Machine Learning Engineer Nanodegree

Capstone Project

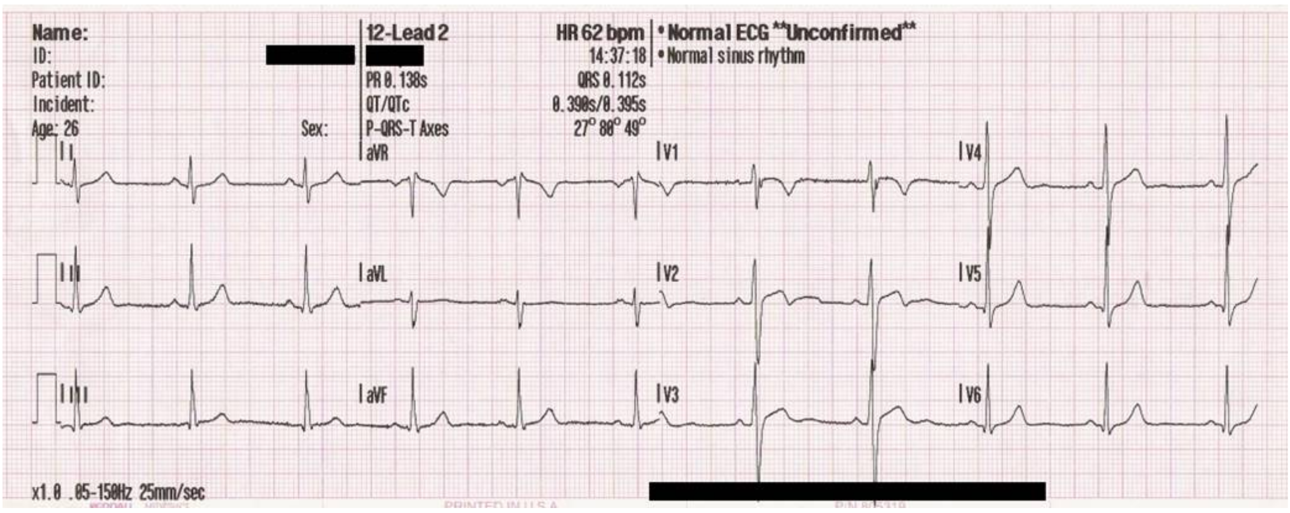
Using artificial neural networks to localize and classify heartbeats in ECG

Piotr Bazan
August 2nd, 2017

I. Definition

Project Overview

Electrocardiography or ECG is the process of recording the electrical activity of the heart over period of time. Measurement is done using electrodes placed on various parts of the skin and is often represent as a graph 1.



Plot 1: ECG graph

An ECG graph consists of series of patterns and is fundamental for understanding the electrical conduction system of the heart. Normal conduction starts and propagates in a predictable pattern, deviation from this pattern can be a normal variation or be pathological.

ECG dates back to 1901 when Willem Einthoven invented first practical electrocardiograph and assigned letters to the waveform [1]. He also described the electrocardiographic features of cardiovascular disorder and won Nobel Prize in Medicine in 1924 for his discovery. For many years ECG required careful manual annotation of patterns in the graph. During computer era first programs were created to automatically annotate diagrams. These programs marked patterns but with some degree of precision and the results required manual correction. With the development of artificial neural networks new approaches were proposed and better results obtained.

First category of solutions use domain knowledge to process signal and then learn neural network e.g. the research [2] uses specialized filters to clean ECG signal. Also authors use specialized algorithm to detect QRS patterns and create segments of data.

Another approach [3] is to decompose ECG signal into time-frequency representations and based on that calculate statistical features which are fed into neural network.

An interesting approach [4] is to avoid transformation into time-frequency domain and heavy signal post-processing. Authors conclude that previous attempts do not perform well due to the inter-patient variations of the ECG signals. They usually do not generalize well and have an inconsistent performance when classifying a new patient's ECG signal and have high variations in their accuracy and efficiency for larger datasets. This leads researchers to use common data as well as patient specific data to learn 1D convolution neural network.

All those approaches have some drawbacks:

- they require advanced prior knowledge about the subject domain
- involve heavy signal processing and signal transformation
- do not generalize well for new/unseen data

In this project I proposed new approach to heartbeat classification and localization. The data was not processed by any advanced filter nor it was transformed to time-frequency domain. Using two types of networks: artificial neural networks and convolution networks and lots of data I managed to achieve state of the art results.

The motivation for doing this project is to help people diagnose their heart. As there are more and more cheap ECG devices anyone can buy such a gadget. Most of them produce ECG diagram but apart from that they do not provide any useful information. Creating a model which could analyze and detect some abnormal beats could be beneficial.

Problem Statement

This is a classification problem, the goal of which is to localize and classify 2 types of beats: normal and atrial premature in ECG diagram. Also the project aims at avoiding previous research drawbacks:

- do not use any domain specific knowledge to extract features, nor clean signal using advanced filters
- do not use time-domain transformation and heavy signal post processing
- learn neural network on one group of data and validate on other group of data

As a result potential solution should have the following advantages:

- fast inference since there are no advanced signal processing
- learning features from data and not imposing domain knowledge
- better generalization for new data

Metrics

The project uses accuracy, precision, recall and f1 score as a metric.

