Assignment 2 - Second Part
ADNE

Mario Gomes, 54667, Analysis and Engineering of Big Data
Simão Gonçalves, 54896, Analysis and Engineering of Big Data

I. Introduction

In this report on the second part of the second assignment of the course *Aprendizagem de dados não estruturados* (ADNE) we aim to compare two different kinds of neural networks with similar number of parameters on the task of identifying QRS complexes on a set of ECG signals based in their respective annotations. This assigment will be composed mainly of 3 sections:

1) **Preprocessing**, where we will explain the steps taken to modify the ECG signal in order to facilitate the learning process of the neural networks;
2) **Model Training**, where we will discuss the models' architecture as well as other quantities of interest, for example, the amount of time used to build the inputs for the different neural networks.
3) **Posprocessing**, final section of this assigment where we will discuss the results of the results of the two neural networks and the process by which the output of the neural nets become annotation files suitable for comparison

II. Preprocessing

In the preprocessing phase we aim at building our inputs and target signal in order to facilitate the learning process of the neural networks. We start by stating that for the construction of the inputs we took two channels of the ECG signals in order to make the learning process more robust and flexible. The order shown below was the one adopted in order to generate the npy files for the training phase.

1) *Resample* of both channels de interest to a sampling frequency of 100Hz along with their annotations file, in order to make the two channels more compressed and easy to manipulate;
2) Application of the Moving Average Technique to time intervals of 1 second by means of the Convolve function of the scipy.signal package in order to normalize the signals;
3) Building of the target signal by putting a parable with fixed length and height one on the positions where the QRS complex was observed. All of the other positions are zero;

Both the two normalized channels and the target sequences are saved in numpy arrays in npy files to posterior use.

III. Model Training

Before actually training both the neural networks we organize the contente of the npy files in 3 3D matrixes, which will be used for 3 different tasks: Train, Validation and Test. After this, we implemented 2 functions to help generate training, validation and testing examples (One tends to randomnly select a ECG file and the second, given the ECG selected takes a random portion of both channels and target with fixed dimensions for both networks.

After this, the two channels are concatenated and turned into a suitable type to feed to the networks.

A. Feedforward Neural Network

For the feedforward model it was used batches with size 8 holding periods of 0.5 seconds. The model is composed by 6 layers where each of the layers is composed by 2000 neurons and has relu activation function. The training process was composed by 90 epochs and is summarized below.

Epoch / Metric	Training set	Validation set
1ª Epoch	0.1356	0.1329
10ª Epoch	0.1315	0.1331
20ª Epoch	0.1286	0.1302
30ª Epoch	0.1261	0.1265
40ª Epoch	0.1256	0.1272
50ª Epoch	0.1249	0.1273
60ª Epoch	0.1243	0.1250
70ª Epoch	0.1241	0.1261
80ª Epoch	0.1241	0.1248
90ª Epoch	0.1231	0.1236

B. Recurrent Neural Network

Contrary to feedforward model the recurrent one aims at capturing time dependencies in data. In this sense, the size 8 batches, contain random portions corresponding to 0.3 seconds of the ECG. The model is composed by 5 LSTM layers with 212 units each and a final Dense that outputs the results. The training process was composed by 135 epochs and is summarized below.

Epoch / Metric	Training set	Validation set
1ª Epoch	0.1351	0.1233
10ª Epoch	0.0682	0.0684
20ª Epoch	0.0512	0.0547
30ª Epoch	0.0435	0.0487
40ª Epoch	0.0383	0.0378
50ª Epoch	0.0516	0.0606
60ª Epoch	0.0314	0.0320
70ª Epoch	0.0272	0.0301
80ª Epoch	0.0252	0.0268
90ª Epoch	0.0245	0.0260
100ª Epoch	0.0232	0.0243
110ª Epoch	0.0224	0.0238
120ª Epoch	0.0219	0.0230
135ª Epoch	0.0205	0.0219

IV. POSPROCESSING

Before discussing the process under which the output signal of both the neural netoworks was subject to in order to retrieve the predicted positions of the QRS complex, we below show the values of the mean absolute error measured on the test set of both models as well as a visualization of the target signal (red color), predicted signal (blue color) and input signal (black color) of a section of the test set.

