CM2007 - Assignment 3: neural network
ECG time series classification

Rebecca Bonato

March 23, 2023

1 Introduction

Electrocardiography is a powerful source of information to asses the cardiac condition of a patient. Automatic computer analysis of standard 12-lead electrocardiograms is an open field of research with the purpose of improving the correct interpretation of the ECG, facilitating health care decision making and reducing cost [5]. Nowadays, Deep learning algorithms are investigating in order to achieve good and reliable performances. In this report, the process of building and training a model for ECG classification will be described and results obtained will be evaluated in order to highlight strengths and limitations. The python code can be found in the file 'Assignment3-ECG.ipynb'

2 Data Analysis

2.1 Dataset description

Data used for training and validating the model are collected and can be downloaded from *PTB-XL, a large publicly available electrocardiography dataset* [1].
The dataset comprises 21837 clinical 12 lead-ECG of 10 seconds length with an original sampling rate of 500Hz. The down-sampled version (sampling rate = 100Hz) is also provided and it is the one that will be used. An example of signal with its 12 leads is shown in Figure 1. Patients from which data are collected are balanced between male and female (52% and 48%) and their ages cover the whole range between 0 and 95. In addition to the signals, in the "ptbxl database.csv" are also collected identifiers, general metadata, signal metadata and ECG statements: these data are not taken into consideration in the model training/validation. Correspondent labels are stored in "scp statements.csv" and the number of elements associated to each label with the specific label meaning is shown in table 1

#Records	Superclass	Description
9528	NORM	Normal ECG
5486	MI	Myocardial Infarction
5250	STTC	ST/T Change
4907	CD	Conduction Disturbance
2655	HYP	Hypertrophy

Table 1: label types and distribution

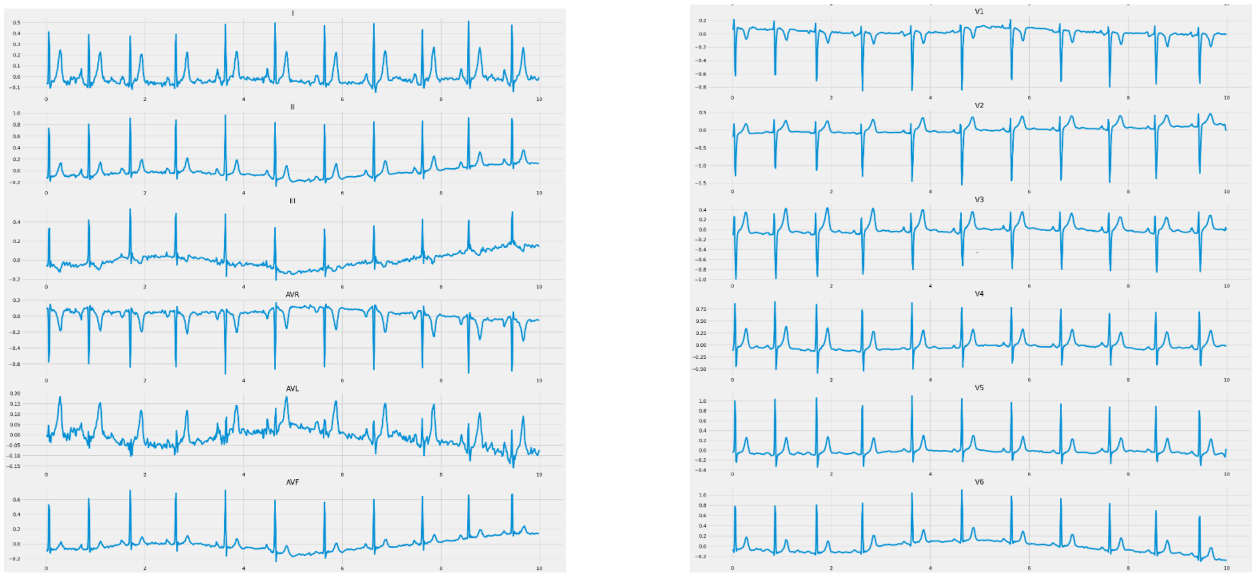
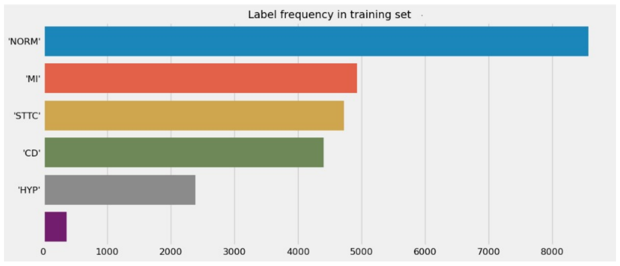
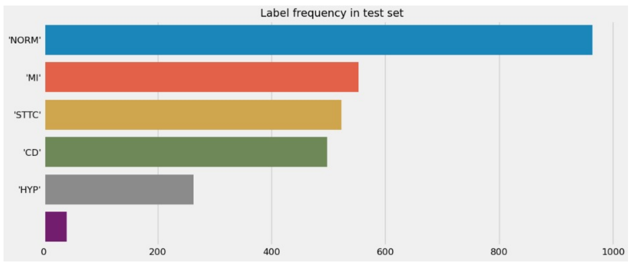


