Using Deep Gated RNN with a Convolutional Front End for End-to-End Classification of Heart Sound

Christian Thomae[1], Andreas Dominik[1]

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

Abstract

Classification of heart sounds of a diverse set of phonocardiograms (PCGs) from different recording settings is the challenging objective of the 2016 PhysioNet Challenge. We suggest an end-to-end deep neural network, which is fed with raw PCGs and which learns to autonomously extract features and to classify the recordings. Our architecture combines convolutional and recurrent layers, followed by an attention mechanism, which weights time steps by importance and a dense multilayer perceptron as classifier.

Whereas currently trending deep neural networks in speech recognition or computer vision use up to a million of training samples, a restricted set of only 3,153 heart sound recordings is available as training data. We workaround this limitation by artificially increasing the training set by means of augmentation of the raw PCGs using various audio effects.

Using this moderately sized neural network, we attain high validation scores of 0.89 on validation data; however the resulting scores on the hidden test data of the challenge diverge in range (0.82).

1. Introduction

The objective of the 2016 Physionet Challenge is the classification of heart sound recordings as normal or abnormal. As heart sounds are a reliable indicator for the health of the heart, automated algorithms for heart sound analysis are widely studied.

Artificial neural networks have been shown to be powerful classifiers when applied on the wavelet transform of heart sounds. [1]

Inspired by the recent success of deep learning, we train neural networks in an end-to-end fashion, from the raw waveform signal to the classification result.

2. Algorithm

In order to achieve end-to-end classification the neural network has to learn three tasks: extract meaningful features from the raw waveform, encode the time-dependent features into a single vector that describes a recording and finally classify as normal/abnormal (see figure 1).

In our neural network, the feature extraction is performed by a 1-dimensional convolutional front end, which learns to extract frequency and waveform features. Subsequent recurrent layers learn sophisticated long-term dependencies hidden in the extracted time-frequency features of the heartbeats.

A feed forward attention mechanism is used to aggregate the distributed information in the output of the last recurrent layer into a single decision. Output of each interval is weighted by importance and a weighted average is computed as a fixed representation of the whole recording. This last step can be interpreted as embedding each processed recording into a fixed vector space. Using this attention-weighted average, a multilayer perceptron classifies each recording as normal/abnormal.

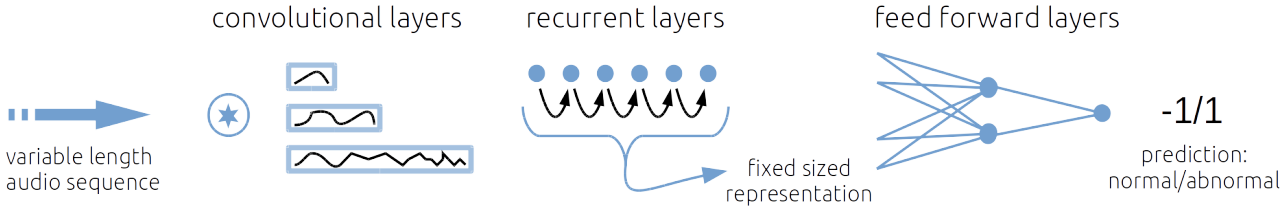

Figure 1. Simplified schematic used for end-to-end classification.

Computing in Cardiology 2016; VOL 43 ISSN: 2325-887X DOI:10.22489/CinC.2016.183-214

2.1. Architecture

In the following we describe the architecture of the neural networks in more detail. We optimised the architecture in multiple steps. The final structure of the deep neural network is displayed in table 1.

Table 1. Layers of the neural network including layer-specific parameters.

