentropy
Pretrained heart rate forecast LSTM model	Adam	0.01 (default)	17.225%	3.7101	17.228%	3.7083	18.65%	4.2188	MSE + Categorical Crossentropy
Improved LSTM model	Adam	0.01 (default)	98.897%	0.0296	98.910%	0.0296	52.135%	3.6519	Categorical Crossentropy
Time Distributed Convolutional LSTM model (BEST ACHIEVED)	Adam	0.01 (default)	99.981%	0.0009	99.978%	0.001	70.44%	2.9604	Categorical Crossentropy
Time Distributed Convolutional LSTM model (Second train)	Adam	0.01 (default)	99.982%	0.0007	99.979%	0.0008	63.12%	3.7098	Categorical Crossentropy