Figure 1: Example of a 12 leads ECG signal



(a) Label distribution in the training set



(b) Label distribution in the validation set

Figure 2: Label frequency in training and validation set

It is important to notice that the sum of the records is greater than the number of signals because each signal can belong to more than one class: it is a multi-label classification problem [3]. In addiction, there is an imbalanced problem: ”hypertropy” is much less represented than the other diseases than in turn are less represented than ”normal ECG”.

2.2 Data preparation

Data are then split in training and validation set according with the information provided in the dataset description: training set is comprised by the 90% of the initial dataset while the validation set by the remaining 10%. The splitting is done in order to maintain proportions between label frequency in training set and validation set as it is shown in Figure 2. It is also possible to notice that there are data that do not belong to any class. Since in the dataset description is not mentioned the reason why these data are not labelled (e.i belonging to other classes, not clear labelling,..) they are removed both from the training and the validation set.
Before providing data to the model each record is normalized with MinMaxScaler technique between 0 and 1 and labels are converted from categorical to binarized vectors through the Hot-Encoding technique as shown in Table 2.

Categorical Classes	**Binarized Classes**
NORM	[0 0 0 1 0]
MI	[0 0 1 0 0]
STTC	[0 0 0 0 1]
CD	[1 0 0 0 0]
HYP	[0 1 0 0 0]

Table 2: In the table it is shown the correspondent binary vector to each categorical label. The Hot-Encoding technique has been used since the labels have no order neither relationship between each other.

3 Model

Different models have been tested in order to obtain good performances: CNN, LSTM/RNN and combinations of both. All of them were characterized by:

1. an input layer with shape equal to [batch size, number of time-steps, number of channels], where number of time-step = 1000 and number of channels = 12

2. as output, a dense layer with five neurons, the same number as the number of possible classes. The last layer´s activation function is sigmoid function in order to get independent probabilities for each class as output of the network.

The model with the best performances has then been selected and it is shown in the Figure 3. It is characterized by two 1D-convolutional blocks with dropout, activation = relu and batch normalization. Each block is then followed by a maxpooling layer with size equal to 2. Two bidirectional LSTM layers are inserted: recurrent dropout is used. Before the last layer mentioned above, two other dense layers are inserted: each of them is followed by dropout, batch normalization and activation function = relu.
Models tested but not used because of lower performances are available at the end of the code in the file 'Assignment3-ECG.ipynb'

