# Predict Parkinson's Disease using wearable data

Aloi Alfredo, Riccio Francesco

January 26, 2022

**Abstract**

This is a project for the Deep Learning course held in the Master Degree in Computer Science of Università degli Studi della Calabria.

# Contents

# 1 Introduction

Parkinson's disease, the second most neurological disorder that causes significant disability, reduces the quality of life and has no cure. Approximately, 90% of affected people with Parkinson's have speech disorders. Deep learning has the potential to give valuable information after processing that can be discovered through deep analysis and efficient processing of data by decision makers. The main problem is the identification of OFF symptoms perceived by the study subject with reasonable accuracy from real-world data collected. OFF periods are times when Parkinson's disease medication, namely Levodopa, is not working optimally. As a result, symptoms return. OFF periods can include both motor symptoms, such as tremor and rigidity, and non-motor symptoms, such as anxiety. The goal is to consolidate data collection by identifying the most important variables to solve the problem.

# 2 Task 1: next value prediction

We have 3 time series (X, Y, Z) recorded each 10 seconds. For the first sub-task we considered, for each time series, sequences of five minutes (`window_size = 30`) every one minute (`window_shift = 6`), predicting the next value in the series. The evaluation metric is the Mean Absolute Error, in particular we have to go below:

- 81.06 for X;
- 85.26 for Y;
- 79.94 for Z.

We then chose, for the second sub-task, the one time series with the worst value of the evaluation metric, looking for a better combination of window size/window shift.

## 2.1 Data understanding and preparation

We have two files, `train.csv` and `test.csv`, each one containing three columns, representing the three time series X, Y and Z. The `test.csv` file was used for the evaluation, while the `train.csv` file, that contains 144910 rows, was used for the training part.
Since the three time series to be used in the training steps are inside one single dataset, we started the preparation of our data by first splitting it into the three different time series. For each one of the generated time series:

1. we produced a set of sequences, basing ourselves on the aforementioned window size and window shift, with the corresponding labels, representing the element that follows each sequence in the series;
2. we normalized the values inside the produced sequences using a Standard scaler, which scales the data according to a Normal distribution;
3. we, additionally, split the produced sequences to obtain training and validation sets to use in the training step, using a ratio of 80:20.

## 2.2 Modeling

We chose to build a neural architecture starting with three LSTM layers followed by four Dense layers, with the last one made of just one neuron, which is our output. Between each layer of the architecture we chose to put