Accuracy is the number of correct predictions divided by the total number of predictions:

$$acc = \frac{correct\ predictions}{number\ of\ predictions}$$

This is a general metric but it has some drawbacks when there are skewed classes. This is the case in this project where dataset has much more normal beats then premature atrial beats.

Precision is the number of True Positives divided by the number of True Positives and False Positives:

$$precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Precision can be thought of as a measure of a classifier exactness i.e. high precision means low number of false positives. This metric is important in case of arrhythmia beats – we would like to avoid warning patient unnecessarily.

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives:

$$recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Recall can be thought of as a measure of a classifiers completeness i.e. high recall means low number of false negatives. This metric is important because arrhythmia occurrences are rare and we would like to avoid miss-classifying those beats.

F1 score is a metric that combines both precision and recall and can be thought as balance between these two:

$$f_1 score = \frac{2 * precision * recall}{precision + recall}$$

In particular f1 score puts equal importance on both precision and recall.

II. Analysis

Data Exploration

The project uses MIT-BIH Arrhythmia Database [5] which contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 47 subjects studied by the BIH Arrhythmia Laboratory. This dataset is published under license [6] and consist of two types of files:

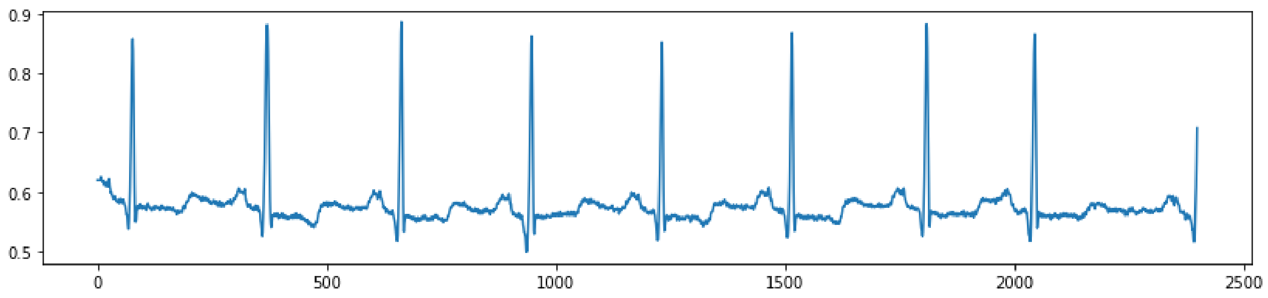
- signal data in PhysioBank specific binary format - .dat files

- annotation data - .atr files

This dataset requires downloading and compiling PhysioNet software called WFDB Applications. The signal files have the following data:

sample	MLII	V5
0	995	1011
1	995	1011
2	995	1011

First column is a sample identifier, other columns are ECG signal values. I used MLII as this signal is the most common in the dataset. Data is sampled at 360Hz frequency. Plot 2 shows example of ECG signal normalized to range [0;1].

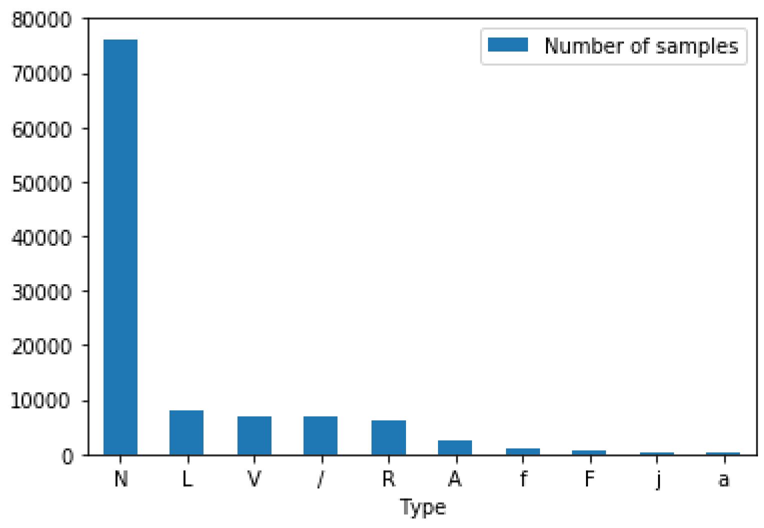


Plot 2: Normalized ECG signal

Annotation files have the following data:

