Predict Parkinson's Disease using wearable data

Aloi Alfredo - 223949
Riccio Francesco - 223951

Introduction

Introduction

- Parkinson's disease, the second most neurological disorder that causes significant disability, reduces the quality of life and has no cure
- Deep learning has the potential to give valuable information after processing
- The main problem is the identification of **OFF symptoms**
- **OFF periods** are times when Parkinson's disease medication (Levodopa) is not working optimally. As a result, symptoms return
- The goal is to consolidate data collection by identifying the most important variables to solve the problem

Next value prediction

Task 1.1

- We have 3 time series (X, Y, Z) recorded each 10 seconds
- We consider sequences of five minutes every one minute
- The goal is to predict the next value in the series
- The evaluation metric is the Mean Absolute Error

Data understanding and preparation

- We have two files:
  - `train.csv`, used for training
  - `test.csv`, used for evaluation
- Each file contains three columns, representing the three time series X, Y and Z
- The three time series to be used are inside one single dataset
- We started the preparation of our data by first splitting it into the three different time series

Data understanding and preparation (cont'd.)

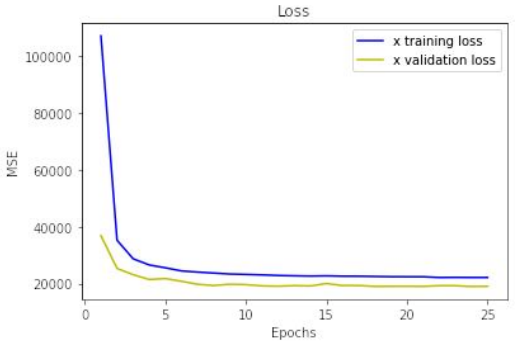
- For each one of the generated time series:
  1. we produced a set of sequences, with the corresponding labels, which represent the element that follows each sequence in the series
  2. we normalized the values using a Standard scaler
  3. we split the produced sequences to obtain training and validation sets to use in the training step, with a ratio of 80:20

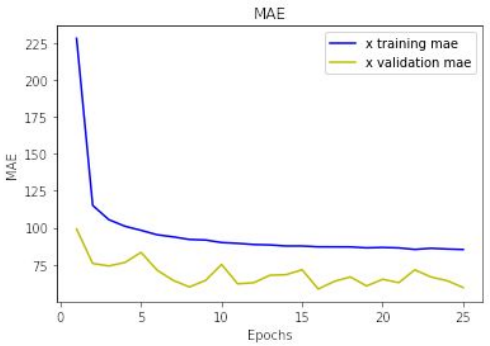
Modeling

- We chose to build the following neural network:
    1. three LSTM layers
    2. three Dense layers
    3. one output Dense layer
- Between each layer of the architecture we put a dropout layer
- We employed a *Hyperband tuner* to make the choice of the best model parameters
- We fit our models for 25 epochs using a batch size of 32
- We used a *Model Checkpoint* to save only the best models

Evaluation

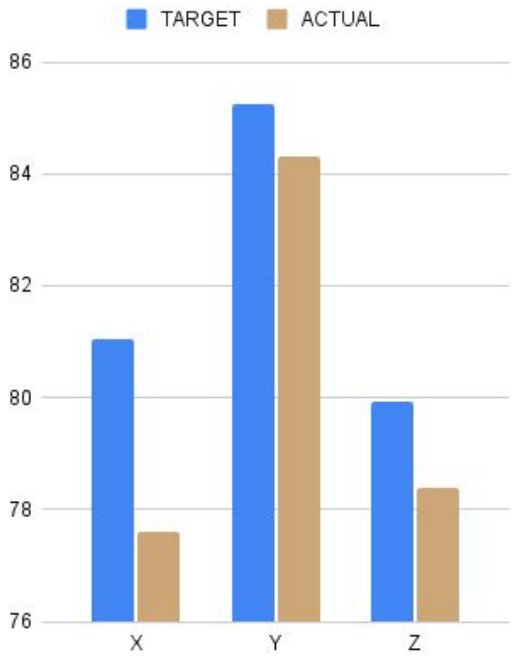
- In the first epochs the model learns very quickly, suffering a sudden slowdown in the following epochs
- The learning curve appears to be quite smooth
- Concerning the Mean Absolute Error, the time series has an irregular behaviour





Evaluation (cont'd.)