4 Metrics for evaluation

The output of the model for each example, is a vector of five elements: each value in the vector represents the predicted probability - between 0 and 1- to belong to the correspondent class. However, the purpose is to obtain a binarized vector to compare with the one created with the hot encoding technique (Table 2): a threshold has to be selected.
The loss function chosen to evaluate if the network is learning properly is binary cross-entropy and it is also plotted the correspondent accuracy in the prediction. At the same time, hamming loss is monitored since it shows the total number of incorrect prediction divided by the total number of prediction considering each label separately as explained in Figure 4.
Recall and precision are also considered because since the label distribution is sparse, It is a good practise to check if the model is likely to predict all zero, or if the error is balanced between false negative and false positive. Finally, macro F1score has been plotted. The rationale behind this, is that there is an imbalanced problem due to distribution of label: this metric assign to all labels the same weight in the evaluation [2].

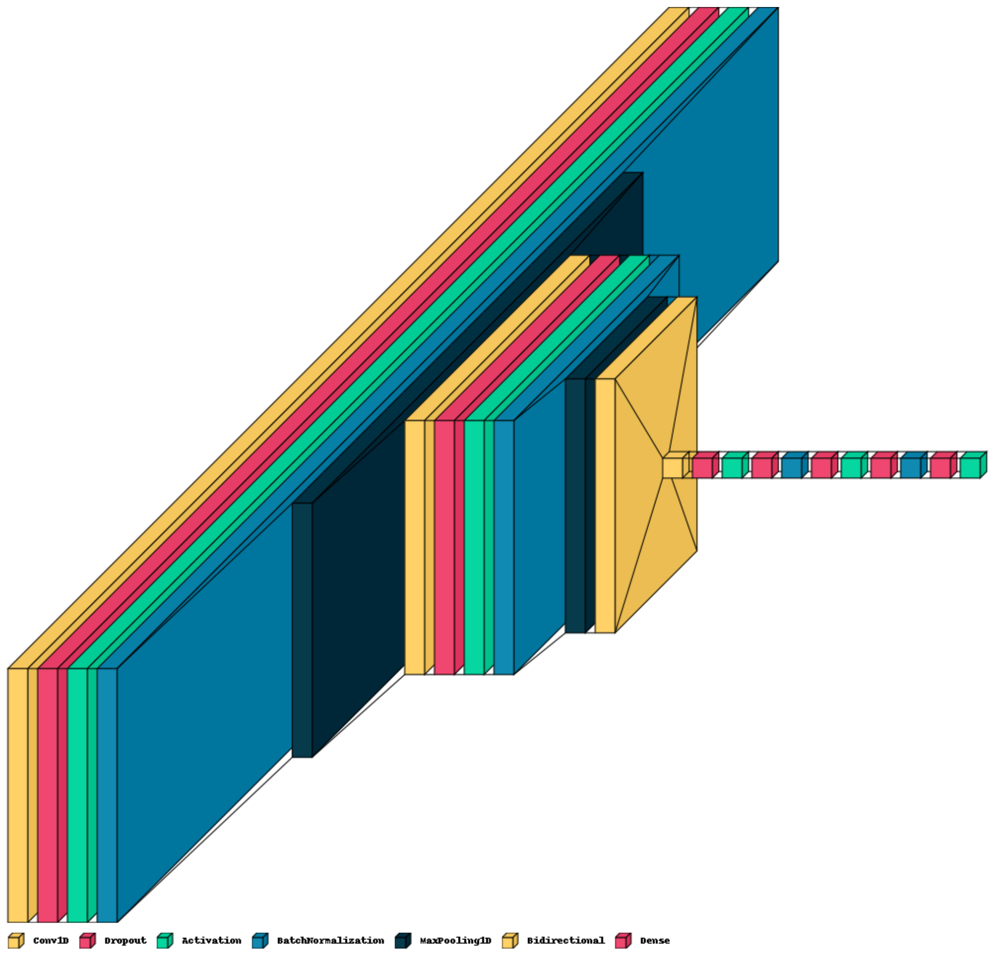
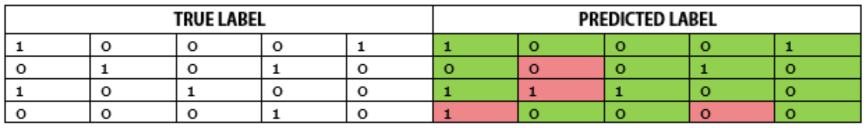


Figure 3: Selected model with the best training trend in terms of Hamming loss, binary crossentropy loss and accuracy [6].