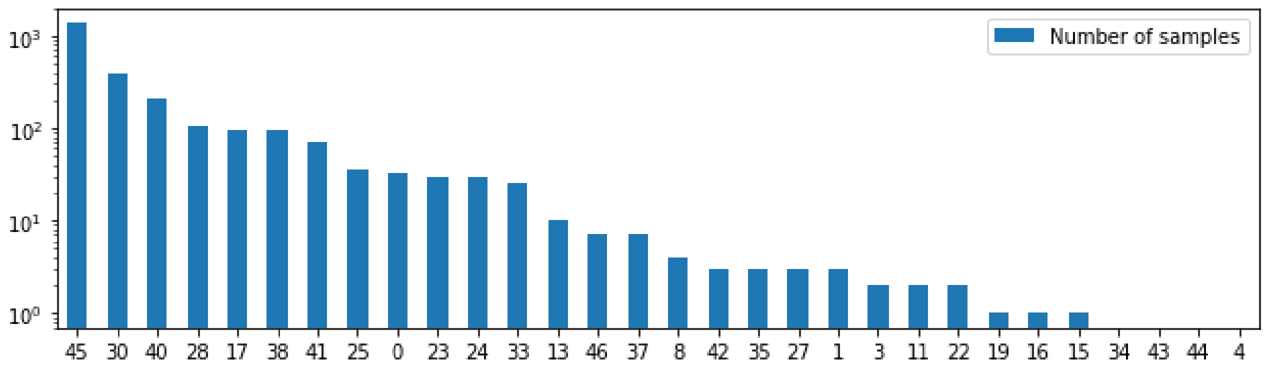
Time	Sample	Type	Sub	Chan	Num	Aux
0:00.050	18	+	0	0	0	(N
0:00.214	77	N	0	0	0	
0:01.028	370	N	0	0	0	

The point of interest is Sample column which allows matching signal with annotation. Type column contains one of the annotation type described [7]. In my research two types are investigated: ‘N’ - normal beat, ‘A’ - atrial premature beat. Plot 3 shows distribution of various beat types.



Plot 3: Number of samples per beat type

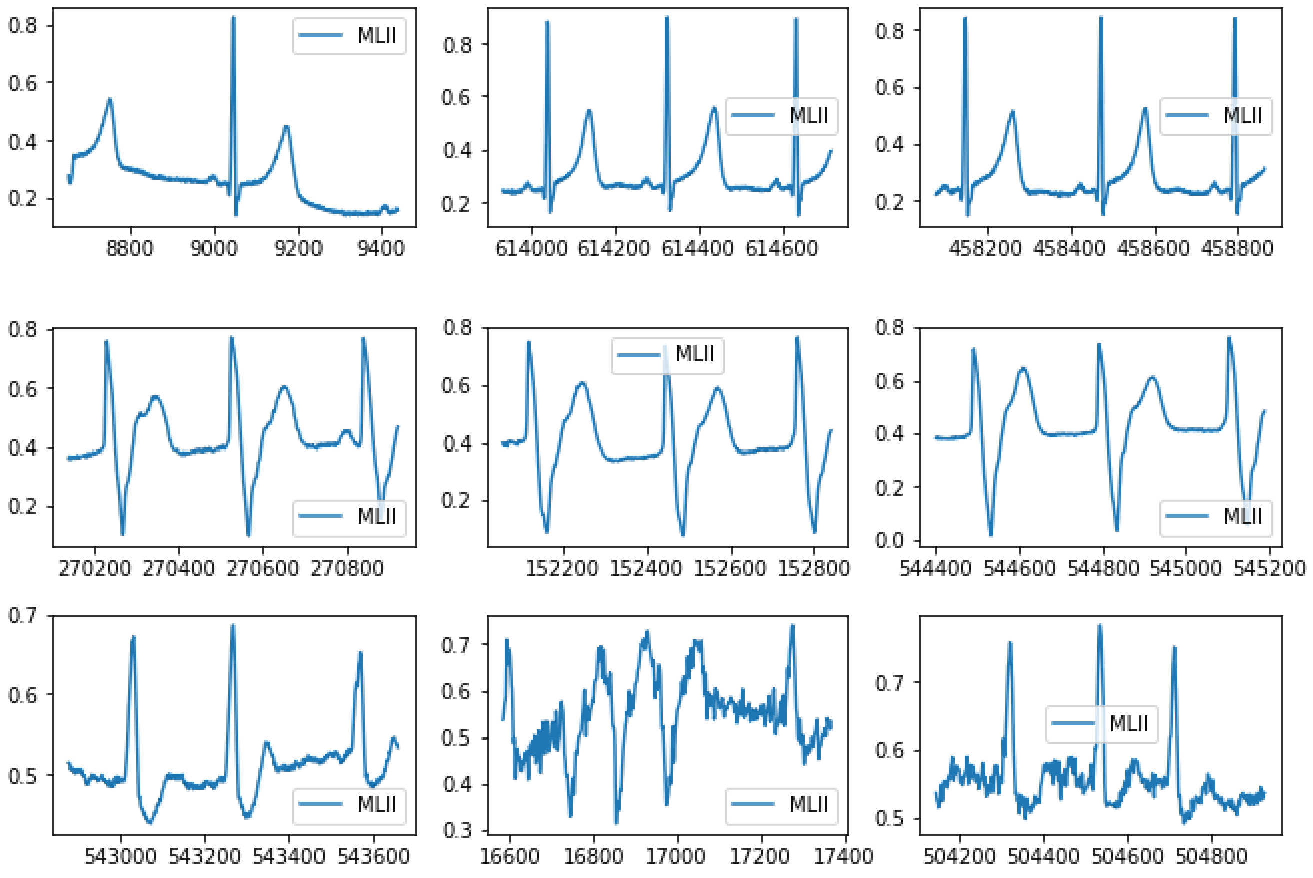
As can be seen class ‘A’ beat is much smaller than ‘N’ beat. This skewed dataset requires careful splitting for train/test dataset. Plot 4 shows distribution of ‘A’ beats per patient in log scale. There are few patients with hundreds of arrhythmia beats, others much less, sometimes in the order of dozens or couple. From the population we can conclude that atrial premature beats are rare.

