Can Electrocardiogram Classification be Applied to Phonocardiogram Data? – An Analysis Using Recurrent Neural Networks

Christopher Schölzel, Andreas Dominik

THM University of Applied Sciences, KITE Kompetenzzentrum für Informationstechnologie
Giessen, Germany

Abstract

Both a Phonocardiogram (PCG) and an Electrocardiogram (ECG) are sequential measurements of heart activity used to distinguish normal from abnormal heart function. Although they measure different physical quantities, we show that training a long short-term memory network on the Physionet challenge using only the ECG data available for the MIT heart sounds database still yields a score of 0.74 compared to the reference score of 0.82 for a similar net trained on the PCG data.

This finding suggests that it may be valuable to train a transformational neural network to produce an artificial ECG from a PCG. Such a transformational net would allow to harness the know-how of decades of research on ECG classification to improve PCG classification. Unfortunately, this task seems too hard for current state-of-the art architectures for neural networks given the data of the Physionet challenge 2016. However, it may be worthwhile to further pursue this approach using data with less variance in the ECG signals or a specialized network architecture.

1. Introduction

A search for the term "electrocardiography" on Pubmed [1] currently lists 190,867 entries, while a search for the term "phonocardiography" returns only 7,758 items.

In essence, both an electrocardiogram (ECG) and a phonocardiogram (PCG) are temporal sequences of a continuous measure that are used to diagnose heart diseases. From a data science perspective, this makes them similar enough to wonder whether we may somehow use the results of the orders of magnitude more numerous research on ECGs to advance the current state of PCG classification.

We could easily achieve this if we were able to transform a PCG signal into an artificial ECG. In machine learning this task falls under the category of sequence-to-sequence learning. There already exist neural networks in this domain that are able to perform natural language translation [2] and speech recognition [3]. Both approaches only have a discrete sequence of events (words or syllables) as target sequence, but in principle there is nothing that stops us from producing a continuous sequence as output. Autoencoder networks, for example, encode an input sequence into a smaller fixed-size representation and decode the entire input sequence from this representation.

Still, the idea of transforming a PCG into an ECG arguably is a more difficult problem than transforming a text in one language to a text in another language. In the latter case, both the input and the output have the same structure (sequence of words), whereas in our scenario we would have to transform sound data into an ECG reading. Both are essential waveforms, but they differ considerably in terms of the shape of the waves and their dominant frequencies. Additionally, of course both signals do not carry identical information. Not every polarization or depolarization visible in the ECG results in a respective sound. Conversely, the murmurs and additional heart sounds visible in a PCG do not have a specific representation in the ECG.

The first question we have to ask is therefore: Can an ECG representation carry the information that is required for PCG interpretation? The 2016 Physionet challenge [4] provides an excellent basis to answer this question. The first part of the training set, the MIT heart sounds database (MITHSDB), contains both ECG and PCG signals. As the task was to distinguish between normal and abnormal PCG recordings, we can train one classifier on the PCG data as intended and another classifier on the ECG data to find out how much useful information the actual ECG contains. This should be a good estimate of the maximum amount of information that can be gained from an artificial ECG.

2. Methods

2.1. Classification

For the classification of ECGs and PCGs, we use long short-term memory networks (LSTMs). LSTMs are recur-

 ISSN: 2325-887X DOI:10.22489/CinC.2016.167-215

rent neural networks (RNNs) that are capable of detecting long-term dependencies in the input data [5]. Since we only wanted to have a competitive classifier to test our hypothesis, we used a rather simple network architecture. The single input neuron containing the ECG or PCG signal is connected to a hidden layer of 100 LSTM units which is again connected to an output layer with only one neuron and a linear activation function. The LSTM layer used a sigmoid activation for the inner cells and tanh activation for the LSTM unit output. Our objective function was the squared error of the activation of the output neuron after the whole input sequence has been fed through the network. In accordance with the definition of the Physionet challenge, the output should be -1 for a normal sample and 1 for an abnormal sample. The implementation of the network was done in Python using the Keras framework [6]. In the following, we will call the ECG classification network LSTMe and the PCG classification network LSTMp.

