Heartbeats

Machine Learning Engineer Nanodegree

Luis Govea
April 3rd, 2018

I. Definition

Project Overview

“According to the World Health Organisation, cardiovascular diseases (CVDs) are the number one cause of death globally: more people die annually from CVDs than from any other cause. An estimated 17.1 million people died from CVDs in 2004, representing 29% of all global deaths. Of these deaths, an estimated 7.2 million were due to coronary heart disease. Any method which can help to detect signs of heart disease could therefore have a significant impact on world health.”

A heartbeat is composed of two sounds “lubb-dupp” or also known as S1 and S2 sounds respectively. The first sound, S1, is caused by the closure of the inflow valves M1(mitralis) and T1(tricuspid) at the beginning of ventricular contraction, systole, when blood is pushed out to from the heart to the body and lungs. The second heart sound, S2, is caused by the closure of the valves A2 (aortic) and P2 (pulmonary) at the end of ventricular systole and the beginning of ventricular diastole. This phase is a bit longer since the heart is being refilled. With S1 and S2 sounds a heartbeat could be classified within the following categories: normal, murmur, extra heart sound.

Normal Category
A normal heart sound has a clear “lub dub, lub dub” pattern, with the time from “lub” to “dub” shorter than the time from “dub” to the next “lub”. This pattern can be described in the following illustration:

…lub……….dub……………. lub……….dub……………. lub……….dub…………….
lub……….dub…

Murmur Category
Heart murmurs sound as though there is a “whooshing, roaring, rumbling, or turbulent fluid” noise in one of two temporal locations: (1) between “lub” and “dub”, or (2) between “dub” and

“lub”. They can be a symptom of many heart disorders, some serious. There will still be a “lub” and a “dub”. One of the things that confuses non-medically trained people is that murmurs happen between lub and dub or between dub and lub; not on lub and not on dub. Below, you can find an asterisk* at the locations a murmur may be:

…lub..*...dub……………. lub..*..dub ……………. lub..*..dub ……………. lub..*..dub … or
…lub……….dub…*….lub………. dub…*….lub ………. dub…**….lub ……….dub…

Extra Heart Sound Category
Extra heart sounds can be identified because there is an additional sound, e.g. a “lub-lub dub” or a “lub dub-dub”. An extra heart sound may not be a sign of disease. However, in some situations it is an important sign of disease, which if detected early could help a person. The extra heart sound is important to be able to detect as it cannot be detected by ultrasound very well. Below, note the temporal description of the extra heart sounds:

…lub.lub……….dub………..………. lub. lub……….dub…………….lub.lub……..…….dub……. or
…lub………. dub.dub………………….lub.……….dub.dub………………….lub……..…….dub. dub……

The goal of this project is to build an audio classifier capable of recognizing different types of heart anomalies.

Traditionally, to diagnose heart conditions, doctors needed to carefully evaluate heartbeats from different locations of the body accurately. Doing this for each patient might result on a highly time consuming task. Apart from this, specialized domain knowledge is needed to correctly identify such sounds.

The recent advancements in hardware, datasets and algorithms has permitted Deep Learning to thrive in recent years. In broadly terms, it is possible to say that Deep Learning is about identifying patterns by “connecting the dots”. Translated into the field of Healthcare such “dots” or symptoms can be related to disorders.

Deep learning algorithms get better at diagnosing the same way a physician would: with practice. Therefore, high quality collections of data are needed to train such algorithms. For this project, an annotated audio dataset is needed, consisting of sound clips and its corresponding labels indicating the diagnosis of the recordings. This would enable the problem to be tackled as a supervised learning task.

To begin with, the dataset from the PhysioNet 2016 Cardiology Challenge will be used. This dataset consists of 3,126 heart sounds of normal and abnormal recordings. Even if this set does not contain specific labels for normal, murmur, extra heart sound, artifact categories, it is still useful to explore and extract common features from the raw audio recordings.

Later a more specialized classifier will be trained to identify the categories of interest. The dataset used for this second part comes from kaggle and has 176 audio files.

One important thing to mention is that once a deep learning algorithm is trained it can operate on a daily basis without rest. Usually such algorithms are trained with thousands of examples, this means that such algorithms can become proficient or even excel in solving their corresponding problem. Sometimes they can even surpass human experts. This is an excellent way to provide highly specialized diagnosis to humanity.

Problem Statement

The problem at hand will consist in the following. Given an audio file of heartbeats be able to classify such recording as either: normal, murmur, extra heart sound and artifact. The artifact category will be used when there is no discernable heart sounds, this is useful to indicate a final user to do another recording.

The audio files provided will be of different lengths and also the contents of such files will vary. This means that even if it’s possible to trim the files to have an equal size length it is not feasible to compare an exact location with different files. In one file a particular time can be the start of a heartbeat and in another file the exact time could be noise. This indicates, that somehow, a method will be needed to represent audio in a different manner.

Once the challenge of finding a way to represent audio in a different way is solved, the challenge then becomes to identify patterns within the audio representations and generalize a solution to the classification problem at hand. All of this points towards a suitable application for deep learning.

As a final step, a measurement for the performance of the model is needed to know whether the approach taken is generating satisfying results.

Metrics

Evaluating a classifier is often trickier than evaluating a regressor. As it will be explained, for a classifier, accuracy is generally not the preferred performance measure. Let's look at an example to explain why. Let's imagine that a classifier was made to tell whether an image is a hotdog or not hotdog. The training set for this imaginary classifier consists of 1000 images out of which 900 of them are hotdogs and the rest are not. Even if a very dumb classifier was made to always say that an image is a hotdog, its accuracy will be of 90%. Excellent right? Not really. This simple example demonstrates why accuracy is not the preferred performance measure for classifiers, especially when dealing with skewed datasets.

A much better approach is to use other kinds of performance metrics. But before telling which ones they are, it is necessary to first explain what is a confusion matrix. The general idea behind a confusion matrix is to count the number of times instances of class A are classified as class B. Coming back to the hotdog example, this will tell us the number of times something that was not a hotdog was classified as a hotdog and vice versa. This is really useful since it will provide information to know: true positives(TP), true negatives (TN), false positives (FP), false negatives(FN). Now that the meaning for a confusion matrix is known, is possible to explain: Precision, recall and f1 score.

