# Assignment 2 - report

By: [Adir Serruya](mailto:adirser@post.bgu.ac.il) , [Benjamin Sayag](mailto:sayab@post.bgu.ac.il), Ziv Yogev

# Part 1

Upon reviewing the documentation for the dataset, we discovered the following:

1. A large portion (80%) of the data consists of "transient activities," which are not relevant to the activity being studied as part of the experiment.
2. The sampling rates of the various sensors vary (e.g., heart rate at ~9Hz and IMU at 100Hz).
3. It is recommended to use only the acceleration data with a scale of ±16g.
4. The orientation sensors are not reliable in this data collection.

## Dataset arrangement:

The dataset arrangement is such that in each row there are 54 numbers:

– 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

## Sensors details:

|  |  |  |
| --- | --- | --- |
| **Sensor** | **unit** | **Sampling frequency** |
| Heart rate | BPM | 9Hz |
| IMU (hand, chest, ankle) |  | 100Hz |
| temperature | C |  |
| Acceleration 3D 16g scale | (ms-2) |  |
| Acceleration 3D 6g scale | (ms-2) |  |
| Gyroscope 3D | (rad/s) |  |
| Magnetometer 3D | (μT) |  |

## Our mission

Predict the activity (classification) based on the multivariate time series.

## EDA

* We have 21 different activities tagged that we are using as targets for our classification task. The distribution of these activities is fairly balanced, with the exception of the "transient activities," which are a smaller portion. Each activity makes up around 5% of the total recorded time.
* The activities of each subject vary, meaning that not all subjects perform all 21 activities.
* We noticed that there were a large number (90%) of missing values in the heart rate sensor data for every subject, which was due to differences in sampling frequency.

### Correlation:

* As expected there is high positive correlation between the three temperature sensors (hand, chest and ankle).
* We found a negative correlation between heart rate and temperature, which was opposite to what we had expected.
* We conducted an analysis of the correlations between different activities, separating them into two groups: static and dynamic. The static activities included lying, sitting, standing, watching TV, using a computer and driving a car. The dynamic activities included walking, running, cycling, Nordic walking, climbing stairs, descending stairs, vacuuming, ironing, folding laundry, cleaning the house, playing soccer and rope jumping. Our analysis showed a significant difference in the correlation matrix between the dynamic and static activities, with generally higher correlations observed in the static activities.

***c***

Since we want our model to learn to generalize temporal data the first suggested task would be to train the model to predict the standard deviation of the next sample of the temperature feature, so given a sequence of 33 features we want the model to correctly predict the standard deviation of the next sample of the temperature so in practice we will perform segmentation on the data such that for each sample we will have a sequence of 400 reading of 33 features and the output will be the standard deviation of one of those variables -> and our task will be to predict the standard deviation of the next 400 reading for that variable.

Our second task will be to do the same but this time instead of regression, we will try to predict the trend of that feature, meaning 1 – UP, 0 – Stale or Down.

In practice we chose to implement the 2nd option (classification of trend) due to the nature of our original problem which was classification.

For this task we chose a random feature (22) arbitrarily just to see the effect and because we implemented a neptune experimentation space it would be interesting to later check which feature is the best candidate for such pre-training tasks.

Later on we also tried to predict the trend on other features such as the **heart rate during a 4 sec window.**