5 Results and discussion

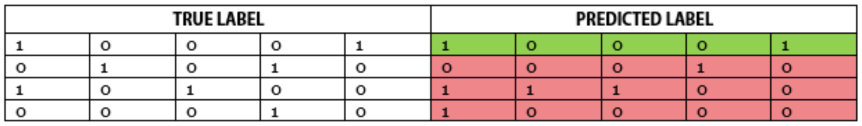
After selecting the model, the parameters have been tuned in order to select the ones with which the network reaches the best performances. These are then summarized in Table 3. The learning curves that represent the learning process of the model during the training are shown in Figure 5. During the best iteration, the network reaches an accuracy of about 69% and an hamming loss of about 12%. The learning curve shows that the model is learning properly since there is convergence but the maximum accuracy score reached is low.

In general, with the different networks I tested, I noticed that the model tends to overfit. The dropout and the recurrent dropout is adopted in the final model to avoid this problem but still the loss function decreases much more on the training set with compared to the validation set. A possible future work to improve the model from this point of view, could be to implement data augmentation in order to generate a bigger amount of data with some variations that can help the model to generalize.
Batch normalization was used as well as regularization technique since it helped in increasing the performances.



TRUE LABEL					PREDICTED LABEL				
1	0	0	0	1	1 🟩	0 🟩	0 🟩	0 🟩	1 🟩
0	1	0	1	0	0 🟩	0 🟥	0 🟩	1 🟩	0 🟩
1	0	1	0	0	1 🟩	1 🟥	1 🟩	0 🟩	0 🟩
0	0	0	1	0	1 🟥	0 🟩	0 🟩	0 🟥	0 🟩

(a) Hamming Loss evaluation in multi-label classification



TRUE LABEL					PREDICTED LABEL				
1	0	0	0	1	1 🟩	0 🟩	0 🟩	0 🟩	1 🟩
0	1	0	1	0	0 🟥	0 🟥	0 🟥	1 🟥	0 🟥
1	0	1	0	0	1 🟥	1 🟥	1 🟥	0 🟥	0 🟥
0	0	0	1	0	1 🟥	0 🟥	0 🟥	0 🟥	0 🟥

(b) Accuracy evaluation in multi-label classification

Figure 4: The difference between accuracy and hamming loss is explained in the figure above. In the accuracy evaluation if one label is predicted wrongly, the example is considered wrongly classified 4b. Instead, using the hamming loss, if only one label is predicted wrongly, the example is considered 20% wrong and 80% right 4a. It is also important to specify that the comparison between them can be done taking 1-hamming loss since the lower is the hamming loss, the better is the model.

The selected learning rate is $1 * e^{-4}$: that makes the network slow at learning on one side, but I noticed that the learning curves are much more stable.
Furthermore, as I said in the dataset description, data are unbalanced since the label frequency varies between different labels. For multi-label classification, a possibility to avoid the class imbalance problem consists in assigning a weight for each training sample. The weight is computed as inversely proportional to the respective class frequency and, if the sample belongs to more classes, the maximum value of the weights of all classes is chosen. The calculated weights are then inserted into the fit function of keras [4]. Unfortunately, this approach failed in increasing performances in my model and I did not use this function in the final training. A weight custom loss function could represent a good way to overcome class imbalance problem in a future work. Moreover, it is a sparse problem: the number of labels that are equal to zero are much more than the ones equal to 1. It would be a good idea also to penalize the predicted list with all classes equal to 0 or in other words with a low probability assigned to all classes.