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$precision = \frac{TP}{TP + FP}$$

Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

$$recall = \frac{TP}{TP + FN}$$

F1 score is the weighted average of Precision and Recall. This score favors classifies that have similar precision and recall.

$$f_1 = 2\left(\frac{precision * recall}{precision + recall}\right)$$

In the Data Exploration section it will be really clear that both datasets, Kaggle and Physionet, are imbalanced. Therefore, besides accuracy, these performance metrics(precision, recall and f1 score) will be used to evaluate the models.

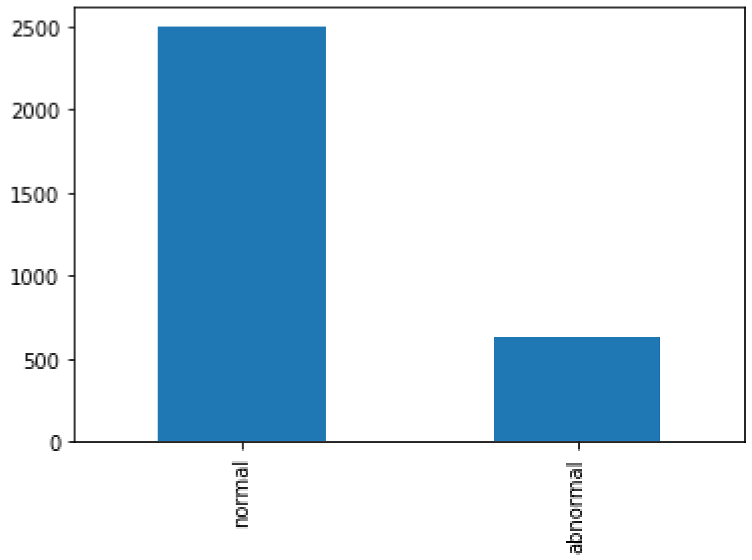
II. Analysis

Data Exploration

In this section the nature of the audio files provided from the PhysioNet 2016 Cardiology Challenge and Kaggle Heartbeat Sounds challenge will be discussed.

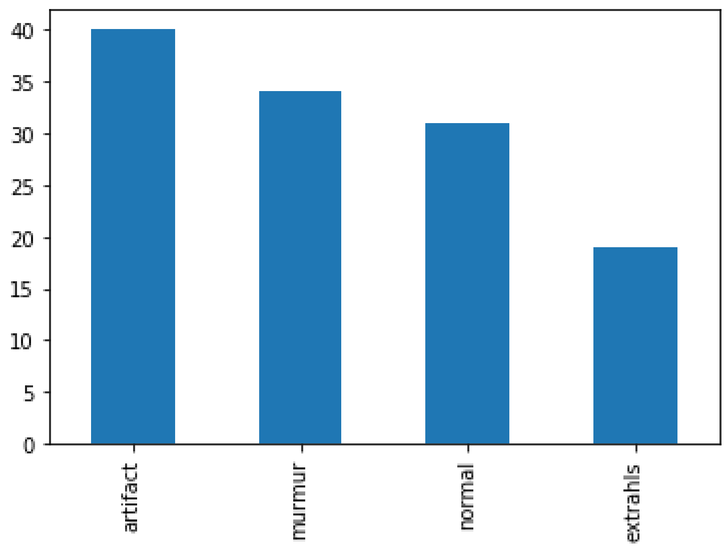
First, lets begin with the PhysioNet 2016 Cardiology Challenge dataset. This dataset has phonocardiogram recordings of normal and abnormal recordings. The training set consists of five databases (A through E) containing a total of 3,126 heart sound recordings, lasting from 5 seconds to just over 120 seconds. All audio files are provided in WAV format and have a sampling rate of 2000 HZ. A CSV file within each database contains the corresponding label for each file.

In order to check the balance of the dataset, it is necessary to join all the five databases provided and then do the corresponding label counts. This produces the following graph:



The normal category has 2495 values and the abnormal category 631.

The second dataset comes from Kaggle and is split into two sources, A and B: A was collected from an iPhone app and B from a clinical trial in hospitals using a digital stethoscope. For the purpose of this project only dataset A will be used and not B. Dataset A contains audio files that are of varying lengths, between 1 second and 30 seconds. Dataset A contains recordings from healthy subjects and patients with different kinds of heart conditions: normal, artifact, murmur and extra heartbeat. There is a total of 176 audio files out of which only 124 files are labeled. Only the labeled audios will be used, segregating the categories generates the following graph:

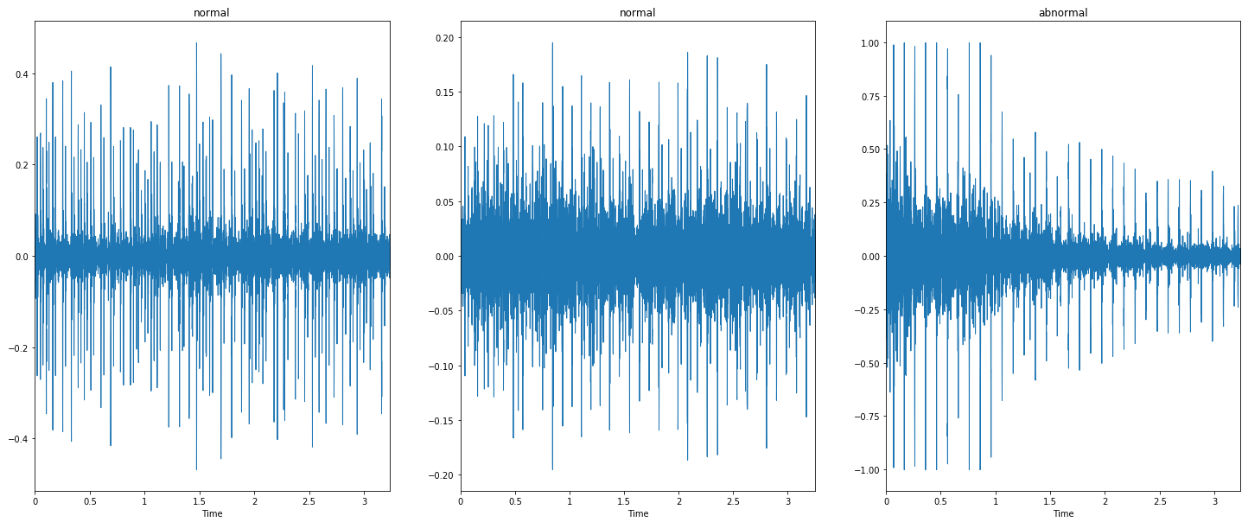


The Artifact category has a total of 40, murmur 34, normal 31 and extra heartbeat 19.