#	Layer Type	Parameters
1	Gaussian Noise	1e-6
2	Batch Normalisation	-
3	Convolution 1D	16x16,relu
4	Convolution 1D	16x32,relu
5	Convolution 1D	16x64,relu
6	Convolution 1D	16x128,relu
7	Max Pooling	l=256,s=128
8	Batch Normalisation	-
9	Bidirectional GRU	16+16
10	Batch Normalisation	-
11	Bidirectional GRU	16+16
12	Batch Normalisation	-
13-15	Feed Forward Attention MLP	64-8-1,linear
16	Temporal Softmax	
17	Weighted Average	
18-21	Dense MLP w/ Dropout	64-8-1,p=0.5

Raw waveform data is fed as input into a batch normalisation layer [2], which standardises the input data using batch statistics. More batch normalisation layers are used after the convolutional layers and after each recurrent layer in order to accelerate training.

The normalised input is symmetrically zero padded and sent to four 1-dimensional convolutional layers with 16 filter maps each. The filter maps have varying lengths (16, 32, 64, 128) and use a ReLU activation. When fully trained, the convolutional layers act as frequency and waveform matcher. While the filters could be compared to wavelets, they don't resemble commonly used wavelets

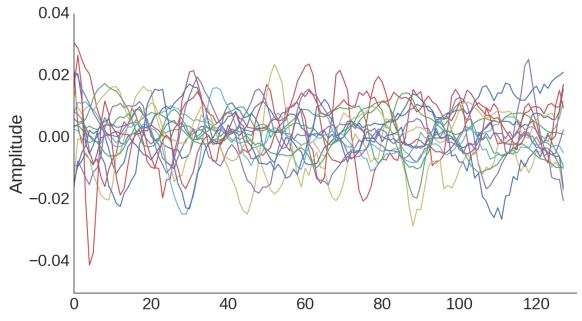

Figure 2. Visualisation of filters; coloured by filter.

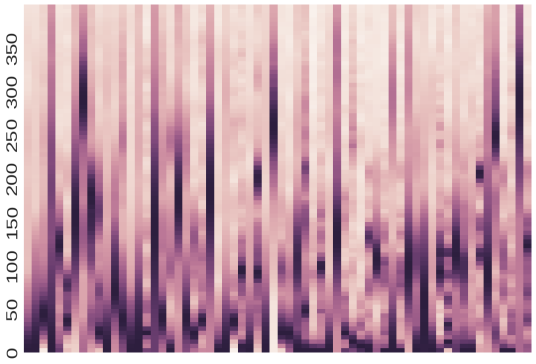

Figure 3. Visualisation of frequency response of each filter; from low response (white) to high response (dark violet).

(see figure 2). Instead, they respond to frequency patterns which are present in the raw recordings (see figure 3).

The filter responses are concatenated and a max pooling layer is used to remove redundancy. The max pooling layer aggregates intervals of length 256 (128 ms) and stride 128 (64 ms) by selecting the maximum value in each interval. The result of this first part of the feature extraction are so-called pooled normalised feature maps.

The following main part of feature extraction is performed by two layers of bidirectional gated recurrent units (GRU) [3]. GRUs have been shown to be comparable in performance to long short-term memory (LSTM), but simpler [4]. Compared to ordinary recurrent neural networks, these units use a gating mechanism to update and reset the hidden state, enabling them to learn long-term dependencies. We use bidirectional GRU layers with 16+16 units. The activations of both recurrent layers are visualised in figure 4.

A feed forward attention mechanism, proposed by [5], is used to aggregate all relevant information of a recording into a descriptive vector: A small dense feed forward network (64-8-1), followed by a softmax over time, is applied

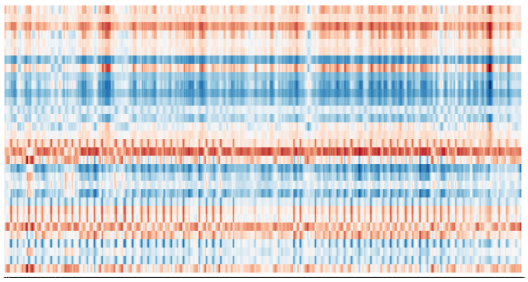

Figure 4. Visualisation of recurrent activations for recording c0014; top: first GRU layer, bottom: second GRU layer; from negative act. (blue) to positive act. (red).