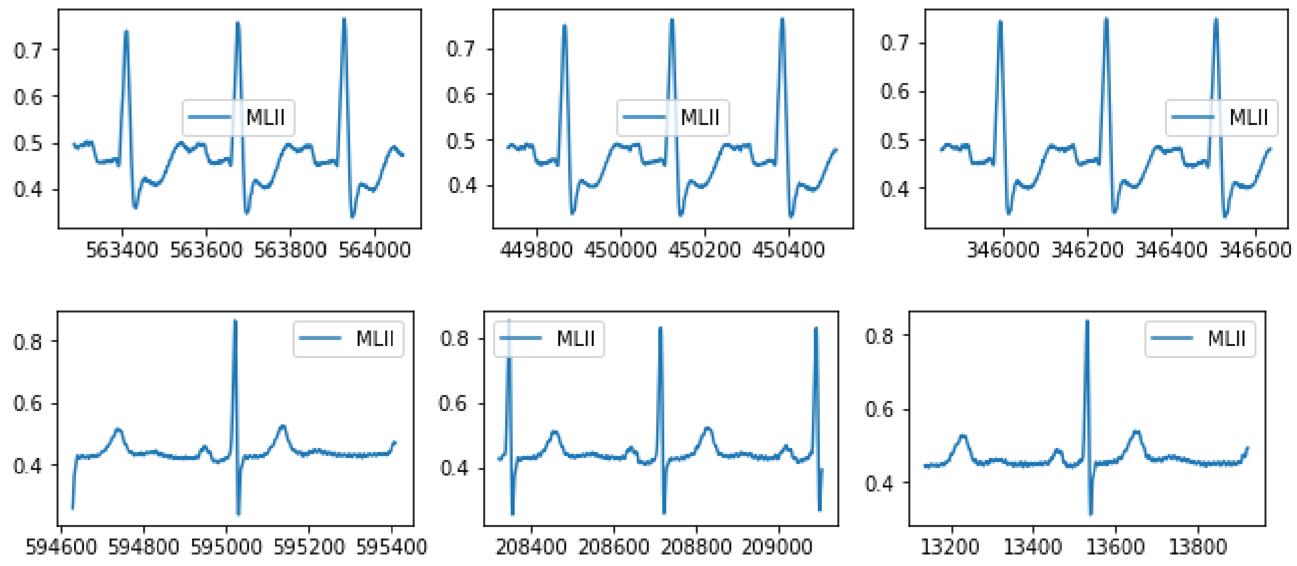


Plot 4: Number of A beats per patient data file (some patients with 0 values were omitted to make the plot more clear)

Exploratory Visualization

I assume dataset contains correct beat annotations. In order to visualize the form of the signal for different patients and to verify annotations I plotted random ECG samples. Each row contains 3 diagrams for a patient and each diagram is a frame of 784 signal values. In the center is a beat. X-axis shows sample id, y-axis signal value. All values have been normalized to [0;1] range.





Plot 5: Random ECG diagrams. Each row contains data for different patient.

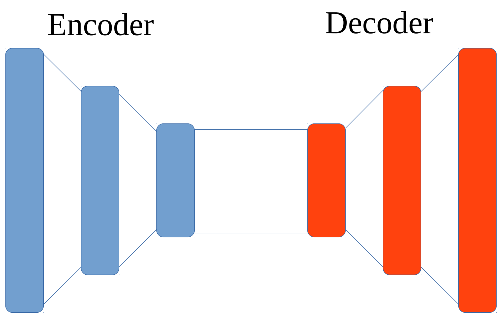
As it turns out there is much variability between patients and sometimes even within data for given patient like in 3rd row.

Algorithms and Techniques

ECG diagrams have high variability between patients and sometimes even within a particular person. Taking into account that model should not only recognize N (normal) and A (atrial premature) beats but also non beats i.e. shifted frames of N, A beats learning neural network could be tricky.