As it can be seen from the graphs, both datasets are imbalanced.

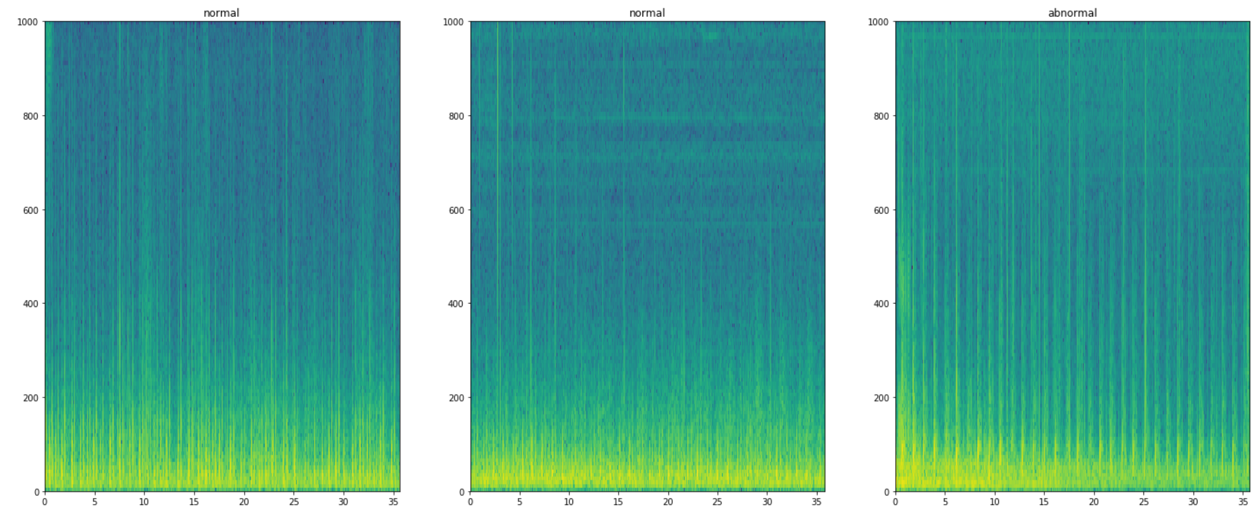
Exploratory Visualization

Audio can be visualized as a waveform plot, which depicts the amplitude of the sound at each successive time interval. Below are shown some randomly chosen waveforms.



Time is represented on the horizontal axis and amplitude on the vertical. As it can be seen, identifying a heart anomaly from a visual interpretation is a really challenging task.

An alternative visual representation for sound is a spectrogram. With them, is possible to see a spectrum of frequencies of sound as they vary with time.



With these plots, is possible to illustrate the complexity of the problem. To compare audio it is not reasonable to compare the respective value at an arbitrary point in time, even if two audio files are of the same size. Instead, it is necessary to to compare patterns over regions of the recording over time.

Algorithms and Techniques

For this project, it is clear that the best choice is to build a classification model using deep neural networks capable of automatic feature learning. There are different kinds of neural network architectures, but for this project two will be chosen: Feed Forward Network and Convolutional Neural Network. As a sanity check is always a good idea to try simple architectures before trying more complex ones, this is the reason why a simple Feed Forward Network will be implemented first and later a Convolutional Neural Network.

Feed Forward Networks, feed information from the front input nodes all the way to the output nodes, through several hidden layers. Then, with an error function, the method of backpropagation calculates the gradient of the error function with respect to the neural network weights. This produces a difference between what is expected and what the network produced. This error is then used to adjust the weights of the nodes, which with several repetitions causes the output to become more accurate.

The difference between a densely connected layer and a convolution layer is that dense layers learn global patterns in their input feature space and convolutional layers learn local patterns. Translating this to the case of images would mean that a convolutional layer would be able to recognize a pattern even if it is found on a different location from where it was learned. In the case of a densely connected network would have to learn the patterns anew if it appeared at a new location.

Benchmarks

Several papers have been published reporting classification performance achieved on the PhysioNet 2016 Cardiology Challenge. This challenge consists in classifying heartbeats as either normal or abnormal. One paper, published by Palo Alto Research Center, investigated using heat maps of the time-frequency distribution of signal energy and uses a deep convolutional neural network to automatically classify normal versus abnormal heart sound recordings. This paper reported having a sensitivity of 76.5% and and specificity of 93.1%.

In another paper, published by Marton Aron Goda and Peter Hajas, investigated using Support Vector Machines and audio feature extraction of time-domain features, frequency-domain features and wavelet envelope features. This paper reported having 83.77% sensitivity and 76.8% specificity.

Before looking into a complicated and computationally expensive model such as a CNN for the classification of normal and abnormal heartbeats, first a simple 2 layer feed forward network will be used to establish a model to use as a benchmark. This is to make sure that any further complexity delivers additional value.

The baseline for the classifier of normal and abnormal heartbeats would be an accuracy of about 60%. This because the Physionet Cardiology Challenge 2016 provided a baseline score of 71%.

For the second classifier, a base benchmark is not present in the Kaggle description of the problem and in order to overcome this, a common sense approach will be taken. From the 176 files present in the Kaggle dataset only 124 are labeled. The category which has the greater count is Artifact with a value of 40. A classifier that always predicts that an input file is an Artifact will be correct 32%(40/124) of the time. From this, it is possible to conclude that the learning approach should beat this value.

III. Methodology

FNN Data Preprocessing

Each second on the audio file of the Physionet Cardiology Challenge 2016 consists of 2000 data values. This means that a 5 second sample would consist of 10000 values. A neural network would need one input node per feature, a 5 second sample would require 10000 input nodes and a number close to this one for the hidden nodes. This scales as the number of seconds per sample increases, which eventually will become a problem.