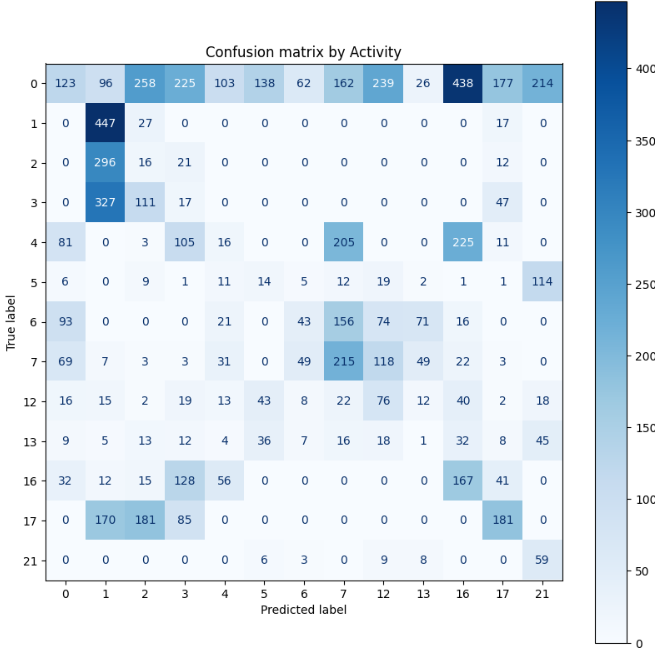
***Part 2***

***Section a***

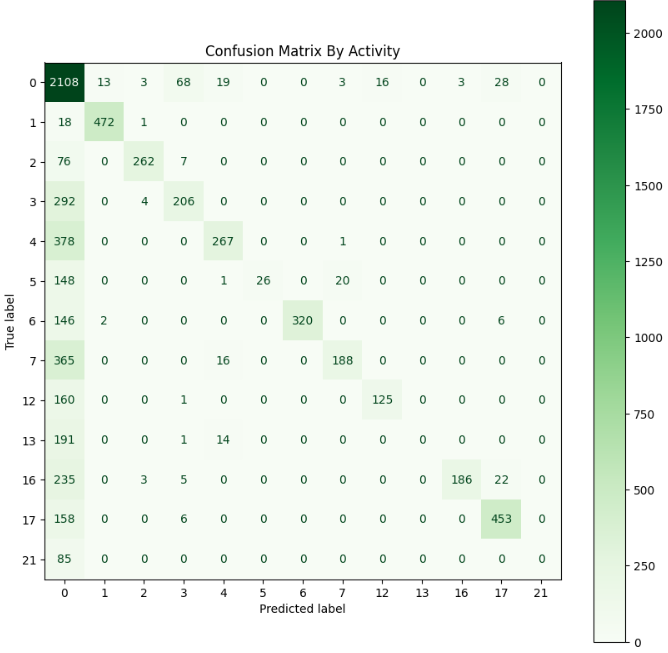
Our validation strategy works in the form of a **train-test split using sliding windows**. For each subject (101,102..109) we extracted the features of each activity he performed and created a new data set in which we actually eliminate the subjects. We did this because we wanted to generalize for a new action that happens so that given a certain time-series, we can classify what the activity is according to the information received from the sensors and not according to the personal characteristics of the subject. In this way we prevent overfitting that could happen if we were to use a *time based group K-fold* strategy in which there is consideration for each validation set in the subject's data. Each sequence we created consists of a window of size 400, with a stride of size 100 (non-overlapping between the windows). Our training set consists of subjects {101:106,109} and the test set consists of subjects {107,108} as we required.

***Section b***

Our naive baseline solution only uses the heart rate for each specific activity. We calculated an **average heart rate for each activity** of the subjects in the training set (after completing the missing data using a linear method). Then for each sequence in the test set we calculated an average of the heart rate, calculated the lowest absolute distance between the heart rate in the test set and the average heart rate in the training set and classified it according to the smallest distance we found. The accuracy percentage for the naive baseline is: 19.29%. The best result presented is for activity 1 - 'lying', but it can be said in general that the model did not classify well, because there was not much reference to the amount of examples from each activity.



***Section c***

In this task we used a basic ML model - **Random forest classifier**. In order to be able to use the model, for each sequence we calculated the averages of each of the features. This is how we converted the 3D table with all the sequences to a 2D table where each row contains concise details for that sequence. We have now trained the tree on the new data set. A significant improvement can be seen both in terms of the overall accuracy (64.7%) and in terms of the additional indices for the classification assessment (precision, recall, f1-score). However, it can be seen that the tree classified activities 13 (descending stairs) and 21 (rope jumping) always incorrectly. A possible reason for this is the low amount of data of activity 21 (only 85 samples in the test set) and another possible reason is that we limited the tree to "max\_features = sqrt".