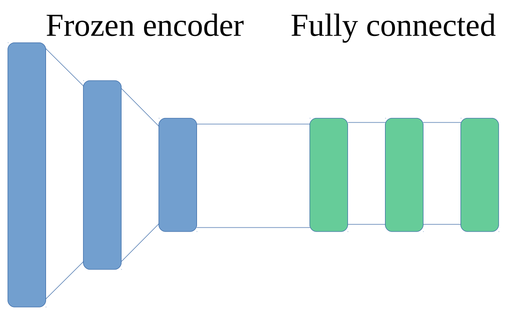
I came up with the following idea:

- train neural network and/or convolutional autoencoder [8] using unsupervised data. The idea behind this is that as the size of the layers decreases in encoder part also amount of information lowers. Then this information is passed to decoder were reverse process takes place. Input and output of the network contribute to a loss function which is used to back propagate gradient and learn network. This architecture 1 allows compressing signal, extracting important features and diminishing noise



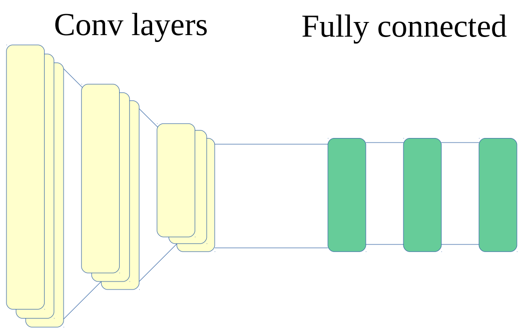
Drawing 1: Autoencoder

- in next step split autoencoder into encoder and decoder. The former would be frozen, fully connected layers attached and model would be trained in a supervised way. The decoder would be discarded



Drawing 2: Model consisting of encoder and fully connected layers

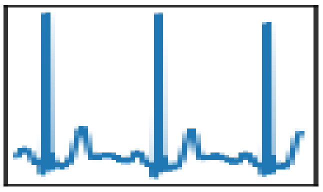
Other idea would be to use Convolution Neural Network [9] model: at the top convolution layers and then attached fully connected ones. Model would be trained in supervised way end to end. CNN have been successfully applied to analyzing visual imagery. They have filters also known as patches – small areas which detect simple features like lines, dots, shapes in the lower parts of the network and combine this information to detect more complicated patterns in upper parts. The most interesting thing is that CNN can learn the patterns itself having enough data. In case of ECG signal CNN could learn patterns which constitute to different heartbeats.

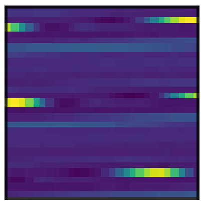


Drawing 3: CNN model

Those two approaches would be checked on smaller dataset. Based on the result top solution would be chosen and trained on full dataset.

Research [4] used 1D convolutions. I would like to use 2D convolutions but instead of mapping ECG frame as 2D image (first dimension signal, second time) I would like to take 784 values i.e. 1D signal and reshape it to 28x28 i.e. 2D signal. This would allow improving performance of learning and inference. Convolutional networks have nice property of scale and translation invariance. Also filters can learn complex patterns in deeper layers. Looking at the reshaped ECG signal CNN could take advantage of it.


ECG beat frame


Reshaped ECG 28x28

Presented ideas require different strategy when searching for optimal network architecture. Models with autoencoder could be investigated in the following way:

1. pick autoencoder configuration.
2. train autoencoder in unsupervised way. Input and output is a beat frame.
3. split autoencoder. Freeze encoder part, discard decoder
4. pick fully connected layer configuration. Attach to encoder and train network in supervised way. Input is a beat frame, output is a beat label
5. record metrics for such configuration
6. if there are any fully connected configurations to check go to point 4. Otherwise, if there are any autoencoder configuration to check go to point 1

On the contrary CNN model could be investigated in the following way:

1. pick configuration of the whole network: convolution and fully connected
2. train in supervised way. Input is a beat frame, output is a beat label
3. record metrics for such configuration
4. if there are any configurations left to check go to point 1

Benchmark

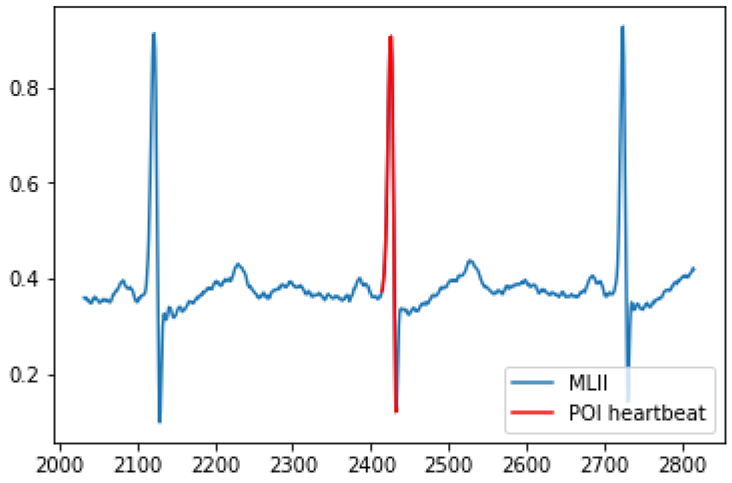
Research [2], [3], [4] classify heartbeats and provide accuracy as a metric. They however either use advanced signal processing, time-frequency transformation, domain prior knowledge or patient specific data. On the contrary this project aims removing this disadvantages. On top of that it also tries to localize heartbeats by moving sliding window and running classifier.