As training data we used all available challenge databases for LSTMp. Since the total count of normal samples was much higher than the count of abnormal samples, we used all abnormal samples, but only a random selection of normal samples of the same size as the abnormal samples (in total n = 1262). For LSTMe we could only use the data of the MITHSDB, because the other databases did not contain ECG signals. We performed the same random sampling on MITHSDB to achieve an uniform class distribution. We did not apply any denoising or normalization for both nets, because the score achieved on this data was already sufficiently high. The dataset was split randomly into a training (n = 1136) and a test dataset (n = 126).

The training algorithm was the same for both nets. We used RMSProp [7] with a learning rate of 0.001, $\rho = 0.9$ and $\epsilon = 10^{-8}$. The loss function used was mean squared error.

2.2. Transformation

For the transformational neural network (TNN), we could not use the network layout proposed for sequence-to-sequence learning by Sutskever et al. [2] or Graves et al. [3], because these networks are trained on discrete symbols while we have a continuous numeric sequence at both ends of the training. We therefore tried several different approaches, some similar to these proposed structures. Unless otherwise stated the training algorithm used is Adam [8] with learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and mean squared error as objective function. For LSTM layers the default choice for the activation function was tanh activation for the whole LSTM unit and sigmoid activation for the inner cells. The approaches are the following:

1. A simple LSTM with 100 hidden neurons, linear activation function and an (also linear) output layer.
2. A bidirectional LSTM (BLSTM) consisting of two 100 unit LSTM layers, where one layer receives the normal input and one layer the reversed sequence. The output of both layers is concatenated and then connected to an output layer with a single neuron with linear activation.
3. Same structure as 2, but with four layers of 50 LSTM units in both paths instead of the single LSTM layer. The training algorithm for this net was RMSProp with the same Parameters as for LSTMe and LSTMp.
4. Evolino network [9] with a single 100 unit LSTM layer with linear activation, which is trained by an evolutionary algorithm, and an linear output layer, whose weights are determined by least squares optimization. For the evolutionary algorithm we used a mutation rate (factor applied to Cauchy noise) of 0.01.
5. Autoencoder network with three fully interconnected feed-forward layers with linear activation. The first and last layers have 6000 neurons. The network is presented with a window of the PCG or ECG of length 6000 and is then asked to reproduce the input exactly on the output layer. When the middle layer is chosen smaller (we experimented with sizes varying from 100 to 3000), the network has to produce a compressed version of the input signal, discarding smaller variations. If one has an PCG autoencoder and a ECG autoencoder, it is possible to train a TNN that transforms the output of the encoding part of the ECG network into the input for the decoding part of the PCG autoencoder. Due to the compression, this may be an easier task than directly transforming the sequences. Here, we used stochastic gradient descent as training algorithm with a learning rate of 0.01.
6. Temporal autoencoder network using a five layer LSTM of sizes $25 \times 15 \times 5 \times 25 \times 25$ and an output layer with linear activation. The network uses 50% overlapping windows of size 200 from the original sequences as input data.
7. BLSTM as in 2, which does not directly translate PCG signals to ECG signals but instead uses the wavelet coefficients of both signals. The coefficients were obtained by a discrete wavelet transform with the sym12 wavelet, which is reported to give good results for ECG denoising [10].

The transformational networks also used a more rigorous preprocessing. We removed baseline wander from the ECG signal by a highpass filter using a smooth cutoff of the fourier coefficients of the signal at 0.3 Hz. Additionally, we applied wavelet denoising with the sym12 wavelet and the NeighBlock [11] method to both the PCG and ECG signals. For the ECGs the estimated standard deviation of white noise was 0.08, for the PCGs it was 200. We noticed that there are large qualitative differences in the shape of the ECGs due to different placement of the electrodes. We therefore only selected a small subset of 134 of the MITHSDB samples for training the transformational networks, which were considered to have a "typical" shape of QRS-complex, P and T wave. As with the classification

net	dataset	sensitivity	specificity	score
LSTMe	test	0.727	0.750	0.739
LSTMe	train	0.613	0.790	0.702
LSTMp	test	0.778	0.857	0.817
LSTMp	train	0.822	0.824	0.823
LSTMp	official	0.682	0.815	0.749

Table 1. Results of ECG classification (LSTMe) and PCG classification (LSTMp) using LSTMs. The Physionet challenge scores provided for the *test* and *train* dataset were computed locally. Only the result marked as *official* was obtained from an official entry to the challenge.

nets we split our dataset into a training (n = 121) and test (n = 13) set. We also experimented with a strategy where we split each sample into non-overlapping windows of five seconds length, yielding larger sample numbers (853 total, 768 train, 85 test) and a slightly easier task for the net. In both setups, all samples were additionally normalized to a median of zero and a standard deviation of one.