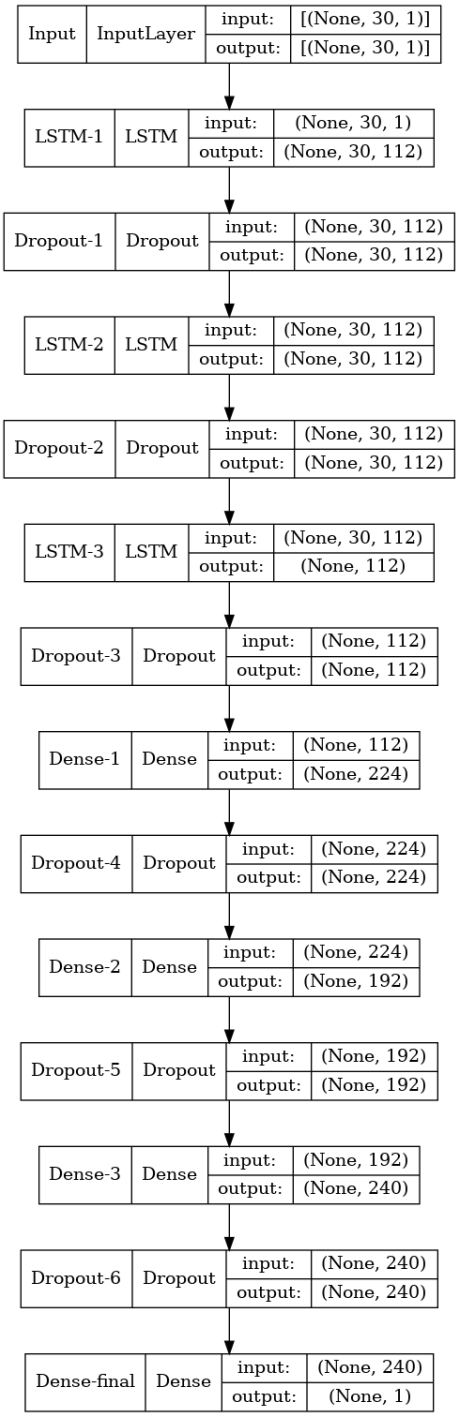


Figure 1: The model for Z time series

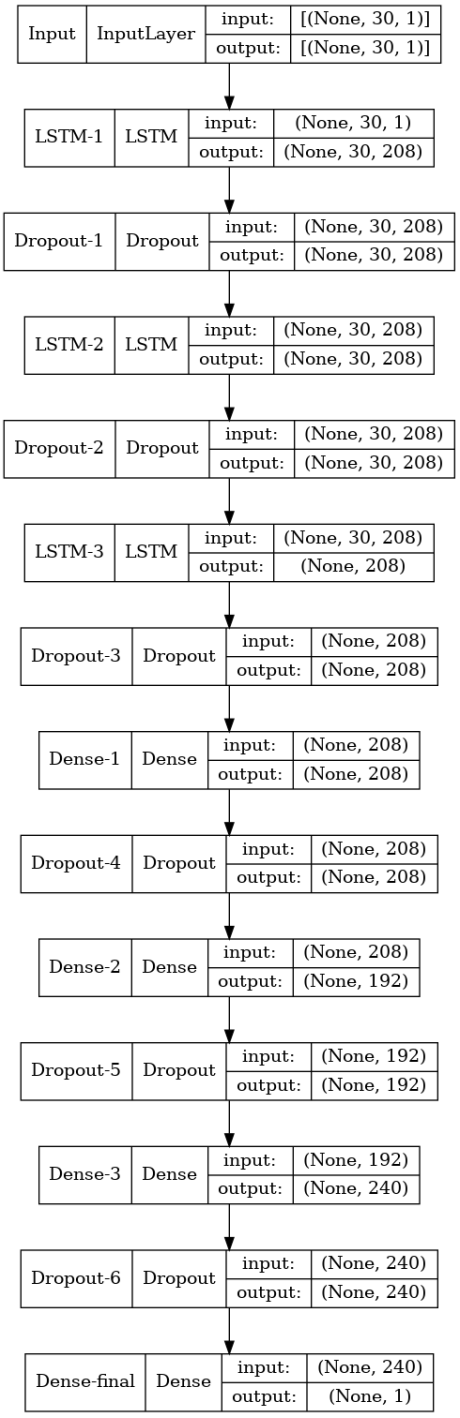


Figure 2: The model for Z time series

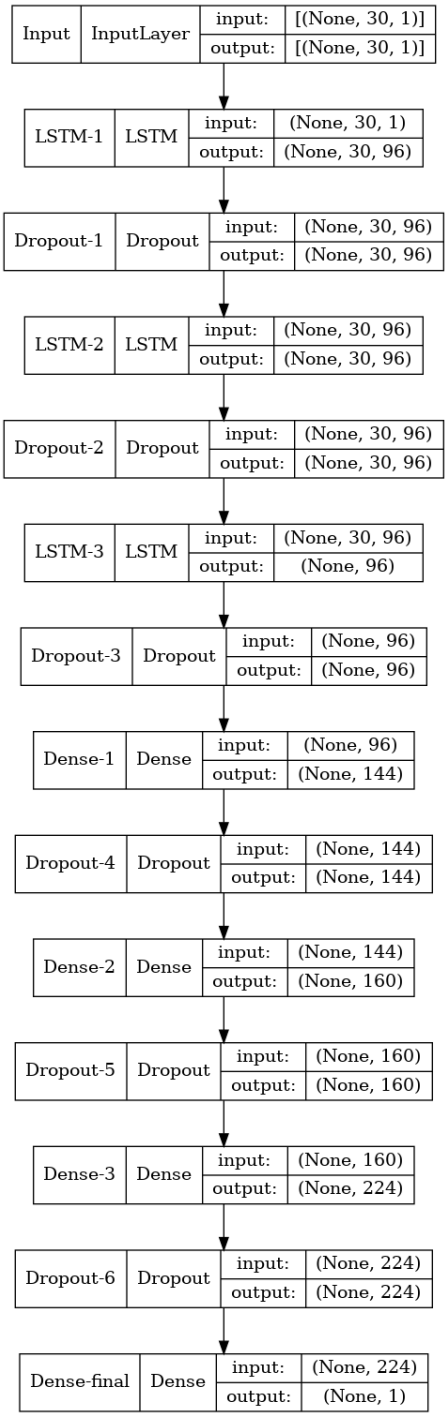


Figure 3: The model for Z time series

a dropout layer to mitigate the overfitting bias. We employed a Hyperband tuner to make the choice of the best model parameters for us. Next, we fit our models for 25 epochs using a batch size of 32, keeping track of the evaluation metric score of the validation data used during training, in order to save only the best behaving model; this was done using a Model Checkpoint as callback to the fitting process.