***Section d***

In this section we builded an LSTM model, stacking five LSTMs together to form a stacked LSTM with output of 31 dimension to match the number of features.

We have trained the model using **80%** of the data from all subjects except for 107 and 108.

We wanted to see the impact of many epochs in comparison to what we have been used to, then we set max epoch to 100 and early stopping to prevent overfit of the model. Early stopping is based on the validation accuracy,so as long as up to 5 epochs there is improvement of accuracy we keep training the model. Patience is the parameter of how many epochs we wait for improvement. In this experiment we used neptune.ai, a monitoring platform to track our experiment. We logged to neptune the accuracy, loss and other parameters than we used the data to evaluate the model.

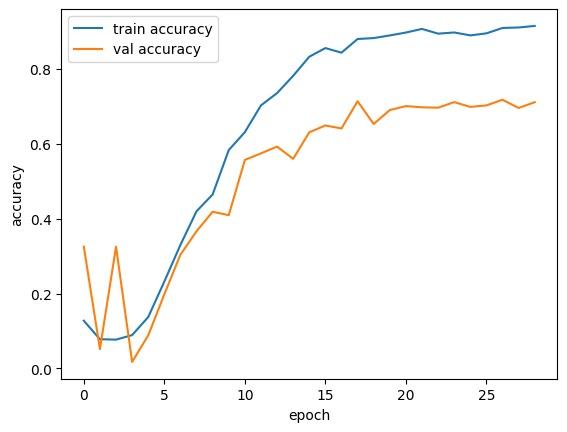
***Model evaluation:***

*Accuracy* & loss

After **23** epochs, we obtained the best loss on the validation set (**0.856**), which was higher than the loss on the training set (**0.33**). When we look at the accuracy metric, we see that the validation set had an accuracy of **0.711**, the training set had an accuracy of **0.897**, and the test set had an accuracy of **0.438**. This suggests that our model may not be complex enough, as it is not able to generalize well enough to the test set. The low accuracy on the test set could be due to the fact that it includes data from subjects 107 and 108, whose activity data has a different distribution. This difference in distribution may have contributed to the poor performance of the model on the test set.

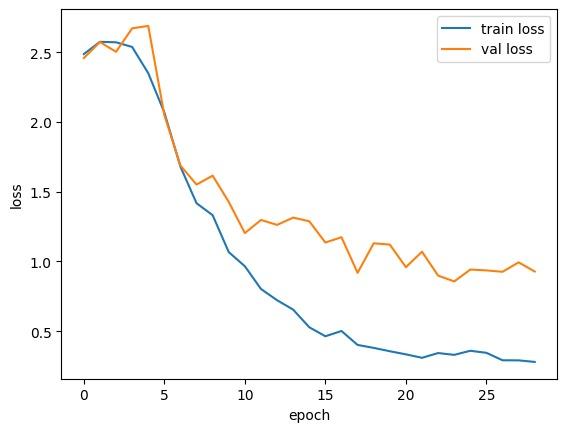
The last epoch (**28**) showed that the accuracy of the validation set (**0.711**) was not as close to the accuracy of the training set (**0.915**) as we would have liked. This doesn't necessarily mean that the model is overfitted, but it is possible that if we had allowed the model to train for more epochs (i.e. had more 'Patience'), it may have fit the data better and avoided overfitting. The graphs below show the accuracy of the model on the training and validation sets over the course of the epochs. In general, we can see that the accuracy on the training set is higher than the accuracy on the validation set.

Training and validation accuracy



Learning curves:

Validation loss is higher than training loss. That means that model performance is better on training dataset, usually we can say it is a good fit but we need to compare the result to the benchmark we created in earlier steps.