The authors of [2], [3] report accuracy: 88.76%, 96,94%, the authors of [4] report accuracy, precision and recall for subset of beats. It is hard to compare this results to my project especially provided I have no knowledge regarding cardiology. Instead I plan to use Dummy Classifier [10] which is a classifier that makes predictions using simple rules and is useful as a simple baseline to compare with other (real) classifiers. It can have several strategies. In this project I chose “stratified” which generates predictions by respecting the training set’s class distribution.

III. Methodology

Data Preprocessing

Each signal file is normalized to [0,1] range. Then data is split for heartbeats: normal and arrhythmia. Specifically for each heartbeat in annotation file we get Sample identifier. Given this id we look into signal file and subsample range of 784 MLII signal values centered at id. Given signal frequency of 360Hz and 784 values we have a frame spanning over 2 seconds which consists of a given beat and at least 2 adjacent beats. Example frame is shown on plot 6. During this process we generate labels for a beat type.

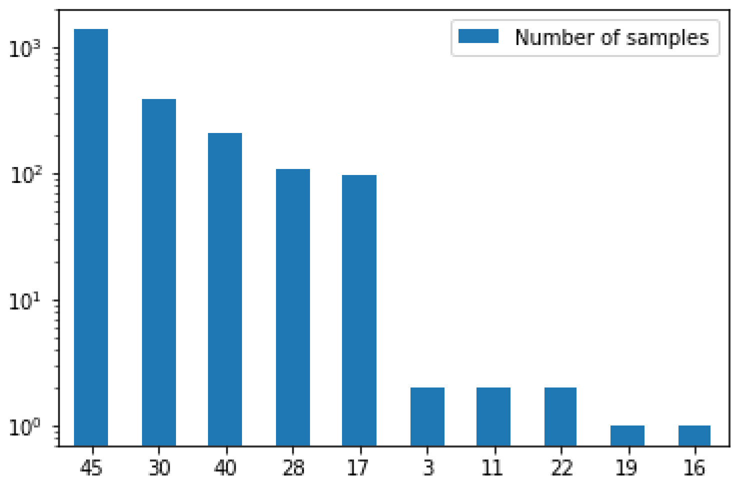


Plot 6: Example heartbeat frame.

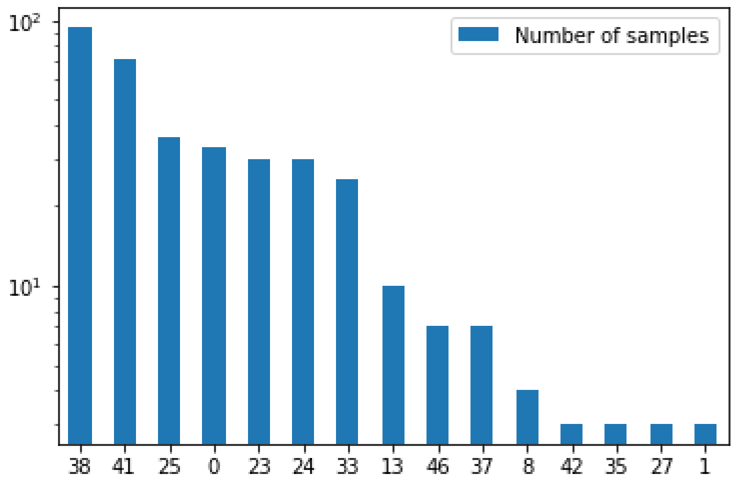
Apart from that we shift the window forward or backward by random value between 6-135 points which corresponds to 16-375ms to generate non beat frames. Also we generate labels of type non beat. This type of data allows model to distinguish between a beat and non beat which will be used for localization in stream of signal data.

Rest of the columns from signal and annotation files is discarded. Some files had to be ignored as they did not have MLII signal or caused parsing issues.

Due to skewed dataset regarding atrial premature type beat I decided to split dataset in the following way: train data will contain most ‘A’ beats from fewer patients – plot 7 whereas test data will contain less ‘A’ beats but from more patients – plot 8. This allows having sufficient data for training and more diverse data for testing and validation.



Plot 7: Training dataset: A beats per patient id



Plot 8: Test dataset: A beats per patient id

It is worth noting that train and test dataset do not intersect i.e. there are from different patients.

Train and test dataset are split. Due to performance reasons and skewed dataset it requires some trade-offs: 10% is used for different model evaluations, 90% is left for top model training and validation. Table 1 shows dataset split.

	Train	Test	Final training	Final validation
A	247	39	2549	360
N	7949	3121	79794	30660
NB	8272	3044	82343	31020

Table 1: Dataset size for different beat types

Implementation

In order to facilitate building and evaluation of various model architectures several source code files were crated.

