Obdelava biomedicinskih slik in signalov

Aleš Kert

Assignment 2 Report

Ljubljana, 2022

Chapter 1

Report on Assignment 2

1.1 Abstract

In this report we describe the results of our evaluation of different ANNs performance when classifying QRS complexes into either normal (N) or a premature ventricular complex (V). The used methods are a simple multi-layer perceptron(MLP), a long short-term memory(LSTM) network, a simple 1D convolutional network and a modified AlexNet, using 1D instead of 2D convolutions. The methods are evaluated on two different types of splits - one implemented by splitting the files, and one by collecting all the complexes and splitting the shuffled data. The best results on the complexes split were obtained using the MLP model, resulting in $Se = 99.0\%$, $+P = 92.9\%$ and $Sp = 99.9\%$. On the file split we chose the CNN model, resulting in $Se = 93.9\%$, $+P = 46.6\%$ and $Sp = 99.4\%$.

1.2 Introduction

Classification of QRS complexes into either normal or premature ventricular complexes(PVC) is an important problem in cardiology, that is used to diagnose various heart diseases. Frequent PVCs could lead to heart rhythm problems, also called arrhythmias, or to weaking of the heart muscle, called cardiomyopathy. Cardiologists presented with ECG data of a patient could use an old-school approach of simply looking at the ECG records and identifiying PVC complexes manually which would be incredibly time consuming since ECG records usually last dozens of hours if not days. An effective classification algorithm would thus save hours of valuable time and would provide a powerful tool for diagnosing one of the most deadly diseases in the developed world.

In this report we decided to tackle this classification problem using deep learning. For this purpose we designed four architectures based on neural networks. The models were trained and evaluated on the MIT-BIH database. In Chapter 1.3 we describe the data processing and the used algorithms. In Chapter 1.4 we describe our results on the two different data splits. In Chapter 1.5 we interpret the results and provide comments on the performance metrices.

1.3 Methodology

The process of creating our models can be divided into three sections - data preprocessing, where we applied the appropriate transformations on the raw ECG data to extract as much useful data as possible and extracted the sections of the ECG around the already detected peaks, dataset splitting, where we decided the appropriate way of choosing training and test datasets, and lastly model training, where we implemented different neural network architectures.

1.3.1 Data preprocessing

The first thing we did when processing the ECG data was extract the isoelectric baseline using the Butterworth filter. This was done by first applying the filter on the signal going forward, then flipping the filtered signal, applying the same filter on the flipped filtered signal and finally flipping it again to obtain the new filtered signal. This is done to make sure the signal doesn't deviate from the baseline 0, de facto creating different types of QRS complexes based on the amount of isoelectric drift the signal is experiencing at a point in time.

After all the preprocessing is performed, we use the preannotated R peak locations available in the database to cut out a section of the ECG signal containing the QRS complex. The length of a healthy person usually lasts around 100ms, but lengths can also go as high as 120ms. The length of our signal extraction was thus decided to be a bit higher than the maximum length, resulting in a length of 160ms, going from $T_{start} = T - 60ms$ and $T_{end} = T + 100ms$, where T is the time of the R peak. The resulting extract contains 58 samples.

Due to a high class imbalance between normal and PVC heartbeats, we oversample the PVC beats by simply appending 6 identical copies of all the PVC heartbeats to the training set.

Before the data is forwarded to the networks, it is also normalized using the mean and the standard deviation of each signal, using the formula

$$X = \frac{X - mean(X)}{std(X)}. \tag{1.1}$$

1.3.2 Set splitting

For splitting the data into training and testing sets, we decided to employ two approaches - splitting along the files and splitting the shuffled collection of QRS complexes, where the data was first extracted from all the files and then shuffled and split. This was done due to concerns that splitting along the shuffled complexes would cause inadvertent overfitting, since complexes from the same patient would likely resemble each other and the models wouldn't be learning the morphology of complexes in general, but of complexes belonging to a patient.

For the split along the files, a 50/50 split was used, where every other file was chosen, odd for the training set and even for the test set, or vice versa depending on the set we want to evaluate. For the complexes split, we used an 80/20 split.

1.3.3 Architectures

For the neural network architectures we chose a range of different designs, each based on a different concept. All the networks use the LeakyReLU activation function and the Adam optimizer.

Multilevel Perceptron

A multilevel perceptron(MLP) is an architecture that is based on fully connected hidden layers. Our model contains an input layer of size 58, then 3 hidden layer of size 40, 20, 10 respectively and an output layer using softmax activation of size 2. The model was trained for 40 epochs.

Long Short-Term Memory

A network can contain a long short-term memory(LSTM) layer, which is a type of layer used in recurrent neural networks(RNN). Unlike regular feedforward architectures, this neural network also includes feedback connection which allows it to store data about the whole sequence. It is a technique widely used in speech recognition, but since audio and ECG recordings can both be presented as a 1D signal, this method is also applicable in ECG processing. Our model contains an input layer of size 58, followed by an LSTM layer with a 100 hidden layers, a fully-connected layer of size 10 and an output layer of size 2. The model was trained for 30 epochs.

Convolutional Neural Network

When processing signals we can also make use of convolution. A convolutional neural network(CNN) is a neural network that employs a convolutional layer and in practice it is looking for abstract features in a signal. It is usually used on images, where 2D convolution is used, but it can also be a useful tool in signal classification, where 1D convolution is used instead. For our model we used an input layer of size 58, followed by three 1D convolutional layers with kernel size 11 and filter sizes 8, 16, 32 respectively, and ends with an output layer of size 2. The model was trained for 30 epochs.