3. Results

The results of the classification networks can be seen in Table 1. The score of LSTMe was roughly 0.1 units lower than for LSTMp. Generally all networks tend to have a higher specificity than sensitivity, with the difference being more pronounced for LSTMe.

Unfortunately, for the TNN, none of our approaches produced anything that would resemble an artificial ECG. The autoencoder-approaches both failed to reproduce the original signal, whether they were trained with ECG or PCG data. Most of the other approaches simply yielded a seemingly random jitter around the mean value of the target ECG. The only notable exception were approaches 2 and 3 involving BLSTMs. As can be seen in Figure 1, these nets seemed to react to the primary heart sounds S1 and S2 by producing first a sharp drop and then a small increase in the output resembling roughly the S and T waves. There is, however, no indication that they are able to distinguish S1 and S2 or to anticipate the next heart cycle, which would be required to also follow the R wave.

4. Discussion

The official score of 0.749 for LSTMp gives good evidence that LSTMs are a suitable classifier choice for this task, especially since we only used a very simple network architecture without any data preparation. Comparing the locally computed scores of LSTMp and LSTMe also suggests that our transformational approach may be viable. We loose around 0.1 in score, which roughly translates to a 10% loss in sensitivity and specificity, but we are still able

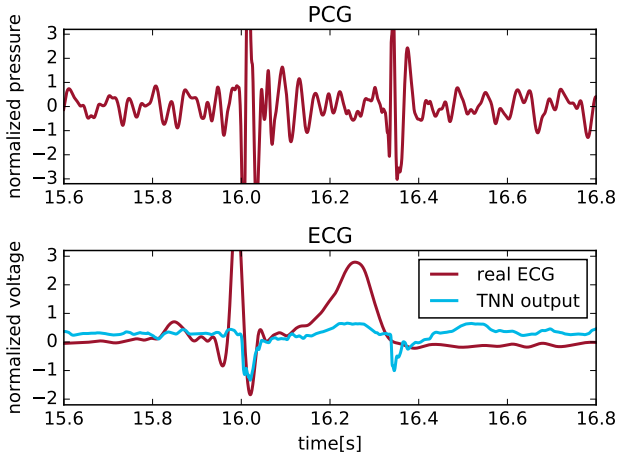

Figure 1. Output of TNN with architecture no. 3 trained on non-overlapping five second windows for sample a0345 (part of the test set).

to classify 74% of the data of the MITHSDB using only ECG information. This is especially interesting since the Physionet challenge was designed to be solved with PCG information.

When we investigated how LSTMe decides between normal and abnormal cases, we found a surprising correlation: Most of the time, the network just outputs the sign of the input and indeed this "feature" is enough to achieve a score of 0.63 on the MITHSDB. These results therefore have to be taken with a grain of salt.

The fact that our TNN approach did not produce any meaningful results shows that the transformation of a PCG to an artificial ECG is an extremely hard task for a neural network. This can be illustrated by a small thought example, where we ask a physician to perform the same task. Given only the PCG, the physician would probably draw an idealized shape of QRS-complex, P and T wave at the positions indicated by S1 and S2 in the PCG. Although retaining the essential information about the heart rhythm, the produced ECG may actually score very poorly when it is compared with the actual ECG in terms of mean squared error. The physician (and our TNN) would never be able to predict the precise placement of the electrodes, which can alter the signal substantially as illustrated in Figure 2.

Another problem that our TNNs had to face was the variance and noise in the PCG data. Unfortunately, the MITHSDB was recorded using 11 different transducer sites across samples, which introduces significant variation in the shape and quality of the input data. In principle, neural networks are able to learn to ignore these variations and concentrate on the relevant information in the input signal. However, since our target signal also had very strong variations, the error that was fed back into the network during