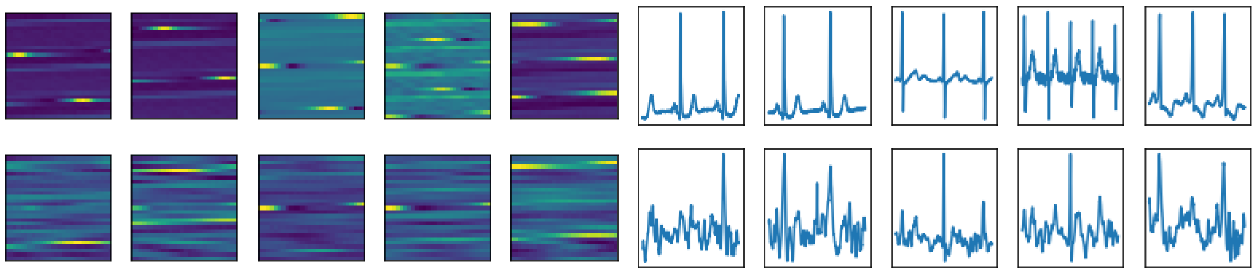
- Preprocess.py contains functions to load dataset files [5] using pandas library. It removes unnecessary columns, normalizes signal values. Such preprocessed data is saved in python pickle format and used on consecutive jupyter notebook runs. There is also a function which creates heartbeat frames of various types.
- Layers.py contains code to build neural and convolutional layers. I decided to use Keras library – it contains high level API and supports both Theano and Tensorflow backend. In this project I used the latter as it has great support for GPU computations.
- Models.py contains code to build models. It uses layers.py for creating layers and it connects them to form different architectures like encoder, decoder, autoencoder and CNN. It also has functions to train this models. Both models and layers are highly parametrized to be able to create architectures depending on provided configuration.
- Evaluate.py contains functions to run experiments on different models. They require configuration as an input and then check different model architectures. They also calculate and record metrics.
- Plots.py constains various plotting functions used throughout the project.
- Utils.py contains helper functions that deal with file paths, reshaping arrays, saving/loading model.

While doing this project I encountered several difficulties. The first one was related to creating autoencoder in such a way that it could be split for encoder, decoder so that they share weights. Moreover encoder is later joined with fully connected layer to form different model. I also tried to parametrize it so that different architecture could be created.

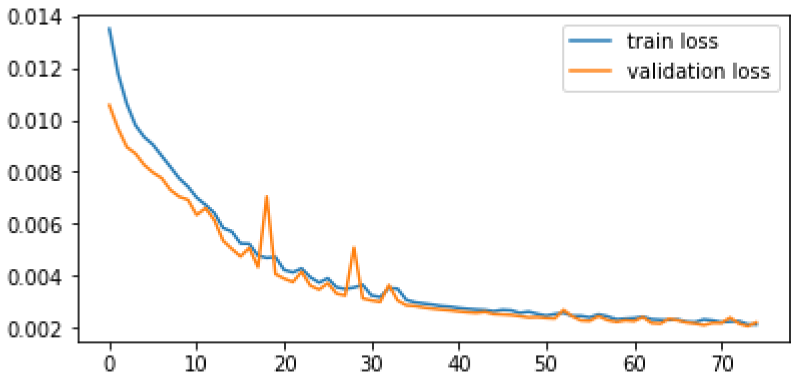
Another problem was related to gradual building the project and evaluation of different architectures especially given the performance issue I had. I implemented several checkpoints – places in the code where model weights can be saved and later loaded. Such spots include results of training autoencoder, fully connected later on top of it or the full sequential CNN model. This allowed having good pace and interactive research and saved literally dozen of hours.

Refinement

At the beginning I was not sure if the autoencoder would be capable of learning any useful features or to clean signal especially given two factors: learning beats and its shifted forms as well as using convolutional layers on reshaped 2D ECG signal. To have better insight into quality of the AE I plotted several signals both as images and diagrams before and after AE transformation. Plot 9 shows example result. As can be seen AE learned to represent signal and some of its features.

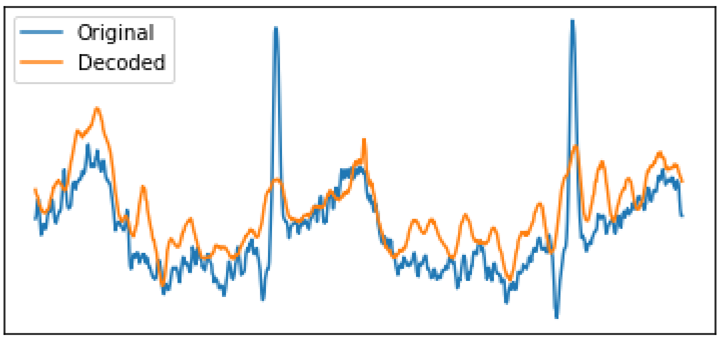

Plot 9: ECG signal as images (left) and diagrams (right). Top row - original signal, bottom - decoded

In order to be able to quantify the quality of an autoencoder configuration I plotted its loss for train and validation where the latter is from separate dataset.