- To evaluate our models, we apply the same preprocessing steps also to the test dataset
- The results obtained after testing are quite good
- All the evaluation metrics respect the required objective values



Finding better parameters

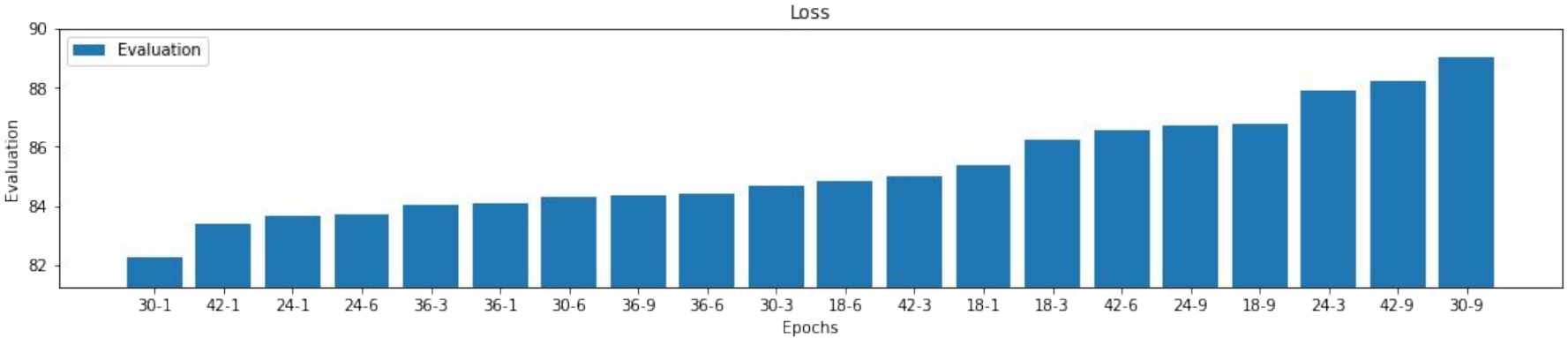
Task 1.2

- We choose the time series with the worst value of the evaluation metric
- We look for a better combination of window size/window shift

Finding better parameters

- We define a search space of:
  - five values for the *window size* (18, 24, 30, 36 and 42)
  - 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

Finding better parameters (cont'd.)



“All in one” model

Task 1.3

- Extra step
- We build a model that takes as input the three time series at once
- Curious to see if we can obtain more accurate results.

“All in one” model

- 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
- Sadly, the evaluation of the trained model on the test set has given a result of **93.86**
- The result is higher than the worst result obtained by the same architecture using the three time series separately

Anomaly detection

Task 2

- 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
- The training set is composed of control patients, i.e. volunteers without Parkinson's Disease
- The test set is composed by patients with Parkinson's Disease

Data understanding

- We have two files:
  - `train.csv`, used for training
  - `test.csv`, used for evaluation
- Each file contains:
  - Identification of patient
  - Accelerometer readings in the three axes
  - Heart rate
  - Date and timestamp

	patient	x	y	z	heartRate	timestamp	tsDate
0	1502	23	569	878	-1	1568073600000	2019-09-10 00:00:00.003
1	1502	23	571	878	-1	1568073601000	2019-09-10 00:00:01.014
2	1502	23	570	878	-1	1568073602000	2019-09-10 00:00:02.025
3	1502	23	570	878	-1	1568073603000	2019-09-10 00:00:03.035
4	1502	23	570	878	-1	1568073604000	2019-09-10 00:00:04.046
...	...	...	...	...	...	...	...
943517	4506	-636	-399	-654	57	1572479994000	2019-10-30 23:59:54.315
943518	4506	-639	-396	-654	57	1572479995000	2019-10-30 23:59:55.316
943519	4506	-638	-396	-655	57	1572479996000	2019-10-30 23:59:56.336
943520	4506	-637	-396	-655	58	1572479997000	2019-10-30 23:59:57.337
943521	4506	-636	-399	-656	58	1572479998000	2019-10-30 23:59:58.338

Data preparation

- Since there are missing values, we remove them
- In the training set there is a record each 1 second
- In the test set there is a record each 10 seconds
- We resample the portion of training set regarding each patient, aggregating by using the mean
- Now the records appear as being recorded each 10 seconds

Data preparation (cont'd.)