Doing some research an interesting approach was taken in order to solve this problem in this blog. The blog post indicates a way to do dimensionality reduction of audio using feature extraction methods provided by the Python library librosa. The feature extractions methods involved are:

- MFCC: representation of the short-term power spectrum of a sound
- Mel-frequency cepstral coefficients: Coefficients that collectively make up an mel-frequency cepstrum
- Chromagram of a short-time Fourier transform: Projects into bins representing the 12 distinct semitones of the musical octave
- Octave-based spectral contrast: Distribution of sound energy over octave frequencies
- Tonnetz: Estimates tonal centroids features

Combining these 5 feature extractions will result in a vector of 193 values. Is important to remark that this vector will always have the same length even if the audio files are of different length.

FNN Implementation

A feed forward neural network was implemented using Keras Sequential Model. The network consists of an input layer and a densely connected hidden layer. The activation function used is relu, with the exception of the last layer which has a sigmoid activation function.

The Physionet Cardiology Challenge 2016 dataset consists of normal and abnormal heartbeat labels, therefore, the loss function chosen was Binary Cross Entropy. This measure is critical since is what the network uses to evaluate mistakes in the networks output. The model is trained over several successive epochs. One would expect that with every iteration the loss value decreases, this would be an indication that the model is learning.

One tactic employed to prevent overfitting was to add a dropout policy before the output layer. The value used was of 25%, this means that 25% of a neuron's activation output will be ignored, and not propagated. The idea is to randomly throw away information so the network learns patterns and not to memorize the data.

Finally the optimizer used was Adam, this optimiser is considered to be an adaptive estimator, meaning that it does not require manual tuning of hyper-parameters.

FNN Challenges Faced

The Librosa library provides all the methods necessary to perform the dimensionality reduction process, which turned out to be a really simple implementation. The only setback is that the amount to process all the files is quite high. To facilitate further research, the processed files are included at the Github repo. The good this is that after the files are processed, training the FNN is extremely fast. Because the network is so simple this can be done with CPU.

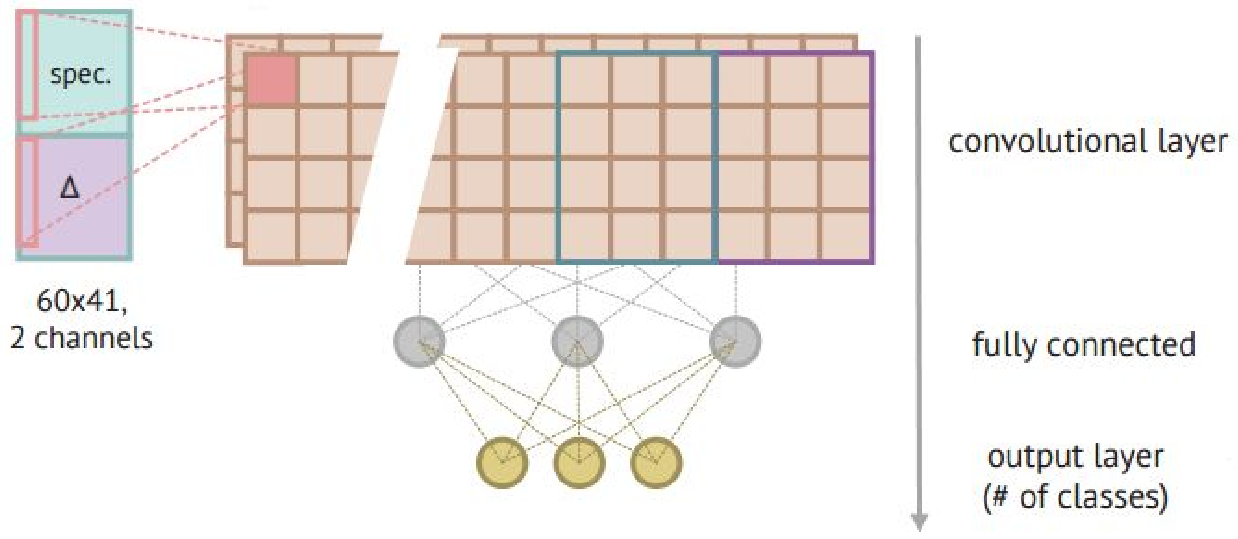
CNN Data Preprocessing

The previous section explained how a method to extract audio features using methods available by the librosa library. This process resulted on a constant array of 193 features independent of the audio size. This section will explain a different approach to process audio files so that the end result can be fed to a Convolutional Neural Network.

Convolutional Neural Networks, also known as convnets, are a type of deep-learning model almost universally used in computer vision applications. Convolutions operate over 3D tensors called feature maps. They are composed of two spatial axes and a depth axis. The convolution operation extracts patches from its input feature map and applies the same transformation to all of these patches producing an output feature map. The output feature map is still a 3D tensor, with the difference in that its depth axis stands for filters. Filters encode specific aspects of the input data.

To classify audio in the same way, a means of extracting audio features is required that creates the kind of input a CNN expects. A paper published by Karo J. Piczak named Environmental sound classification with convolutional neural networks, describes how to get equal size segments from varying length audio clips and how audio features can be fed into the network as separate channels.

The idea is to calculate log-scaled mel-spectrograms and their corresponding deltas from a sound clip. Because a fixed size input is needed, each sound clip is split into 41 overlapping frames, one for each of the 60 mel-bands, giving an array of 60 rows and 41 columns. The mel-spectrogram for each band/segment and its time-series deltas will become two channels, which becomes the input to feed into the CNN. Other features could be calculated in the same way and supplied as a separate input channel, but for this project just the mel-spectrogram will be used.



The implementation was based on sample code from this blog post. It is important to point out that this process will use a window size and an audio file will be split into several parts. This will serve as Data Augmentation since a single file will produce several training samples. In the method used for the FNN a constant array of 193 features was produced for each files, this will not be the case this time. In notebook 3 the same example file used for the FNN was processed with this method and instead of producing 193 features it produced a total of 369,000 features. The hope is that by having more information the end results will be better.

CNN Implementation

The model implemented in notebook 3 begins with two convolutional layers. The first with 48 kernels and the second with 96. In between these layers a Max Pooling operation was added.

Max pooling consists of extracting windows from the input feature maps and outputting the max value of each channel. The reason to use downsampling is to reduce the number of feature-map coefficients to process.