AlešNet

AlexNet is a CNN architecture that is used for classification of images and employs 2D convolutional layers, combined with max pooling layers. Since 2D convolution is not applicable here, we modified the AlexNet by switching the 2D with 1D convolution and dubbed the modified network AlešNet. The layers have a kernel size of 5. The network contains an input layer of size 58, a convolutional layer with 8 filters, followed by a max pooling layer, a convolutional layer with 16 filters, follower by another max pooling layer, then three convolutional layer with size 32, 32 and 24 and finished with a fully connected layer of size 20 and the output layer. The model was trained for 30 epochs.

1.4 Results

We trained and evaluated the models described above on the MIT-BIH database, using the two different split methods, either using the regular signal or the first derivative of the signal. The metrics used are sensitivity(Se), positive predictivity(+P) and specificity(Sp) where PVC beats are considered as 'positive' cases. The results of the evaluation can be seen in Table 1.1.

Split type	MLP			LSTM			CNN			AlešNet		
	Se	+P	Sp	Se	+P	Sp	Se	+P	Sp	Se	+P	Sp
Whole	99.0	92.9	99.9	99.1	92.0	99.9	97.9	90.7	99.7	98.5	90.1	99.9
File split 1	88.3	60.4	99.0	88.1	40.6	98.9	92.7	56.9	99.3	93.0	42.5	99.3
File split 2	96.8	40.2	99.6	96.5	44.2	99.6	95.2	39.1	99.4	96.3	38.0	99.5
File split 1+2	92.4	48.3	99.2	92.1	42.3	99.2	**93.9**	**46.6**	**99.4**	94.5	40.2	99.4

Table 1.1: Comparison of the model performance based on the chosen data split and input data type. Results in [%].

As can be seen in the table above, different methods of splitting give different results. The most noticeable difference is in the positive predictivity(+P) where an extreme drop in quality happens. After looking at the metrics, we decided to use the CNN model when predicting a given file, since it can be seen as middle ground between the better +P of the MLP model and the better Se of the AlešNet model. This model is the second best in both metrics. Specificity seems to hold up fairly well regardless of split method choice.

1.5 Discussion

As can be seen from the results chapter, different methods of data splitting give us very different results. When splitting along the files, the positive predictivity is substantially lower than when splitting along the shuffled complexes. We think this is because of two factors - first is the fact that normal heartbeats are simply overrepresented in our dataset. The impact of this can be seen in the confusion matrix for the CNN model using the file split 1 method, shown in the Figure 1.1.

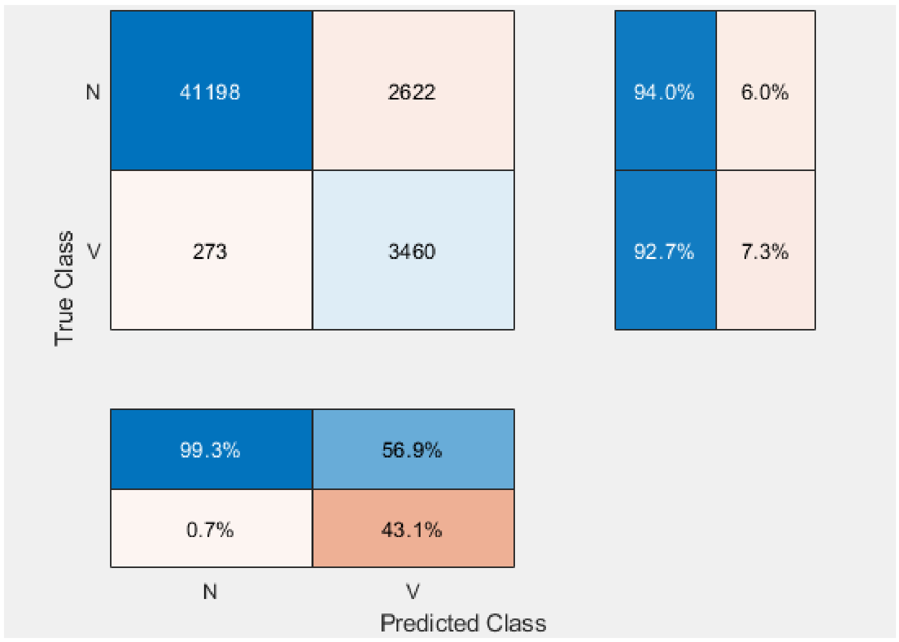

Figure 1.1: Confusion matrix for the CNN model.

We can see in the matrix that our model is still able to correctly identify a normal beat 94% of the time, but because there's more than 12 times as many normal than PVC beats, it makes the +P metric that much worse. This problem could be in part avoided by using a more appropriate loss function, such as focal loss, which is capable of giving better results on imbalanced datasets. This was considered during training, but due to inexperience with deep learning in Matlab, we weren't able to choose a different loss function.

The second reason might have to do with the fact that the MIT-BIH database contains only 48 records from 48 patients. When further splitting this database into two 24 record parts, it further dilutes the already quite low number of records. We think this makes the learning and evaluation process more difficult, since 24 records isn't enough to account for variations in the data recording process, where the electrodes might not be optimally placed, or even just placed differently, resulting in a slightly different ECG recording which a neural network then has problems classifying. Different patients might also have slightly different QRS complexes. This could be mitigated by using a more expansive database such as LTST or by using data augmentation on the existing dataset to account for these variations.

It would also be interesting to see if other methods, that have been proven to be successful in audio classification problems(like music genre classification), are also applicable in QRS complex classification. A method worth looking into is using the short-term Fourier transform(STFT) of the signal and treat it as an image, which was able to achieve 92% accuracy in a 10-class classification problem and could prove to be even better in our binary classification.