Metric	Test set
Feedforward Model	0.1356
Recurrent Model	0.0283

We now describe the process of postprocessing. We begin the process by considering a dictionay data structure where the keys are the name of the testing file and the values a list, whose first element is the array containg both signal channels and the second element is the target. We consider two dictionaries, each associated with each model. Subsequently the dictionaries undertake the following sequence of modifications:

1) **Selected Cropping**, where the array is going to be sliced into a series of arrays whose dimension match the dimension of the input for the respected network;
2) **Prediction Generation**, where the output signals for each of the previous created arrays are going to be concatenated in order to reproduce the prediction for the entire ECG file;
3) **Position Generating**, where from the concatenated output signals and with the *find_local_peaks* function from wfdb library are going to be produced an array of the positions where hopefully the QRS complex is;
4) **Annotation Production**, where the annotations are going to be wrote for posterior comparison;
5) **Metric Calculation**, where the annotation associated with the prediction is going to be compared with its reference and the metrics of interest (Positive Predictivity and Specificity) are going to be saved as values;

The metrics associated with the annotations comparison for each test files is summarized in table below.

Records / Metrics	Feedforward Model		Recurrent Model	
	Positive Predictivity	Specificity	Positive Predictivity	Specificity
100	0.0567	0.0396	1.00	0.9934
101	0.5097	0.2402	0.9962	0.9871
102	0.7491	0.6936	0.9771	0.9556
103	0.6141	0.4184	0.9966	0.9928
104	0.3540	0.3324	0.9670	0.9215
105	0.8125	0.6925	0.9866	0.9755
106	0.4865	0.3207	0.9955	0.9827
107	0.6563	0.6051	0.8713	0.8489
108	0.5013	0.5423	0.9862	0.9733
109	0.8893	0.8120	0.9980	0.9921
111	0.8174	0.8432	1.00	0.9939
112	0.0973	0.0866	1.00	0.9941
113	0.5459	0.5499	0.9732	0.9928
114	0.1924	0.0431	0.9973	0.9761
115	0.6381	0.4496	0.9985	0.9923
116	0.8698	0.7475	0.9966	0.9855
117	0.3072	0.0945	0.9993	0.9941
118	0.3084	0.2463	0.9934	0.9860
119	0.5222	0.3256	0.9955	0.9919
121	0.6989	0.7649	0.9995	0.9930
122	0.0385	0.0327	1.00	0.9943
123	0.3431	0.3004	0.9974	0.9928
124	0.1786	0.1248	0.9981	0.9901
200	0.4605	0.2018	0.9788	0.9604
201	0.4027	0.3245	0.9859	0.9633
202	0.5286	0.4022	0.9806	0.9686
203	0.4531	0.1393	0.9449	0.8688
205	0.8353	0.5633	0.9989	0.9861
207	0.5784	0.5516	0.9417	0.9640
208	0.5555	0.2220	0.9844	0.9614
209	0.8686	0.7438	0.9976	0.9887
210	0.5054	0.2668	0.9856	0.9574
212	0.1886	0.1099	0.9993	0.9942
213	0.9383	0.8600	0.9985	0.9917
214	0.5491	0.3117	0.9964	0.9867
215	0.5668	0.1362	0.9922	0.9831
217	0.6045	0.4900	0.9428	0.9108
219	0.729	0.6045	0.9855	0.9791
220	0.7071	0.7202	0.9985	0.9927
221	0.3326	0.1240	0.9888	0.9782
222	0.6806	0.3878	0.9793	0.9315
223	0.8180	0.6814	0.9981	0.9889
228	0.5385	0.4160	0.9887	0.9810
230	0.4659	0.3639	0.9987	0.9929
231	0.1789	0.1025	0.9968	0.9936
232	0.0987	0.0601	0.9949	0.9921
233	0.748	0.632	0.9905	0.9792
234	0.0752	0.0570	0.9996	0.9942