Once the convolutional layers were set in place, the output was fed into densely connected layers. The same technique of adding a Dropout policy was added to prevent overfitting, but this time instead of having a dropout of 25% one of 50% was used.

As with the FFN, the same activation function, Relu, was used throughout the network and the last layer has a sigmoid activation function and the optimizer used was Adam.

CNN Challenges Faced

As with the case with the preprocessing for the FNN, the librosa library provides provides the methods necessary to calculate the log-scaled mel-spectrograms. The tricky part was to iterate a fixed window over the audio files and do the necessary processing, but if the code found at notebook 3 is taken as an example it can be easily be adapted. It is import to remark that the time to process the files is way higher than the process done for the FNN, one mistake in the output shape of the processed files could cost several hours of processing. To overcome this, a small sample file can be at the repo to validate that the shape of the outputs in the correct format. Once the output files are in the correct format training the CNN is a really straight up process.

Transfer Learning Preprocessing

Up to this point all the analysis has been made to the Physionet Cardiology Challenge 2016 dataset and none has been made to the Kaggle dataset. The reason for this is because the Kaggle dataset is so small that first it was necessary to do analysis on a bigger dataset so at the end transfer learning could be applied to the smaller dataset.

Nonetheless, the different approaches taken to do feature extraction for the FFN and the CNN explained in the previous sections will be applied to the Kaggle dataset to compare results with and without transfer learning.

Transfer Learning Implementation

Transfer learning is very powerful technique popular in deep learning where a model developed for a task is reused as the starting point for a model on a second task.

The aim of this project is to be able to classify between four different kinds of heart conditions: normal, murmur, extra heart sound and artifact. The Kaggle dataset is the one that contains this information but as explained earlier a total of usable files is of 124 which is way too little to train a reliable model. To try to solve this issue, a CNN was trained with a similar dataset, the Physionet Cardiology Challenge 2016, to later use this pretrained model as a base for the Kaggle dataset.

There are two ways to use a pretrained network: feature extraction and fine-tuning. Feature extraction consists of using the representations learned by a previous network to extract interesting features from new samples. Then, these features are run through a new classifier. Feature extraction on the other hand consists of unfreezing a few of the top layers of a frozen model base and train the newly added layers and these top layers. This slightly adjusts the more abstract representations of the model in order to make them more relevant for the new problem at hand. For this project only feature extraction will be used.

A Convolutional Neural Network basically consists of two part: convolutional layers and a densely connected classifier. To be able to implement Feature Extraction is necessary to remove the densely connected classifier and just leave the convolutional base. The idea behind this approach is that the representations learned by the convolutional base are more genetic and reusable. Before adding a new densely connected classifier is necessary to freeze the convolutional base so the weights do not get modified with the new set of information and in this way preserving the more abstract patterns learned. As a final step a new densely connected classifier is added.

Transfer Learning Challenges Faced

In the previous sections all the problems related to the preprocessing of the files were solved and in this section all pieces of the puzzle were put together. Once a model is trained, transfer learning turns out to be a really simple process. No particular challenges were faced in this section.

IV. Refinement

The performance of deep neural network highly depends on a set of configurable parameters called hyperparameters. A classical approach is to use GridSearch to train different instances of a model that changes one hyperparameter at a time and then compares results and returns the best set of parameters. To accomplish this, the Keras scikit-learn interface was used to be able to use the GridSearch method available at the Sklearn library.

It is important to mention that because of the complexity of the models and the amount of time required to train such models is really high, just a few set of parameters were fined tuned.

This refinement part was only made to the Models used for the Physionet Cardiology Challenge 2016 since this model was later going to be used to implement transfer learning. The set of parameters used for GridSearch are the following:

- FFN: Epochs and dropout rate
- CNN: Dropout rate

The hyperparameters used to do the refinement of the FFN were the number of epochs ranging from 10 to 40 with a step of 10. The dropout rate ranging from 0.1 to 0.5 with a step of 0.1. To find more detailed results this information can be found in notebook 2 but in summary the number of epochs had the greater effect in the improvement of the FFN. As the epochs increase from 10 to 40 the accuracy of the model increases from about 0.85 to about 0.90. The dropout rate only caused small fluctuations in the accuracy. The best set of parameters for the FNN turned out to be epochs equal to 40 and dropout rate of 0.2.

In the case for the CNN only the dropout rate was used to do the refinement. The values used were a range between 0.1 to 0.5 with a step of 0.1. More details of the results can be found in notebook 3. In summary the values of the dropout did not cause much improvement of the network. The best value for the dropout rate turned out to be 0.4.

V. Results

This section reviews the results collected from all notebooks:

- FFN Physionet dataset:
  - Accuracy: 0.901534526854
  - Precision: 0.952459016393
  - Recall: 0.923688394277
  - F-Score: 0.937853107345
- CNN Physionet dataset:
  - Accuracy: 0.930317169361
  - Precision: 0.949037534374
  - Recall: 0.960876482499
  - F-Score: 0.954920315504
- CNN Physionet dataset GridSearch:
  - Accuracy: 0.933399988891
  - Precision: 0.962228240978
  - Recall: 0.950607463118
  - F-Score: 0.956382553021
- FFN Kaggle dataset:
  - Accuracy: 0.39
  - Precision: 0.394736842105
  - Recall: 0.394736842105
  - F-Score: 0.394736842105
- CNN Kaggle dataset:
  - Accuracy: 0.78
  - Precision: 0.775716694772
  - Recall: 0.775716694772
  - F-Score: 0.775716694772
- CNN Transfer Learning Kaggle dataset:
  - Accuracy: 0.78
  - Precision: 0.784148397976
  - Recall: 0.784148397976
  - F-Score: 0.784148397976

For this project two different benchmarks were set. One for the Physionet Cardiology Challenge 2016 and another for Kaggle problem.

The Physionet Cardiology Challenge 2016 provided a baseline score of 71%. From the results above is possible to see that this benchmark was surpassed by both models. The Kaggle problem description did not contained a base benchmark so one was set to beat random chance. This specified that the classifier should be right more than 32% of the time. Also from the results above is possible to see that this was also achieved.