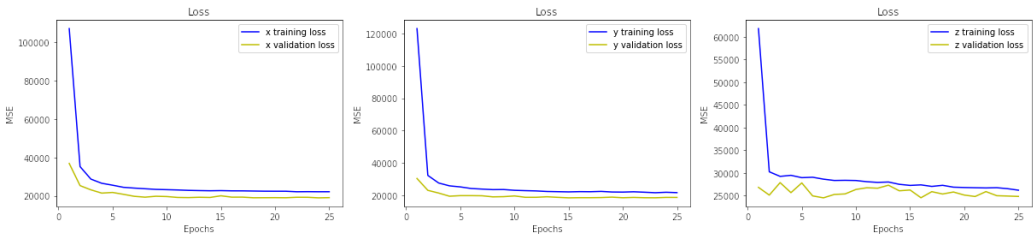


Figure 4: The losses during training

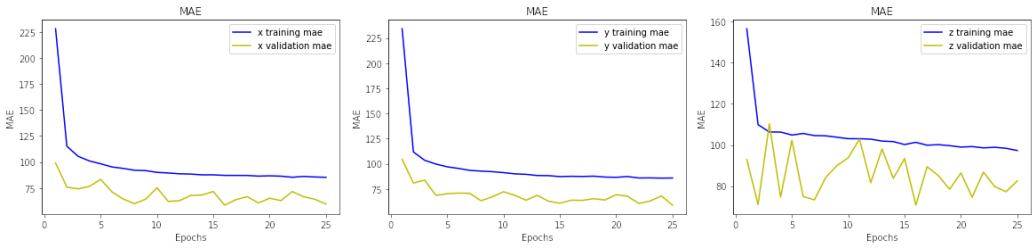


Figure 5: The MAE during training

## 2.3 Evaluation

Looking at the plots regarding the loss (Figure 4), we can observe how in the first epochs the models learn very quickly, suffering a sudden slowdown in the following epochs, but, anyway, the learning curve appears to be quite smooth for each time series. On the other hand, we can notice, in the plots regarding the Mean Absolute Error (Figure 5), that each time series has an irregular behaviour, especially the Z time series, with a high variability in the values of said metric during training.

To evaluate our models, we first had to apply the same preprocessing steps also to the test dataset, namely: splitting of the different time series, generation of sequences and normalization. Concerning the results obtained after testing (Table 1), these are quite good, all the evaluation metrics respect the required objective values. In particular, with regards to the X time series,

| | Target | Actual |
|---|---|---|
| X | 81.06 | **77.62** |
| Y | 85.26 | **84.31** |
| Z | 79.94 | **78.39** |

Table 1: Testing results obtained

we are far below the target threshold (77.62 compared to 81.06), while for the other two the difference is smaller (84.31 and 78.39 compared to, respectively, 85.26 and 79.94).

## 2.4 Finding better parameters

The objective of this subtask is to try to improve the results obtained previously, for the Y time series. Specifically, we wanted to find a better combination of window size and window shift. To make this possible, we defined a search space, consisting of five values for the window size (18, 24, 30, 36 and 42) and four values for the window shift (1, 3, 6, 9). For each combination of window size/window shift we fit the model and test it, saving the respective evaluation. It can be noticed (Figure 6, Table 2) how models with a

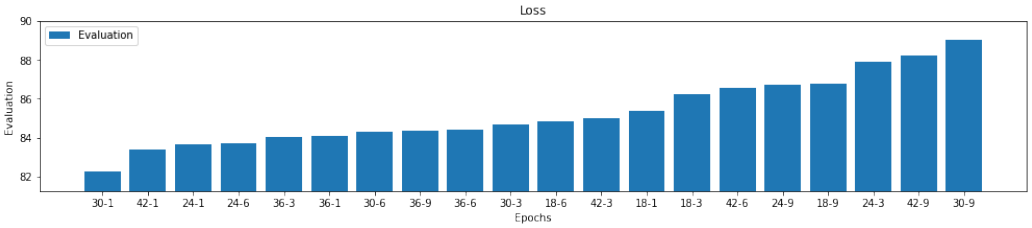


Figure 6: The evaluations for each combination

lower window shift tend to perform better, as opposed to those with a higher window shift, that do not perform in the same way.

## 2.5 All in one model

As an extra step we decided to build a model that takes as input the three time series at once, in order to see if we can obtain more accurate results. We used the same preprocessing approach used in the previous steps. Also here we employed a Hyperband tuner to find the best parameters to use for our model, that is then fit and tested.

| **Window size** | **Window shift** | **Score** |
|---|---|---|
| 18 | 1 | **85.35** |
| 18 | 3 | **86.21** |
| 18 | 6 | **84.83** |
| 18 | 9 | **88.20** |
| 24 | 1 | **83.65** |
| 24 | 3 | **87.91** |
| 24 | 6 | **83.71** |
| 24 | 9 | **86.74** |
| 30 | 1 | **82.24** |
| 30 | 3 | **84.69** |
| 30 | 6 | **84.31** |
| 30 | 9 | **89.01** |
| 36 | 1 | **84.06** |
| 36 | 3 | **84.03** |
| 36 | 6 | **84.41** |
| 36 | 9 | **84.37** |
| 42 | 1 | **83.36** |
| 42 | 3 | **85.01** |
| 42 | 6 | **86.55** |
| 42 | 9 | **88.20** |