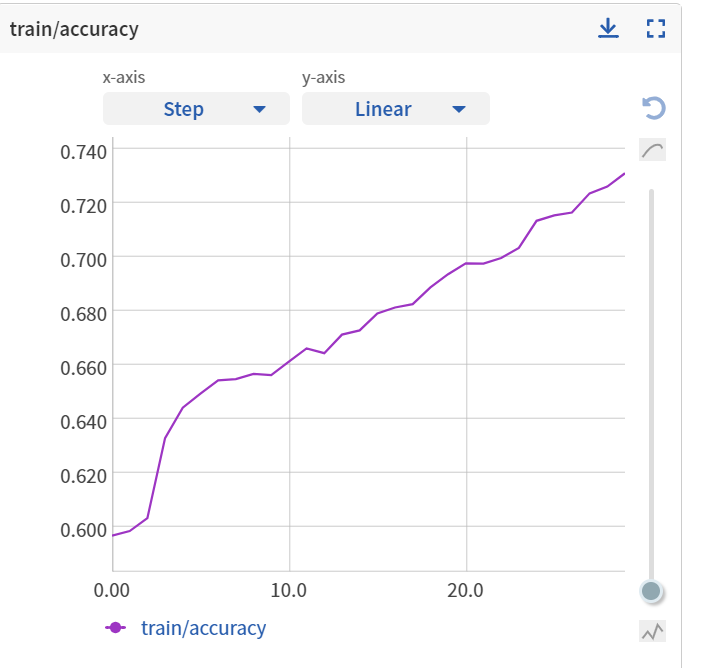
***Section e***

***Pretraining the model***

In this section we pretrained our model on the self-supervised tasks we created at 1-c.

During our first trial, we trained our LSTM Model to predict the trend (1 for uptrend 0 for downtrend or stale) for a random feature and got 87% Train accuracy on that task

While achieving around 85% Accuracy for the validation set. We then pretrained our model on predicting the heartbeat trend. The pretrained model achieved abysmal results and stagnated around 63% Validation accuracy. As can be seen in the charts below.



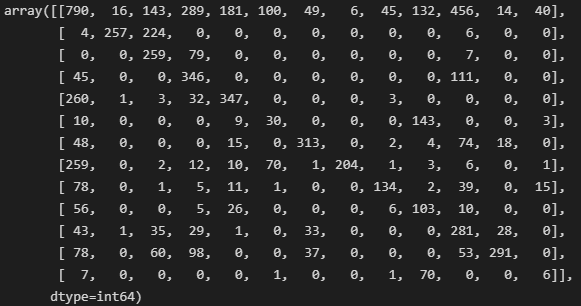
Even though the results we’re not so good for our pre-training task we decided to still see if there was an impact on the original model after being fine tuned on the original task.

Surprisingly we improved our results even when using a pre training task that is not complicated at all and even when we did not achieve good results on the pre training task itself.

The results of the fine-tuned model are:

**Training accuracy of 91.51%** and **Validation accuracy of 80.1 % as can be seen below.**



Looking at our results, it seems the model is doing ok but alot of the time predicting the class 0, and that's probably due to the fact that 90% of the labels are 0, we already chose some preventive actions to solve that issue, we implemented a WeightedRandomSampler instance from torch library which did improve the results but as we can see in the confusion matrix below the model is still struggling. (The results are not on the validation but on the actual test subjects 107, 108)