During training, the network weights (trainable parameters) that allow the network to reach the lower Hamming loss were saved and the prediction on the validation set are calculated through this model. For each example, the independent probability of belonging to different classes is returned: I chose a threshold of 0.5 to binarize the vector since what we want to obtain is a classification more than a probability. Finally, the predicted classes are converted in categorical values.

6 Conclusion

To conclude the model is able to implement a rudimentary classification between different kinds of ECG. However, the performances are still low, maybe due to the complexity of the problem (multi-label and multi-class classification). Consequently a lot of work is still needed to improve the model itself.

Table 3: Chosen parameters

Network Parameters	
Parameter	Value
Optimizer	*Adam*
Learning rate	$1e-4$
batch size	32
filters convolutional	64
dropout convolutional	0.2
batch normalization convolutional	*True*
filters lstm	32
recurrent dropout	0.2
filters dense	128
dropout dense	0.4
batch normalization dense	*True*

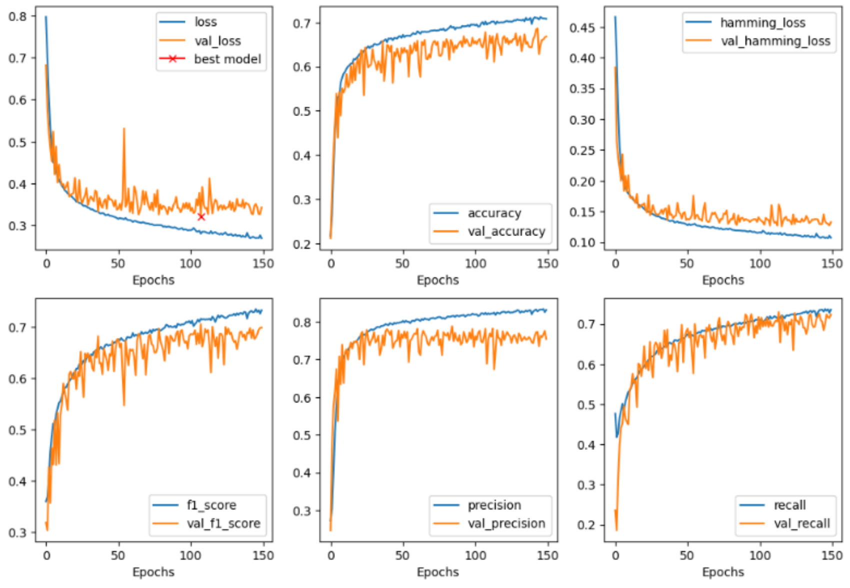


Figure 5: Best trend reached during the training set. The best model is characterized by an accuracy of 69% and hamming loss of 12%. Precision, Recall and Hamming Loss are evaluated setting a threshold of 0.5.

References

[1] Wagner et al. *PTB-XL, a large publicly available electrocardiography dataset*. 2020. DOI: https://doi.org/10.13026/x4td-x982.

[2] *An introduction to MultiLabel classification*. https://www.geeksforgeeks.org/an-introduction-to-multilabel-classification/.

[3] *Multi-Label Image Classification in TensorFlow 2.0.ipynb*. https://github.com/ashrefm/multi-label-soft-f1.

[4] *Sample weights for one-hot encoded multilabel classification with Kerasb*. http://fmnt.info/blog/20201029_sample-weights.html.

[5] Jürg Schläpfer and Hein J. Wellens. "Computer-Interpreted Electrocardiograms: Benefits and Limitations". In: *Journal of the American College of Cardiology* 6 (2017), pp. 1183–1192. DOI: https://doi.org/10.1016/j.jacc.2017.07.723.

[6] *visualkeras for Keras/TensorFlow*. https://github.com/paulgavrikov/visualkeras.