Table 2: Testing results obtained

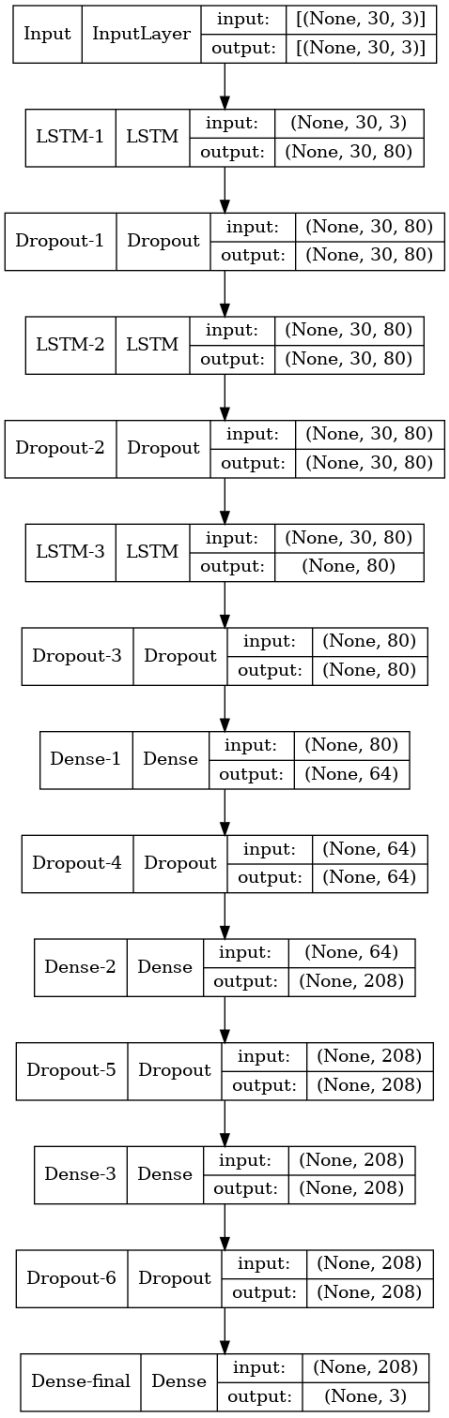


Figure 7: The all in one model

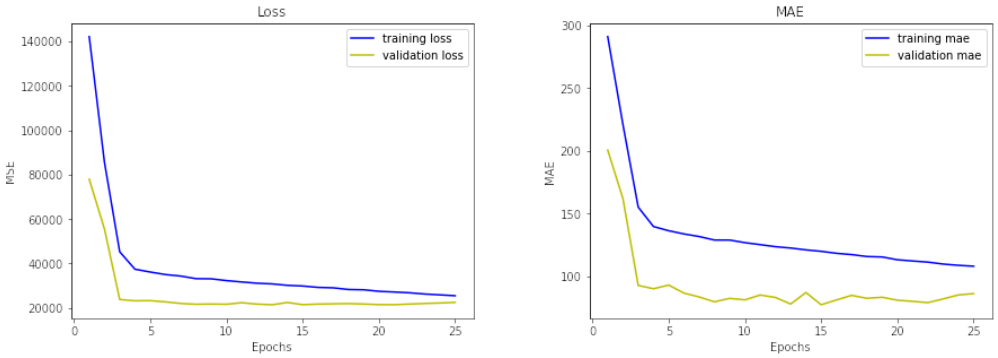


Figure 8: The all in one model loss and MAE

We were not surprised to see (Figure 8) that the plots are similar to the ones observed before. We can also notice that the model is now more easily keeping a not so high variation in evaluation metric during training, whereas before the values could vary wildly. In spite of this, the evaluation of the trained model on the test set has given a result of 93.86, that is higher than the worst result obtained by the same architecture using the three time series separately.

# 3 Task 2: anomaly detection

Given data from wearable devices, the goal is to identify anomalous events, like tremors, in a set of observations, that we refer to as OFF periods. Data are collected by patients with and without Parkinson's Disease. Given the nature of the task, we opted for training a model capable of capturing the trend of the observations of people without Parkinson's, in order to spot deviations from said trend, and thus the presence of OFF periods in the patients with Parkinson's.

## 3.1 Data understanding and preparation

We have two files, `train.csv` and `test.csv`, each one containing seven columns. Features include:

- Identification of patient (`patient`);
- Accelerometer readings in the three axes (`x`, `y` and `z`);
- Heart rate (`heartRate`);
- Date and timestamp (`tsDate` and `timestamp`).