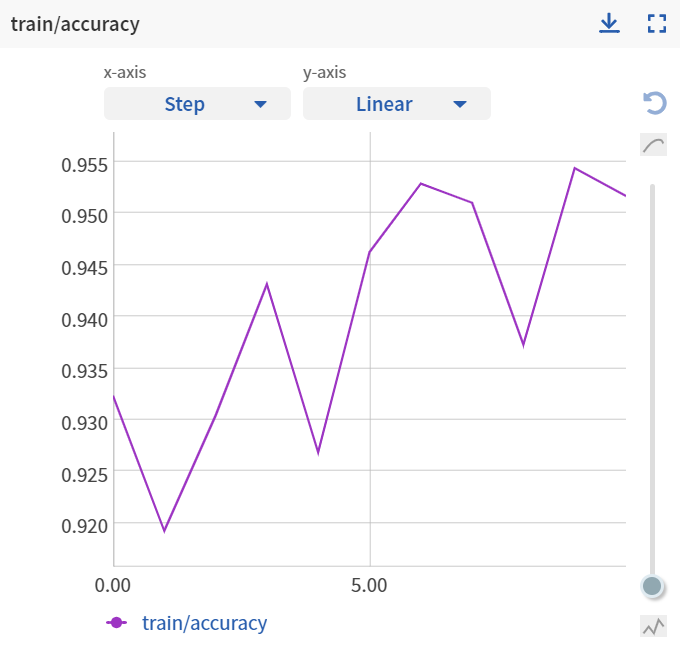
E. 3 Suggestions to improve our are models are:

1. Work with different windows sizes - It could be that a window size of 400 (4 Secs) is not enough to learn discriminating features properly (maybe a longer time frame is needed).
2. Try a different hyperparameter for the current LSTM Model.
3. We can try to retrain our model with a more complex task which could be more representative of the original task.

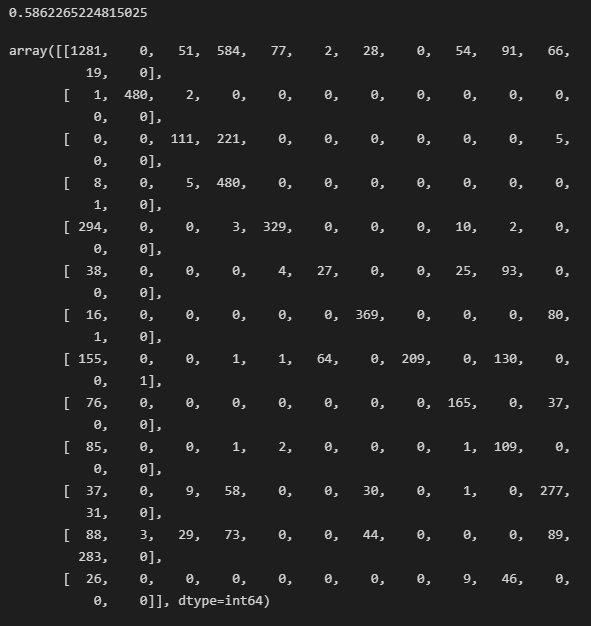
We will try to implement the first 2 suggestions as they seem more reasonable and quicker to check.

Trying windows of size 800 instead of 400 - meaning each sample represents an 8 second time window, hoping it could improve our model results.

We loaded the pretrained model and finetuned it on a the new dataset which is sliced to 800 samples sequences as our samples, looking at the results it seems like our hypothesis is somewhat empirically correct, we can see that our model now generalize better and achieves higher accuracy results both on the training set and the **validation set measuring 88.2% Validation accuracy.**



As we can observe the results on our test set improved drastically as well as can be seen below:



For our 2nd Suggestion we implemented the same LSTM Model but with a tweak, this time allowing for more historic information to be referenced by increasing the size of the hidden dim, meaning the model will be able to track more information from the past.

We doubled the hidden\_dim from 256 to 512 because we saw in the previous improvement that the window size (time frame in seconds) affects the results drastically, making us suspect that allowing the model to refer to later historical data will allow for better learning and historical referencing.

This time, we checked the the effect of such change without pre training the model, we’re observing a big increase over the original trial which achieved maximum validation accuracy of 71%, After increasing both the hidden dims and the window\_size we’re witnessing already above 80% accuracy on the validation set, which is over 9% improvement from the smaller model.

What we mainly wanted to check was the different effect of different window sizes and the ability to store historical data as a reference and how does the model ability to do so affects the results.

As we can see in plots below we could have kept on training the model and maybe get better results but decided to stop there due to computational constraints.

