Normal / Abnormal Heart Sound Recordings Classification Using Convolutional Neural Network

Tanachat Nilanon[1], Jiayu Yao[2], Junheng Hao[3], Sanjay Purushotham[1], Yan Liu[1]

[1] University of Southern California, Los Angeles, USA
[2] Emory University, Atlanta, USA
[3] Tsinghua University, Beijing, China

Abstract

As part of the PhysioNet / Computing in Cardiology Challenge 2016, this work focuses on automatic classification of normal / abnormal phonocardiogram (PCG) recording, with the aim of quickly identifying subjects that need further expert diagnosis. To improve the robustness of the classifiers by increasing the number of training samples, the recordings were windowed into 5 second segments and our classifiers were trained to classify these segments. Overall recording classification was then generated using a voting scheme from classification results of its segments. Our features include spectrograms and Mel-frequency cepstrum coefficients. Our best submission result during the official phase, evaluated on the whole hidden test set, has a score of 0.811, with 0.770 sensitivity and 0.853 specificity.

1. Introduction

Automatic heart sound classification has promising potential to accurately detect heart pathology [1]. It can be used in non-clinical environments such as patient's residence by medical personnels as a quick heart pathology screening technique, or it can be used as a component in an efficient triaging system in clinical environments. Despite many attempts to develop a general purpose algorithm, most of the existing literature on this topic rely on small, private datasets for training, validating, and testing of their algorithms. Some published works even train and evaluate on the same set of data. The PhysioNet / Computing in Cardiology Challenge 2016 aims to provide the largest dataset to date along with a platform for common evaluation of different algorithms.

Although there are many different types of heart pathology that are audible via auscultation, the focus of this challenge is to classify whether the patient has "normal" or "abnormal" heart sound. This grouping merges distinct types of heart pathology under a common "abnormal" label, which must be properly handled. Heart sound recordings can also be classified as "unsure" when there is a significant presence of noise. Challenge participants are given a common evaluation metric to guide their algorithm design. More details can be found in [1].

2. Methods

We made two assumptions in designing our algorithm. The first is that abnormality of heart sound can be adequately determined in 5 seconds. This is based on the shortest recordings in the challenge being 5 seconds in length. We consulted general practitioners and they concurred with this assumption. Another assumption we made is that if there is a presence of heart pathology, corresponding sound(s) can be observed in every beats. One practitioner commented on this that while this is generally true, the amplitude of some heart pathology sounds can vary with intrathoracic pressure; an example would be aortic stenosis murmur (Gallavardin phenomenon). We did not directly handle such cases.

Our classifiers are trained to classify normal / abnormal label from 5 second segments extracted from a recording instead of from the whole recording. This helps us in two ways: it increases the number of available training samples from around $3,000$ to around $60,000$, significantly reducing overfitting, and it also forces the classifiers to "see through the noises". Instead of having to address another task of classifying whether the recordings are too noisy or not, our models can learn to ignore the noises. Our reasoning behind this is that the classifiers, as opposed to human doctors, do not suffer from limited working memory and should outperform humans in ignoring noises. The 5 second segments are extracted using standard sliding window method with a stride of 1 second.

We begin this section by describing features we used to train our classifiers in Subsection 2.1, then continue with baseline classifiers and our model in Subsection 2.2. In Subsection 2.3, we describe data splits we employed in training our models. Lastly, we discussed how we combine

segment-level classification into record-level classification in Subsection 2.4.

2.1. Feature extraction

Our input features are based on time-frequency features of the raw signal. There are many proposed methods that attempt to capture the intensity and the pitch of sound. In this work, spectrograms was picked as our main choice due to its robustness and interpretability.

2.1.1. Spectrograms

We extract Power Spectral Density (PSD) features with a window size of 150 ms and a stride size of 50 ms. This is based on the expected duration of the Fundamental Heart Sounds (FHSs). The shortest expected FHS is 50 ms long [2] while the expected S1 sound is 122 ms with ± 32 ms 95% confidence interval and the expected S2 sound is 92 ms with ± 28 ms 95% confidence interval [3]. The heart sound recordings are normalized by its root-mean-square energy before spectrogram calculation. The spectrogram calculation was done in the power spectral density mode. We also derive energy distribution features from spectrogram by taking its average along the time dimension.

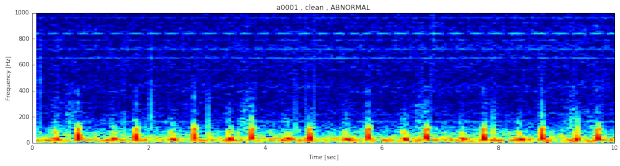

Figure 1. Example of a spectrogram

2.1.2. Mel-frequency cepstrum coefficients

Mel-frequency cepstrum coefficients [4] (MFCCs) is a feature widely used in speech recognition. It is designed to mimic human sound perception and has been shown to work extremely well for speech. We extract 13 coefficients using analysis window of 25 ms with 10 ms stride.