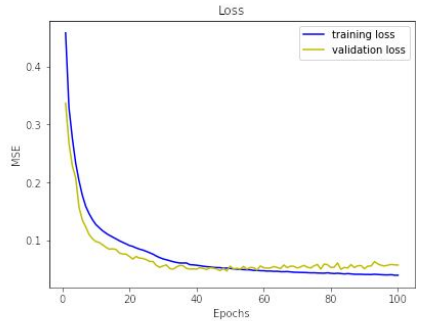
- We normalized the values in the training set using a standard scaler
- We segmented the dataset in sequences of fixed window size/window shift
- We split the produced sequences to obtain training and validation sets to use in the training step, with a ratio of 80:20

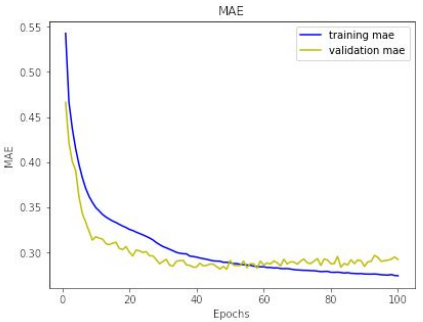
Modeling

- Since this is an anomaly detection task, we used a **Variational Autoencoder** (VAE)
- The definition of our model goes as follows:
  - The encoder is made of two LSTMs and three Dense layers
  - We employ two additional Dense layers, one that encodes the mean and the other that encodes the variance of the distribution
  - These two layers are linked to a sampler that generates the resulting point according to a Normal distribution
  - The decoder is made of three Dense and two LSTMs
  - 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

Modeling (cont'd.)

- After the model definition (Figure 11), we fitted it for 100 epochs using a batch size of 32
- We used a *Model Checkpoint* to save only the best models





Threshold computation

- We have chosen a threshold to understand whether a given sequence is an anomaly or not
- We let the model predict the sequences in the training set
- Those will be used to compute the mean absolute error between them and the legitimate sequences in the training set

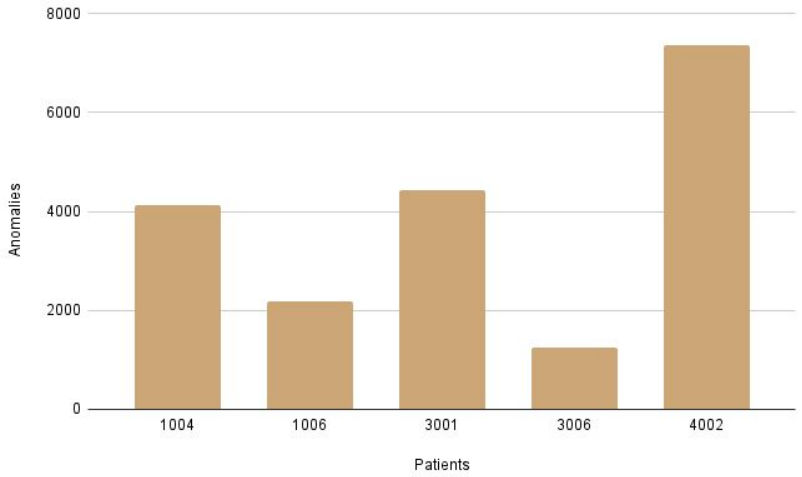
$$T = \mu(MAE_{seq}) + \sigma(MAE_{seq})$$

Detection of anomalous samples

- We loaded the test dataset and applied the same preprocessing steps applied to the training data
- For each patient we let the model predict the sequences of the test set and computed the error
- Using the threshold, we then identify the anomalous sequences in the test set
- We check if the samples in the test set are found in `window_size` number of anomalous sequences, in which case we can safely assume that the sample itself is anomalous
- Those that do not satisfy this condition aren't anomalous

Detection of anomalous samples (cont'd.)

- Patient 4002 is the one with the highest number of OFF periods, suggesting that for them the medication is less effective
- Patient 3006 has less anomalies detected and, therefore, the medication is quite effective



Conclusions

- Regarding the first task, we are satisfied with the results obtained, since the evaluation metric for each time series is below the required objective values
- Changing the window shift from 6 to 1 helps with bettering the performances of the training stage
- The All-In-One model 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
- Nevertheless, we can say that the likelihood of them being realistic is quite high

Thank you for listening!