The `test.csv` file was used for the detection of anomalies, while the `train.csv` file was used for the training part. The training set is composed of control patients, i.e. volunteers without Parkinson's Disease; there is a record each 1 second and there are missing values on the heart rate attribute (labeled with `-1`). The test set is composed by patients with Parkinson's Disease and there is a record each 10 seconds. We decided to remove the missing values from the training set (Figure 9 and Figure 10). Given the granularity mismatch between training set and test set, we decided to resample the portion of training set regarding each patient, in order for them to appear as being recorded each 10 seconds, aggregating by using the mean.
Next, we normalized the values in the training set using a standard scaler and, afterwards, we segmented the dataset in sequences of fixed window size/window shift (30/1). Eventually, we split the produced sequences to obtain training and validation sets to use in the training step, using a ratio of 80:20.

## 3.2 Modeling

Since this is an anomaly detection task, we used a Variational Autoencoder (VAE). Autoencoders are neural networks whose goal is, given an input,

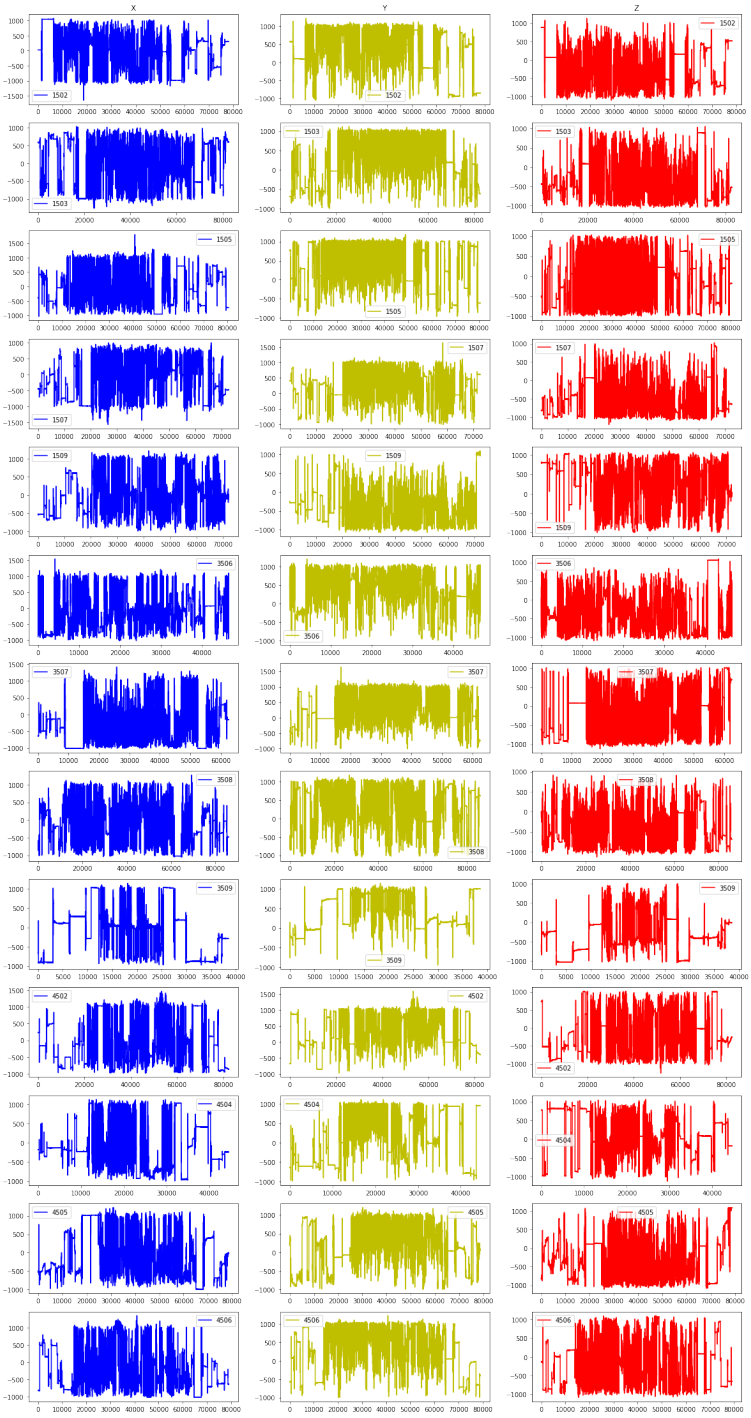


Figure 9: The X, Y and Z spatial coordinates for each patient

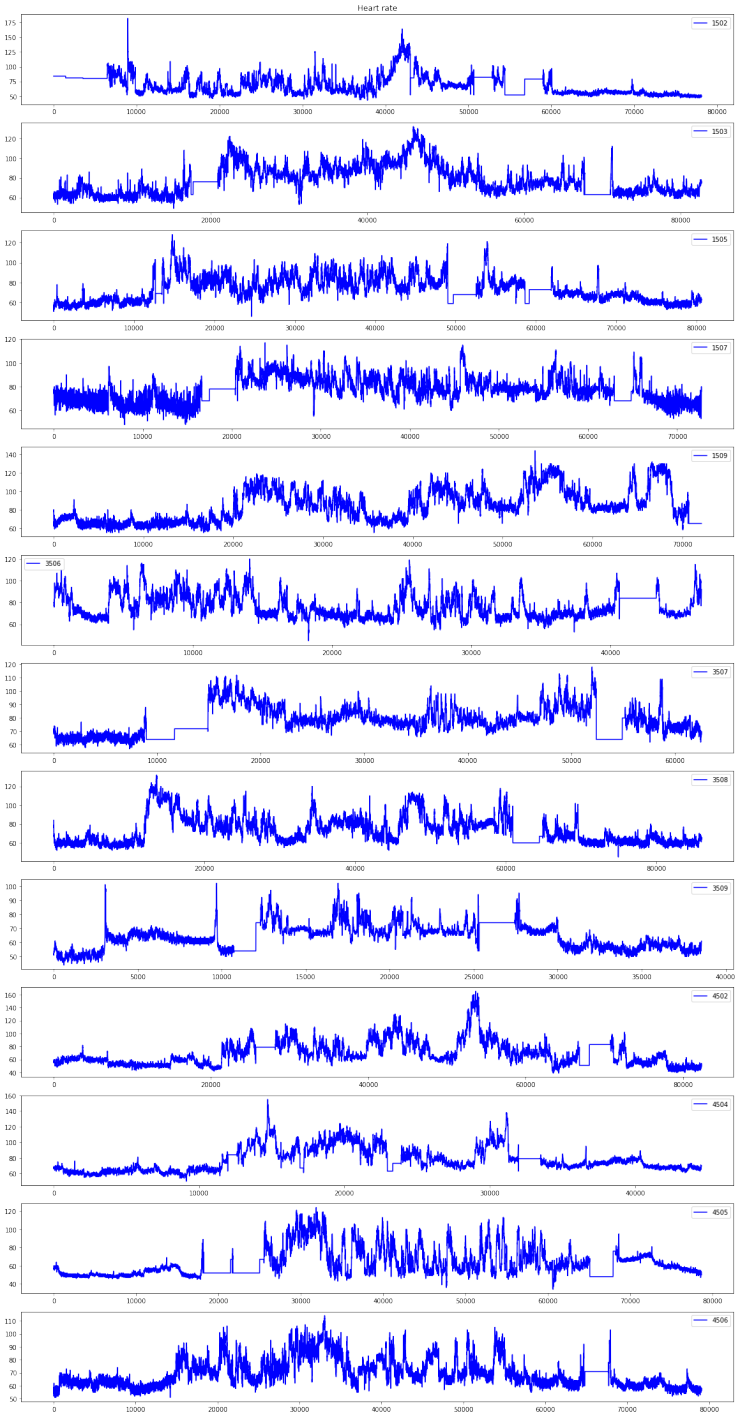


Figure 10: The heart rate for each patient

to reproduce the same input as their output, with the constraint of having to translate the high dimensional information on a lower dimension space, called latent space. This means that the network has to reproduce the input with less information, effectively filtering out irrelevant information. A VAE is a special type of autoencoder that tries to learn the Normal distribution followed by the input data.
The definition of our model goes as follows:

- The encoder, which has to compress the input into the latent dimension, is made of two LSTMs and three Dense layers, in this order. To generate a point in the latent space, we employ two additional Dense layers, one that encodes the mean and the other that encodes the variance of the distribution. Finally, these two layers are linked to a sampler that generates the resulting point according to a Normal distribution;

- The decoder, which has to decompress the point in the latent space back to point in the input space, is made of three Dense and two LSTMs, in this order. Eventually, since the last LSTM has an activation function that constrains the output values in the range [-1, 1], we employed a final Dense layer with a linear activation function;

- The loss function of the network was constructed by summing the Mean Square Error to the Kullback-Leibler Divergence between the inferred distribution and the prior, which is a Normal distribution with mean 0 and the same inferred variance.

After the model definition (Figure 11), we fitted it for 100 epochs using a batch size of 32, keeping track of the MAE of the validation data used during training, in order to save only the best behaving model; this was done using a Model Checkpoint as callback to the fitting process.