2.2. Classifiers

2.2.1. Baseline models

Our baseline classifiers include logistic regression, support vector machines, and random forests.

We trained logistic regression both on the power spectral density features and the energy distribution features. Due to class imbalance, a class weight ratio of 1 to 4 (normal to abnormal) was used in the training process.

We trained support vector machines both on the power spectral density features and the energy distribution features. A class weight similar to logistic regression was used also due to class imbalance. We tried three different kernels: linear, polynomial, and radial basis function. Hyperparameters are adjusted using validation split.

We trained random forests using a forest size of 100 and 200 with default parameters of scikit-learn Python package.

2.2.2. Convolutional neural networks

Each PSD of a 5 second window is treated as if it is a one channel image. In the lower part of the network, we use a convolutional layer to capture local changes. One hidden dense layer sits atop the convolutional layer. The output layer contains a single neuron with sigmoid activation function. Binary cross entropy is used as the objective function. A graphical illustration of our model can be seen in Figure 2

We train our models using stochastic gradient descent using Adam [5] optimizer with Theano [6] and Keras [7] Python library. Dropout [8] is also applied to help reduce overfitting.

2.3. Experimental design

We employed two schemes in splitting the training sets (a - f) into training split, validation split, and testing split.

The first split, denoted by A, uses stratifed 5-fold cross validation splitting technique. The recordings are stratified by database source, diagnosis label, and signal quality index (SQI) label. The ratio of (training split : validation split : testing split) is $(3 : 1 : 1)$. We denote them as $A1$ to $A5$. These splits allow the classifiers to see the data from all databases in the training set, which does not mimic the challenge's evaluation scheme that contains two hidden databases.

The second split (B) is our attempt at mimicking the challenge's evaluation scheme. We have three splits in this scheme and they only contain training and validation split. The splits are described in details in Table 1.

Table 1. Second data splitting scheme (B)

Split name	Training split	Validation split
$B1$	a + 70% {b, c, d, e}	f + 30% {b, c, d, e}
$B2$	f + 70% {b, c, d, e}	a + 30% {b, c, d, e}
$B3$	70% {b, c, d, e}	a + f + 30% {b, c, d, e}

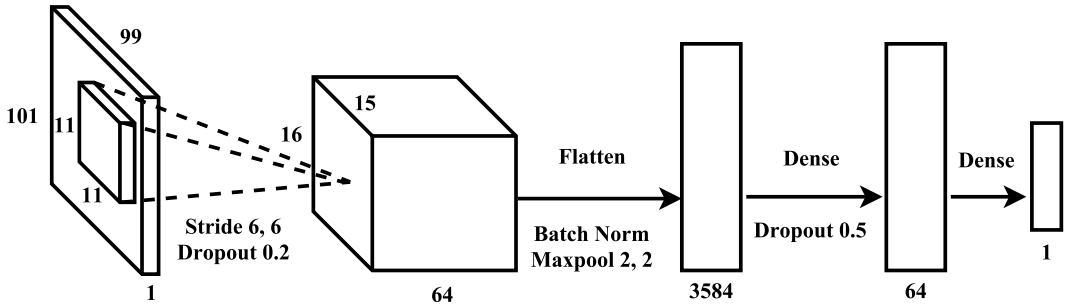

Figure 2. Our convolutional neural network architecture. We had to restrict the size of our model to deal with small sample size and overfitting.

2.4. Combining segment-level classification into record-level classification

We experimented with two schemes for combining segment-level classifications: majority voting and mean raw value.

In majority voting, the number of segments classified as normal / abnormal are counted. Whichever class has a larger number of segments is output as the record-level prediction. Ties are broken using mean raw value scheme described next.

In mean raw value scheme, we compute the average over the raw sigmoid output of each segment. We then classify based on abnormality threshold (0.5 by default)

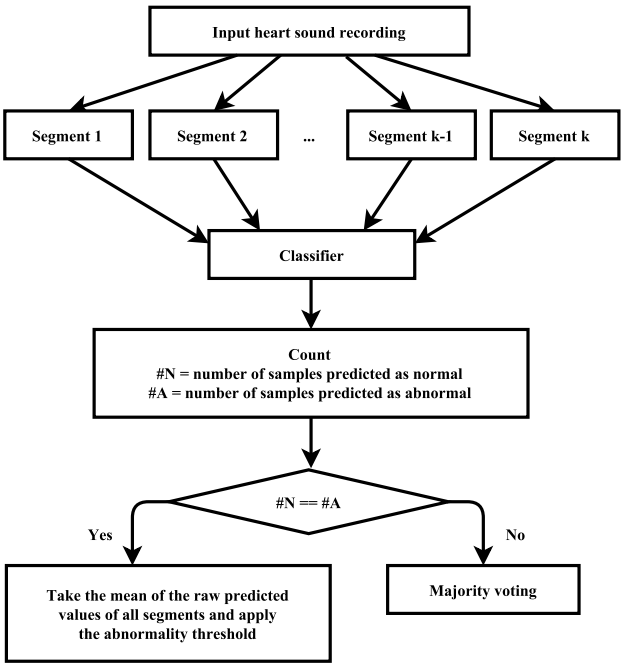

Figure 3. Majority-voting scheme for record-level classification

3. Results

The results we obtain on the training set do not match the results on the hidden test set.