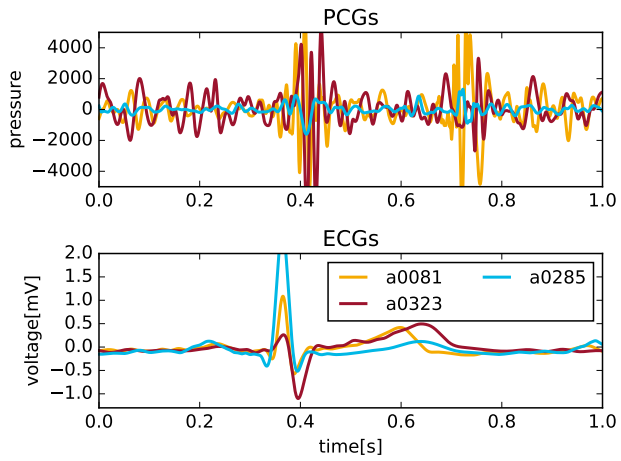

Figure 2. Variations among the hand-selected normal samples with "typical" ECG shape manually aligned at the R peak. It can be seen that from the PCG it is not predictable how large the Q, R, S and T waves are in the ECG.

training may simply have been too irregular to guide the network to filter out these input variances.

It may still be the case, that we did not find the right settings for training parameters or a fitting network structure. However, the results of our best TNN shown in Figure 1 suggest that a BLSTM can at least learn the simple and strong correlations between S1 and the S wave and S2 and the T wave. With more regular data, that exhibits clearer correlations between other parts of the signals, our transformational approach may therefore still be viable.

5. Conclusion

What can be learned from our approach? First of all, ECG data may be more interesting for diseases that are typically detected using a PCG than one would expect at first glance. It may be interesting to use more sophisticated classifiers and other datasets to validate or falsify our findings with respect to specific diseases. At the same time, one should investigate the classifier looking specifically into the features that it learns and whether they are only rhythm-related or really reflecting more subtle changes in the heart function.

A transformation from PCG to an artificial ECG seems infeasible with the given data. To have a realistic chance at this task, we have to use ECG data that either have a very exact electrode placement or that is presented in a form that is invariant to electrode placement, such as 3D vectors obtained by 12-lead ECGs. In the same line, one should also work with PCG data that only features a single transducer placement setup.

Even if we have a net that can do the transformation, we still have to find ECG-related methods that can be of use in this area. One interesting proof of concept may be to just create a very simple artificial ECG using a template shape that is only translated in time according to the location of the primary heart sounds S1 and S2. With this approach, we only retain rhythmic information, but we do not need a complicated TNN. Such an artificial ECG may already be enough to apply some rhythm-related analysis of PCG data using methods developed for ECG without the need to adapt the method itself.

References

[1] NCBI. Pubmed, 2016. URL `http://www.ncbi.nlm.nih.gov/pubmed`. (visited 24.08.2016).

[2] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In Advances in neural information processing systems 27. Montréal, Canada, 2014; 3104–3112.

[3] Graves A, Fernández S, Gomez F, Schmidhuber J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning. Pittsburgh, Pennsylvania, 2006; 369–376.

[4] Liu C, Springer D, Li Q, Moody B, Juan RA, Chorro FJ, Castells F, Roig JM, Silva I, Johnson AEW, Syed Z, Schmidt SE, Papadaniil CD, Hadjileontiadis L, Naseri H, Moukadem A, Dieterlen A, Brandt C, Tang H, Samieinasab M, Samieinasab MR, Sameni R, Mark RG, Clifford GD. An open access database for the evaluation of heart sound algorithms. Physiological Measurement 2016;37(9).

[5] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation 1997;9(8):1735–1780.

[6] Chollet F. Keras, 2015. URL `https://github.com/fchollet/keras`. (visited 31.08.2016).

[7] Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.

[8] Kingma D, Ba J. Adam: A method for stochastic optimization. ArXiv e-prints arXiv:1412.6980 [cs.LG], 2014.

[9] Schmidhuber J, Wierstra D, Gagliolo M, Gomez F. Training recurrent networks by evolino. Neural Computation 2007; 19(3):757–779.

[10] AlMahamdy M, Riley HB. Performance study of different denoising methods for ECG signals. Procedia Computer Science 2014;37:325–332.

[11] Cai TT, Silverman BW. Incorporating Information on Neighbouring Coefficients into Wavelet Estimation. Sankhyā: The Indian Journal of Statistics, Series B (1960–2002) 2001;63(2):127–148.

Address for correspondence:

Christopher Schölzel
Technische Hochschule Mittelhessen
Wiesenstraße 14, 35390 Gießen, Germany
christopher.schoelzel@mni.thm.de