## 3.3 Threshold computation

Once the model has been trained, we have chosen a threshold to understand whether a given sequence is an anomaly or not. First, we let the model predict the sequences in the training set, to be used to compute the mean absolute error ($MAE_{seq}$) between them and the legitimate sequences in the training set. The threshold, then, is given by:

$$T = \mu(MAE_{seq}) + \sigma(MAE_{seq})$$

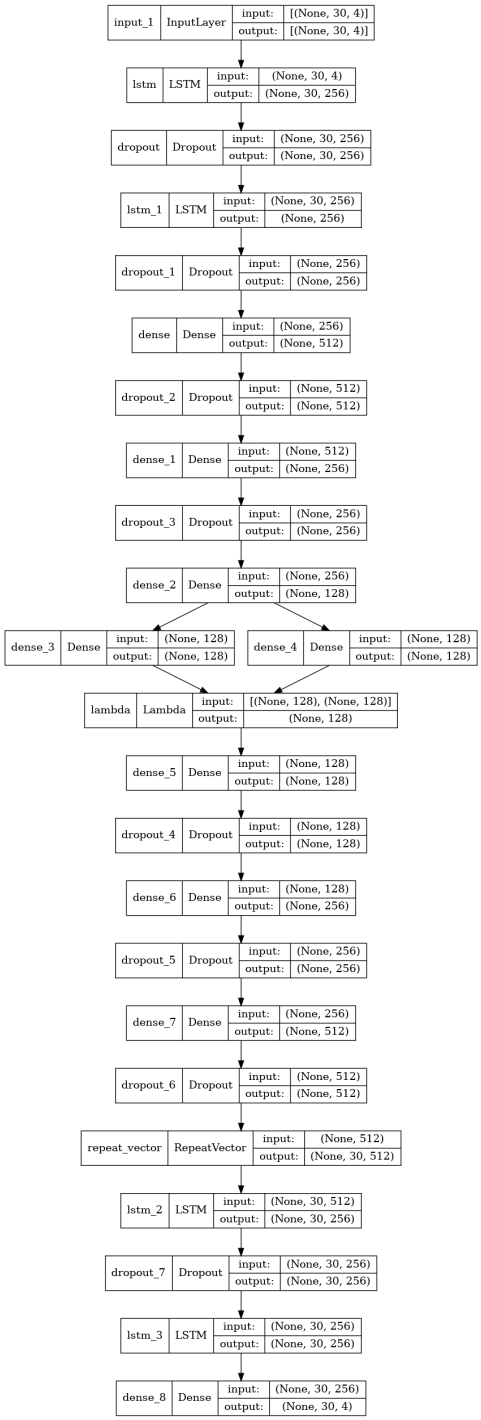


Figure 11: The VAE model

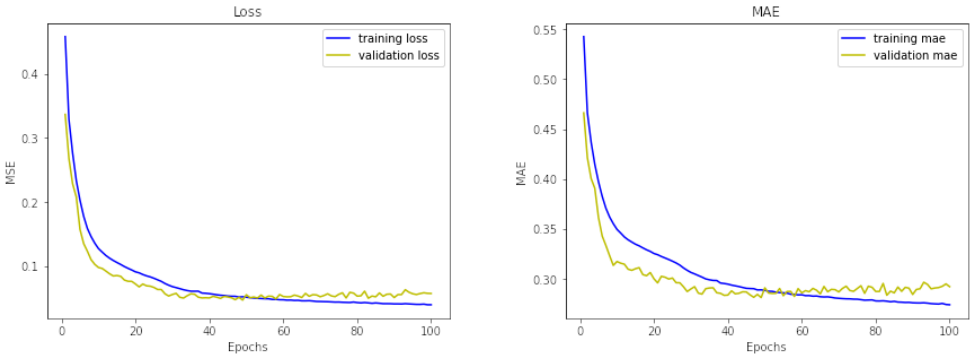


Figure 12: The VAE loss and MAE

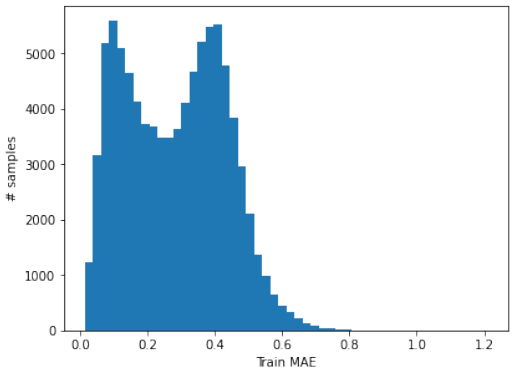


Figure 13: The distribution of MAE train samples

| | | | |
|---|---|---|---|
| **1004** | Observations | 144911 | ∼402 hours |
| | Anomalies | 4136 | ∼11 hours |
| | % of anomalies | ∼3% | |
| **1006** | Observations | 148371 | ∼412 hours |
| | Anomalies | 2171 | ∼6 hours |
| | % of anomalies | ∼1% | |
| **3001** | Observations | 188904 | ∼525 hours |
| | Anomalies | 4421 | ∼12 hours |
| | % of anomalies | ∼2% | |
| **3006** | Observations | 109895 | ∼305 hours |
| | Anomalies | 1249 | ∼3 hours |
| | % of anomalies | ∼1% | |
| **4002** | Observations | 130706 | ∼363 hours |
| | Anomalies | 7366 | ∼20 hours |
| | % of anomalies | ∼6% | |