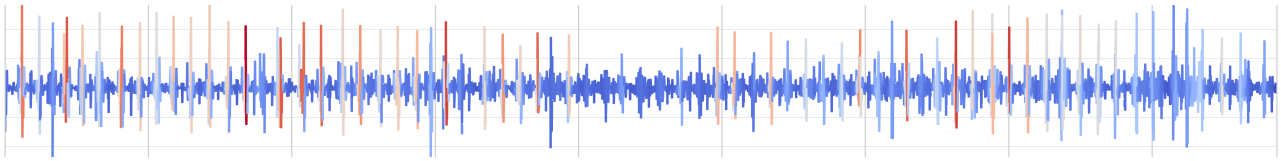

Figure 5. Visualisation of attention mechanism for recording c0014; colour corresponds to importance/attention from low (blue) to high (red).

to the output of the last recurrent layer. Using the output of the attention network as weights, a weighted average of all intervals is calculated (see figure 6). Figure 5 shows how the network attends to different parts of a recording, especially heartbeats.

We use dropout [6] as a regulariser to prevent overfitting.

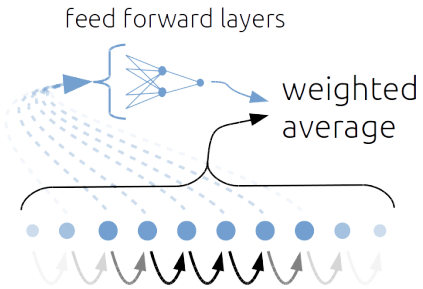

Figure 6. Schematic of attention mechanism; modified from [5].

The last step is the classification of the aggregated describing vector, implemented as a second (64-8-1) multilayer perceptron with a single output neuron with tanh activation.

2.2. Training

Hinge loss is used as optimising objective, using a weighting with inverse class frequencies 0.789/0.211. The differences between cross entropy loss and hinge loss turned out to be negligible.

The stochastic gradient descent variant Adam is applied as optimiser [7]. We used a training/validation split of $85\%/15\%$, with mini batches of 128 recordings using prepadding with zeros for shorter sequences. The order of samples is permuted after each epoch. Instead of training a fixed number of epochs, we perform two training runs of 96h with learning rates of 10^{-4} and 10^{-5} respectively.

We preprocess the recordings by normalising the amplitude using $1\%/99\%$ percentiles and clipping exceeding values. The weights of the best epoch measured by performance on whole dataset are used for the submitted challenge entry.

In addition to dropout we use l2 weight regularisation of 0.01 for dense feed forward layers, convolutional layers and GRU layers. Convolutional layers were initialised with sine waves of random frequency between 10 Hz and 100 Hz. Glorot uniform initialisation is used for dense feed forward layers and GRU input weights and orthogonal initialisation for recurrent weights.

Networks were implemented and trained using Keras [8] running on top of Theano [9]. The training was done on CPUs. Up to 64 GB of memory were temporarily required, depending on batch size and length of the recordings.

3. Data augmentation

Big neural networks can be quite hungry in terms of training samples required. Whereas currently trending deep neural networks in speech recognition, machine translation and computer vision use up to a million samples, a restricted set of 3,153 heart sound recordings is available for this Physionet Challenge. We work around this limitation by augmenting the heart sound recordings using various audio effects provided by SoX [10]. All effects used a listed in table 2.

Table 2. Effects used for data augmentation.

Effect	Parameters
Tempo	+10%,-10%
Speed	+5%,-5%
Dither	6/7/8 bits
Volume	+30%,-30%
Mix Speech	OSR 16/36 [11]
Pitch	+semitone,-semitone
Mix PASCAL	Atraining_normal 201101070538 [12]
Random	Combination of everything above

This results in a artificially increased dataset of 53,601 augmented recordings, which should prevent memorisation and improve generalisation. To further impede memorisation we randomly crop up to 5% from the beginning and up to 20% from the end of a recording dynamically during training.