3.1. Results on the training set

Our attempts at mimicking the evaluation scheme resulted in a widely diverse model performance. The results are listed in Table 2 and 3.

Table 2. Results on training set using A split (averaged over 5 folds)

Model	Sensitivity	Specificity	Score
LR (best)	0.710	0.688	0.699
SVM (best)	**0.826**	0.832	0.829
RF (best)	0.687	0.685	0.686
CNN (best)	0.810	**0.960**	**0.885**

Table 3. Results on training set using B split

Model	Split	Sensitivity	Specificity	Score
LR (best)	$B1$	0.722	0.896	0.809
	$B2$	0.845	0.489	0.667
	$B3$	0.778	0.628	0.703
SVM (best)	$B1$	0.846	0.615	0.730
	$B2$	0.868	0.593	0.731
	$B3$	0.877	0.552	0.715
CNN (best)	$B1$	0.891	0.797	0.843
	$B2$	0.725	0.820	0.772
	$B3$	0.494	0.873	0.683

3.2. Results on 20% of the hidden test set

The CNN model clearly outperforms baseline models as can be seen in Table 4.

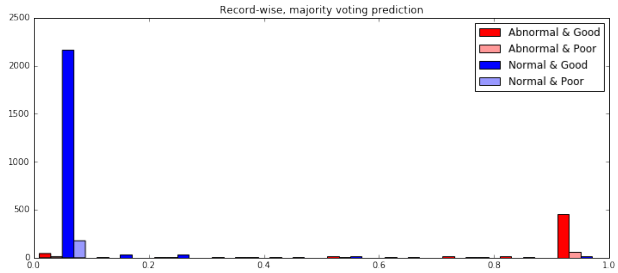

Figure 4. Histogram of classification from the same split. x-axis is the sigmoid output value. y-axis is the number of recordings. Ground truth classification is denoted in the legend.

Table 4. Results on random 20% of the hidden test set

Model	Sensitivity	Specificity	Score
LR (best)	0.726	0.758	0.742
SVM (best)	**0.756**	0.756	0.756
CNN (best)	0.735	**0.892**	**0.813**

3.3. Best result on the whole hidden test set

Our best result on the hidden test set achieved **0.770** sensitivity and **0.853** specificity with **0.811** score. Only the best result was provided by the challenge organizer in an effort to discourage overfitting to the hidden test set.

4. Discussion

There is still room for improvement as our CNN model has low sensitivity. Several hyperparameters such as the class weights and the abnormality threshold can be adjusted to achieve higher score. However, the underlying problem with statistical methods such as the models used here is that they do not generalize well. Incorporating domain knowledge as has traditionally been done could improve the models' generalizibility.

Regarding the option to classify the recordings as unsure, we have found that we are better off without classifying that label. Doing so changes the task from two class classification to three class classification or two phase two class classification (classifying unsure first, then classify normal / abnormal). Lack of training samples with poor signal quality makes it hard to achieve high recall for unsure class classification. Furthermore, empirically, the classification performance on poor quality signal is only slightly worse than their good signal quality counterpart. In table 5, we list the classification result of a CNN model from one split. Figure 4 shows the histogram of classification of the same CNN model from the same split.

Table 5. Results from one split

Reference		Prediction		
Diagnosis	SQI	Abnormal	Normal	% correct
Abnormal	Good	97	21	0.82
Abnormal	Poor	16	6	0.73
Normal	Good	15	448	0.97
Normal	Poor	4	37	0.90

Acknowledgements

We would like to express our thanks to Nvidia for their generous GPU donations.

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(9).

[2] Springer DB, Tarassenko L, Clifford GD. Logistic regression-hsmm-based heart sound segmentation. IEEE Transactions on Biomedical Engineering 2016;63(4):822–832.

[3] Schmidt S, Holst-Hansen C, Graff C, Toft E, Struijk JJ. Segmentation of heart sound recordings by a duration-dependent hidden markov model. Physiological Measurement 2010; 31(4):513.

[4] Ganchev T, Fakotakis N, Kokkinakis G. Comparative evaluation of various mfcc implementations on the speaker verification task. In Proceedings of the SPECOM, volume 1. 2005; 191–194.

[5] Kingma D, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv14126980 2014;.

[6] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. arXiv e prints May 2016;abs/1605.02688. URL `http://arxiv.org/abs/1605.02688`.

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

[8] Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 2014;15(1):1929–1958.

Address for correspondence:

Tanachat Nilanon
Department of Computer Science
University of Southern California
Los Angeles, California, 90089 USA
nilanon@usc.edu