Table 3: Testing results obtained

## 3.4 Detection of anomalies

Having computed the threshold, we loaded the test dataset and applied the same preprocessing steps applied to the training data. Then, for each patient we let the model predict the sequences of the test set and computed the $MAE_{seq}$ as described before. Using the threshold, we then determine the anomalous sequences in the test set. Once this has been done, we want to extract which are the samples in the test set that correspond to anomalous behavior, checking if the sample is found in `window_size` number of anomalous sequences, in which case we can safely assume that the sample itself is anomalous, discarding all samples that are not found in all of them.

As shown in Table 3, patient 4002 is the one with the highest number of OFF periods, suggesting that for them the medication is less effective, as opposed to patient 3006, which has less anomalies detected and, therefore, the medication is quite effective.

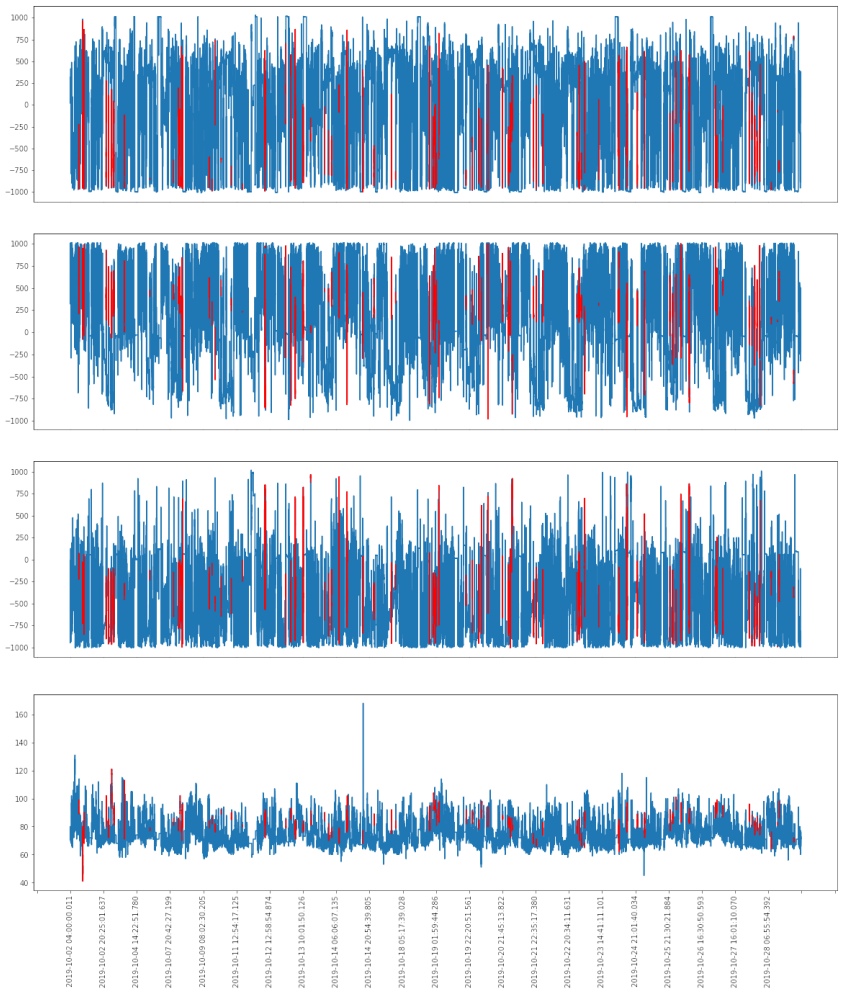

Figure 14: The anomalies for patient 3006

# 4 Conclusions

Regarding the first task, we are satisfied with the results obtained; the evaluation metric for each time series is below the required objective values. Also, changing the window shift from 6 to 1 really helps with bettering the performances of the training stage, yielding better overall results. This, in turn, has led us to using the same combination of window size/window shift for the second task as well. On a different note, the "All-In-One" model, considering all the three time series together, did not give us the results we hoped for.
Regarding the anomaly detection task, we cannot be sure as to how much the model is good at performing its task, given the unsupervised nature of it. In fact, we were not able to compute precision metrics (i.e. F1-score), that would have given some information about the behavior of the model. Nevertheless, looking at the results and the information given to us, we can say that the likelihood of them being realistic is quite high.