4. Results

Using a moderately sized neural network and a training/validation split of $85\%/15\%$, we can report a result of 0.89 using the non-revised challenge scoring (0.90 on training data). A detailed listing per dataset is shown in table 3. The highest submitted phase two entry has a non-revised score of 0.77 and a revised score of 0.827.

Concerning the challenge sandbox our submitted entry uses less than 20% of the available computation quota.

Table 3. Classification results for the challenge datasets A-F, without training/validation split.

Dataset	# Recordings	Sensitivity	Specificity	Score
A	409	0.99	0.11	0.55
B	490	0.79	0.53	0.66
C	31	1.00	0.00	0.50
D	55	1.00	0.07	0.53
E	2054	1.00	0.98	0.99
F	114	1.00	0.02	0.51
Total	3153	0.96	0.83	0.89

5. Discussion

We present a deep neural network as solution to the 2016 PhysioNet/CinC Challenge. The network is strictly regularised and small in terms of weights ($21,924$) compared to commonly used deep architectures. This reflects the limited number of only 3153 training samples, although the number of actual training patterns has been artificially increased by a factor of 17 by augmentation.

Applying a $85\%/15\%$ training/validation split, we achieve scores between 0.89 and 0.90 with a difference of typically about 0.01 between training and validation data. At first glance this results indicate a good generalisation of the classifier. However, performance on the subsets of the dataset differs dramatically (from 0.99 for subset E to 0.51 for F, see table 3). It seems that during training the network mainly adapts to dataset E that in fact comprises the majority of the PCGs. The final score of only 0.83 on the hidden challenge dataset is consistent with this findings. Interestingly, the revised scoring resulted in an increased score and a reduced difference between performance on training data and hidden data.

For the final classification step, the feed forward attention mechanism turned out to be superior to naïvely using the output of the last interval of the recurrent layer.

We have to conclude that, in contrast to the apparently good generalisation, prediction of unseen data is still a challenging task for the neural network. Additional work is necessary to further improve its predictive power.

A bigger neural network with more adjustable weights might be necessary to represent all different types of PCGs in the highly diverse training and validation data. However, training of such a bigger neural network would require additional methods for augmentation or at best, (much) more example PCGs for training.

References

[1] Liu C, Springer D, Li Q, Moody B, Juan RA, Chorro FJ, Castells F, Roig JM, Silva I, Johnson AE, 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(11).

[2] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR ;abs/1502.03167.

[3] Cho K, van Merrienboer B, Bahdanau D, Bengio Y. On the properties of neural machine translation: Encoder-decoder approaches. CoRR ;abs/1409.1259.

[4] Chung J, Gülçehre Ç, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR ;abs/1412.3555.

[5] Raffel C, Ellis DPW. Feed-forward networks with attention can solve some long-term memory problems. CoRR ; abs/1512.08756.

[6] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 2014;15:1929–1958.

[7] Kingma DP, Ba J. Adam: A method for stochastic optimization. CoRR ;abs/1412.6980.

[8] Chollet F. Keras. `https://github.com/fchollet/keras`, 2015.

[9] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. arXiv e prints ;abs/1605.02688.

[10] Sound exchange. `http://sox.sourceforge.net/`.

[11] Open speech repository. `http://www.voiptroubleshooter.com/open_speech/`.

[12] Bentley P, Nordehn G, Coimbra M, Mannor S. The PASCAL Classifying Heart Sounds Challenge 2011 (CHSC2011) Results. `http://www.peterjbentley.com/heartchallenge/`.

Address for correspondence:

Christian Thomae
THM University of Applied Sciences
Wiesenstraße 14
D-35390 Gießen, Germany
christian.thomae@mni.thm.de