There are several things that are worth mentioning in this section. The processed used to train the FNN for both databases demonstrated to have a huge difference in results. For the case of the Physionet dataset a f1 score was obtained of 0.9378, but this same method for the Kaggle dataset produced a f1 score of 0.3947. This probably has to do with the amount of data present in the Kaggle dataset. But, the method used to train the CNN produced a f1 score 0.7757 in the Kaggle dataset. This means that the method to do data augmentation turned to work wonderfully.

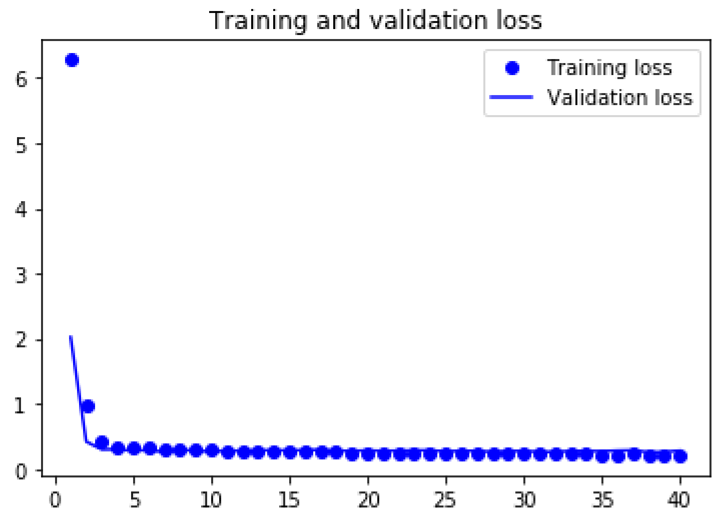
Personally, even if the results of transfer learning were not that different from the CNN trained with Kaggle dataset, this was kind of surprising in a positive manner. It is important to mention that the model trained to be used for transfer learning came from a different dataset in which the methods of recording are completely different. The Kaggle dataset presents recordings made by regular Iphone users in which they vary from user to user. Some have clear heart sounds, others a lot of noise and some are just not useful. The Physionet dataset contains audio files into which more attention was put into the recordings of the files. They contain some noise but in general more heartbeats can be listed to. Even if this is the case, some patterns were found in the Physionet data by the neural network that were useful for the Kaggle dataset. Finally, this project is not meant to be used in real clinical trials.This project only demonstrated the potential of neural networks in the Healthcare field. To use a similar model in the real world requires more data and also a more deeper analysis of the results.

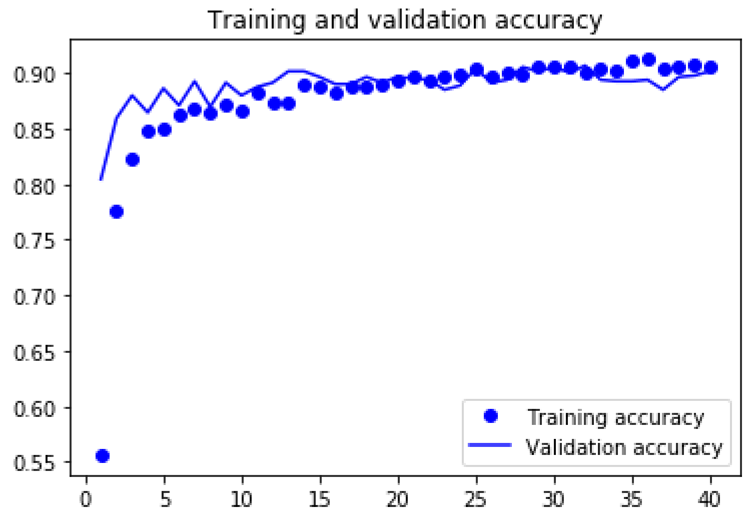
Conclusion

Free-form Visualization

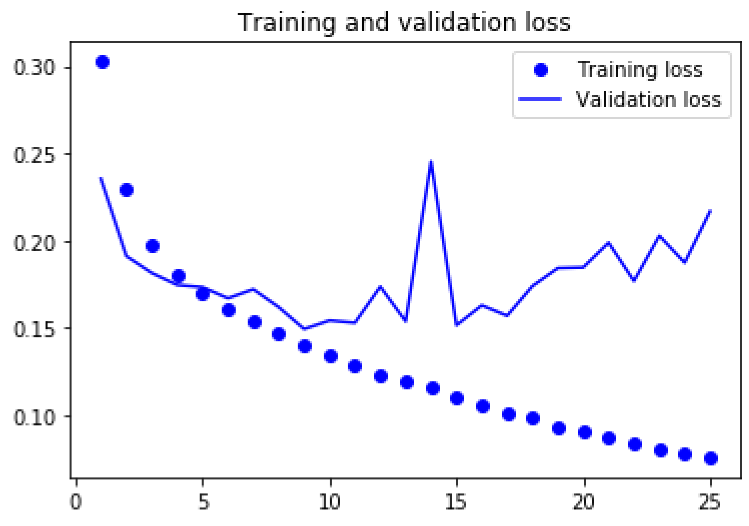
For this section the accuracy and loss graphs for each model will be presented as visualisations.

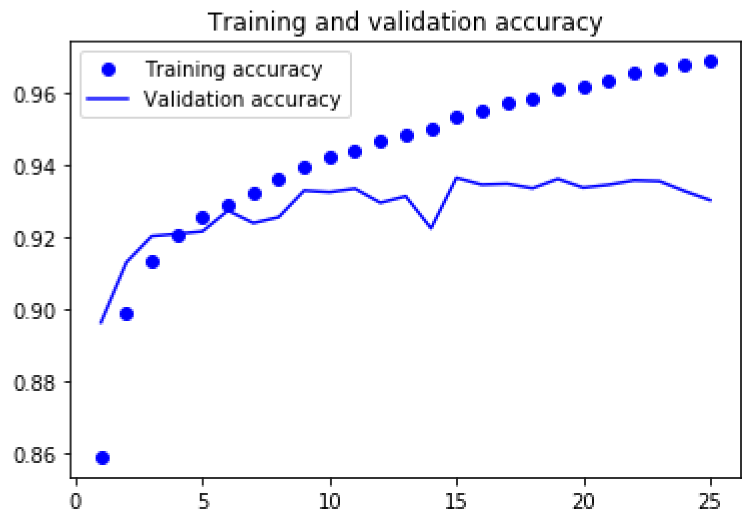
FFN Physionet dataset



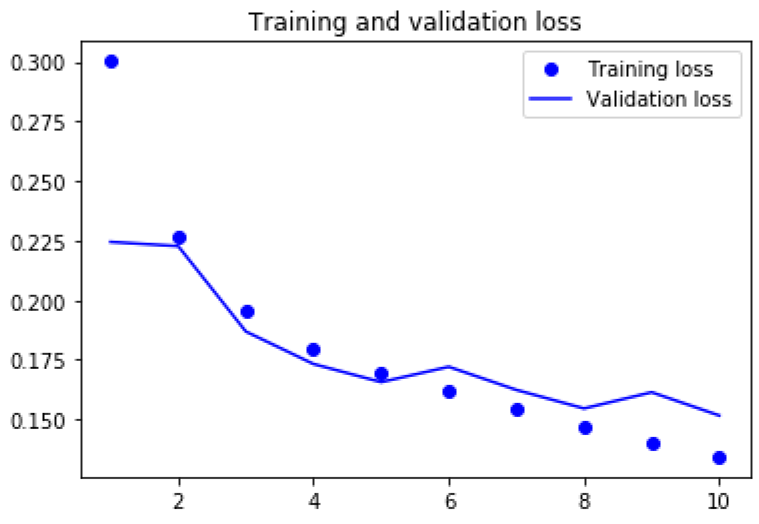


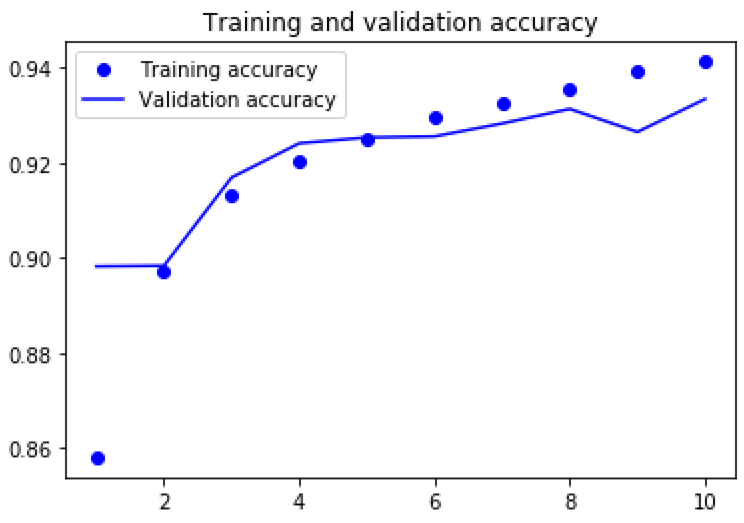
CNN Physionet dataset



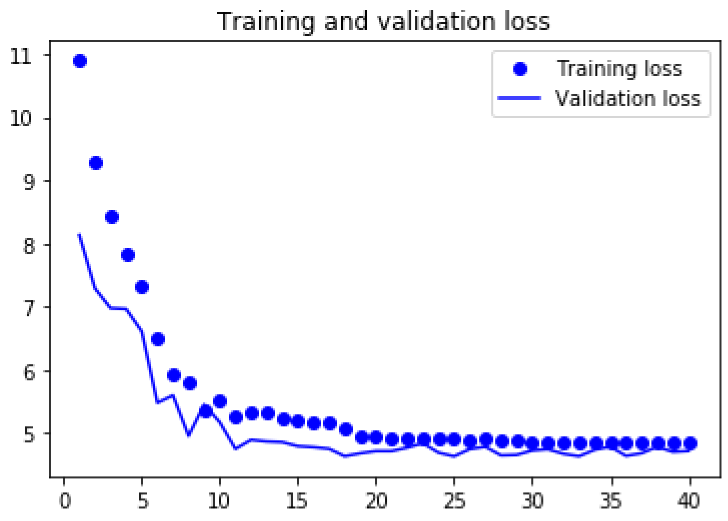


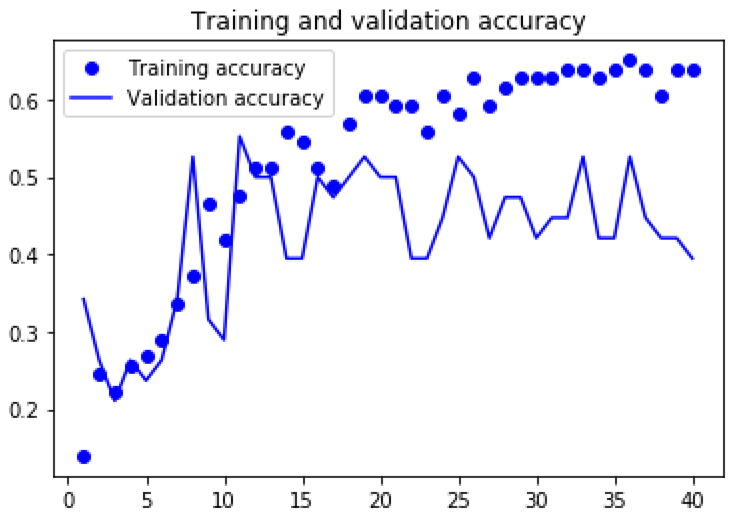
CNN Physionet dataset GridSearch



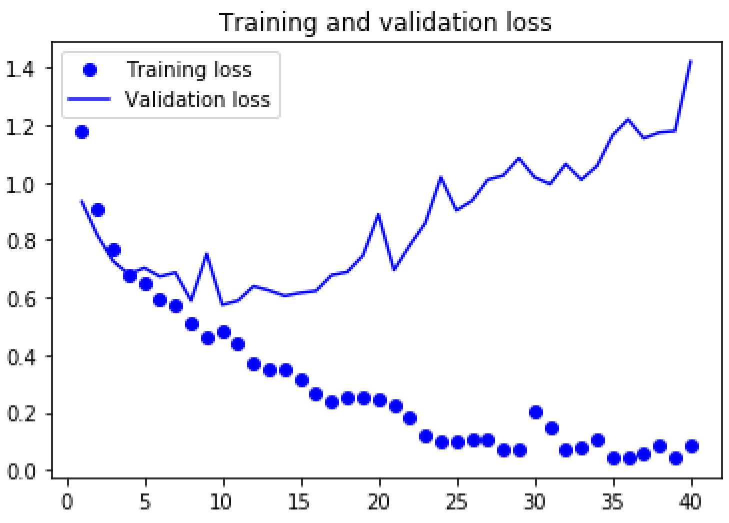


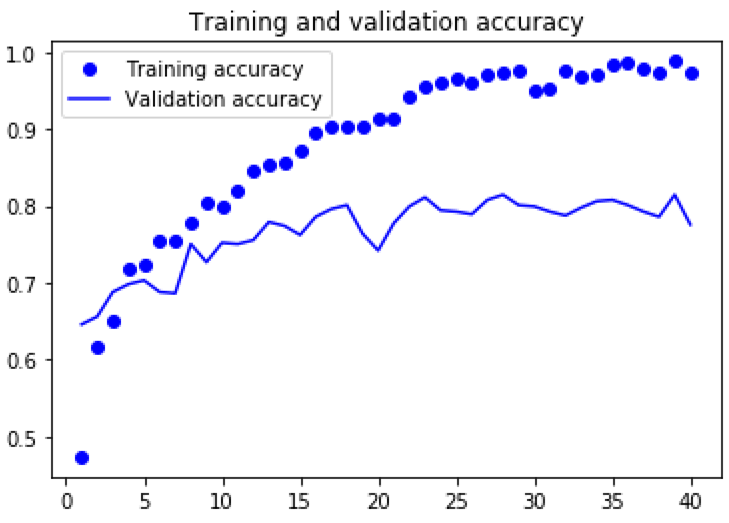
FFN Kaggle dataset



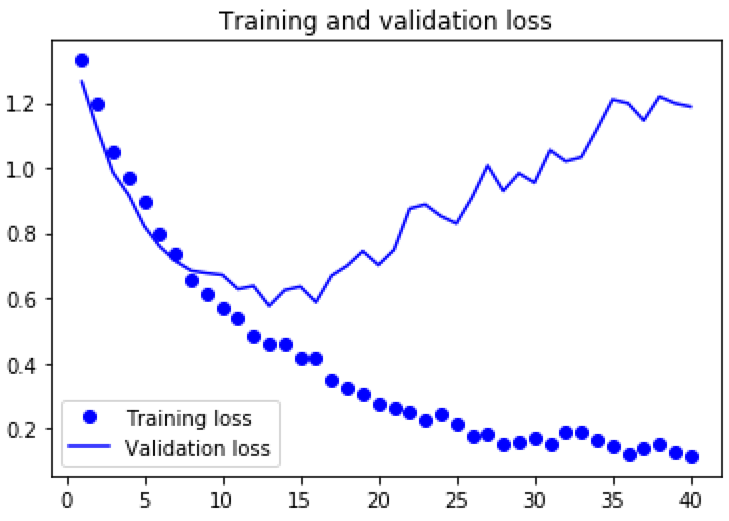


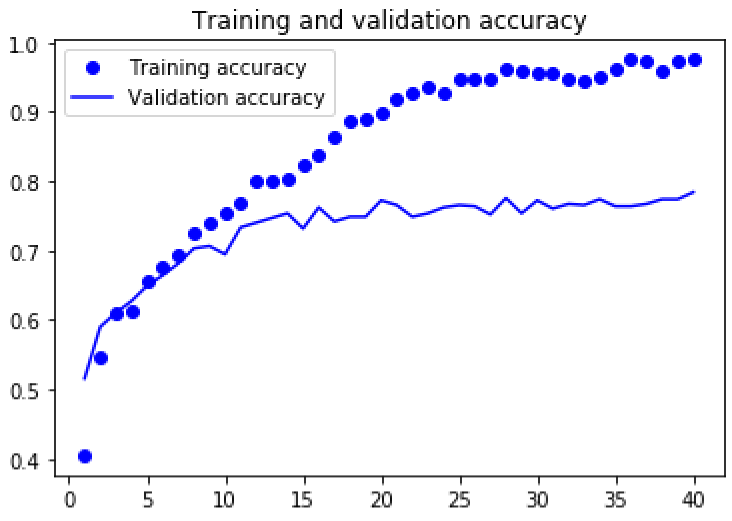
CNN Kaggle dataset





CNN Transfer Learning Kaggle dataset





Reflection

This project started as a personal interest in the area of healthcare and machine learning. An interesting dataset was found in the Kaggle platform related to heartbeats classification. The problem was that the dataset provided was too small and the first thought to come to mind was to use transfer learning to overcome this problem.

The most challenging part and also probably the one that had most direct impact to the results besides the architecture of the networks was the preprocessing of the files. Surfing the web and finding adequate implementations and then adapting them to the problem at hand required a lot of research and time.

Even if the results obtained at the end surpassed the benchmarks established, the end results of using transfer learning were not that different from not using transfer learning. But, as explained in the results section, if one thinks that the datasets used come from different sources and different types of recordings being able to obtained such results demonstrates the power of transfer learning.

Improvement

The simplest way to make improvements, of course, is to use more data. Besides data another challenge present with the dataset was the variety of recordings. Some contained a lot of noise. Probably a different approach could be to first apply a filter to remove noise from the audio files and just concentrate on low frequencies that most heartbeats may fall into.

Because of a limit on resources not many parameters were fine tuned but definitely different kinds of architectures may end up in better results.

References

https://brilliant.org/wiki/backpropagation/
http://karol.piczak.com/papers/Piczak2015-ESC-ConvNet.pdf
http://aqibsaeed.github.io/2016-09-24-urban-sound-classification-part-2/
https://www.physionet.org/challenge/2016/
https://en.wikipedia.org/wiki/Heart_sounds
https://www.kaggle.com/kinguistics/heartbeat-sounds
http://www.peterjbentley.com/heartchallenge/

http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/
https://en.wikipedia.org/wiki/Spectrogram
https://physionet.org/challenge/2016/papers/
https://en.wikipedia.org/wiki/Mel_scale
https://en.wikipedia.org/wiki/Spectrogram
https://physionet.org/challenge/2016/papers/rubin.pdf
https://physionet.org/challenge/2016/papers/goda.pdf
http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/