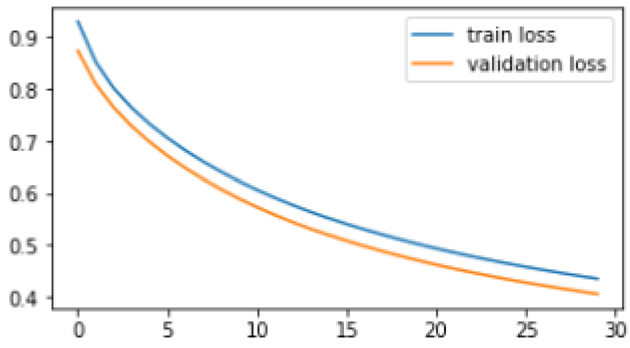
Plot 10: Autoencoder loss. Along x-axis epochs.

Also I plotted more detailed sample of encoded-decoded signal. Plot 11 shows an example.



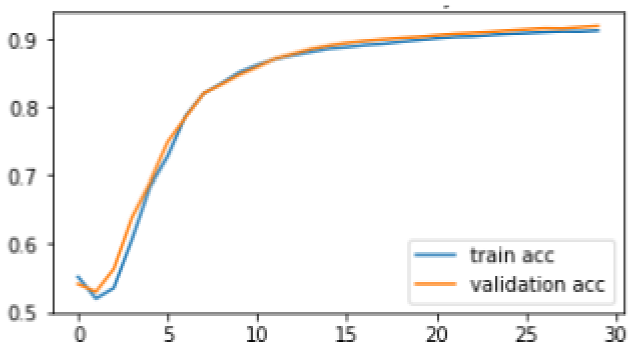
Plot 11: ECG original and decoded signal

After training autoencoder, fully connected layers were attached and model was trained. The learning process was represented by its loss function (12).

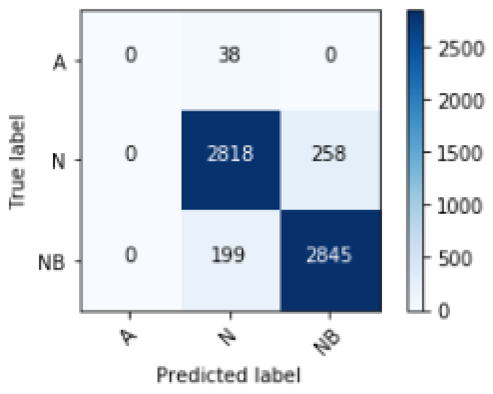


Plot 12: Full model loss. Along x-axis epochs.

To visualize metrics I plotted accuracy (13) and confusion matrix (14).



Plot 13: Full model accuracy. Along x-axis epochs.

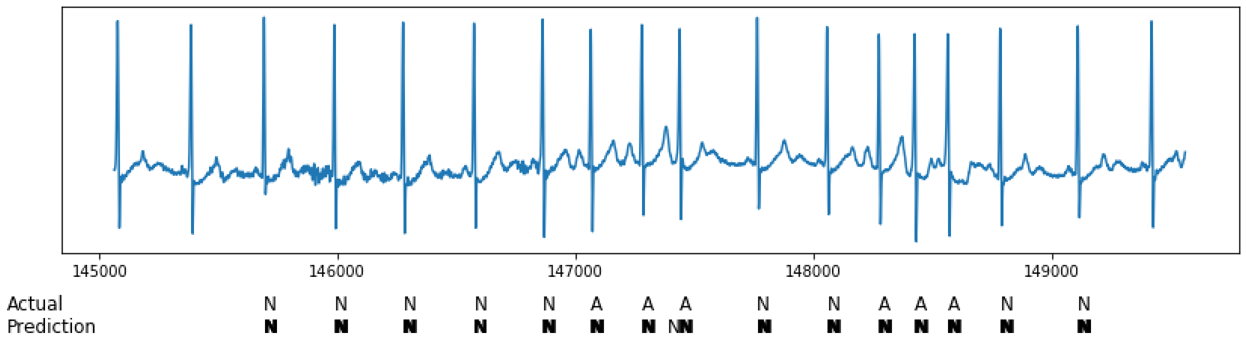


True label \ Predicted label	A	N	NB
A	0	38	0
N	0	2818	258
NB	0	199	2845

Plot 14: Confusion matrix

Confusion matrix presents true positives, false positives and false negatives for each beat type. In the example (14) all ‘A’ beats were classified as ‘N’ beats – these are false negatives. There are no false positives for this beat type.

To check how the model acts in terms of heart beat localization I created a diagram (15) which shows ECG spanning over several beats. At the bottom there are correct annotations, below the predictions.

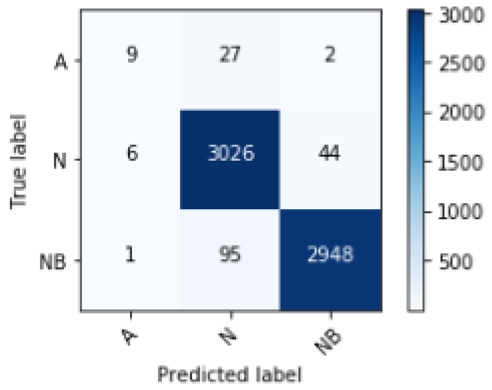


Plot 15: ECG validation diagram.

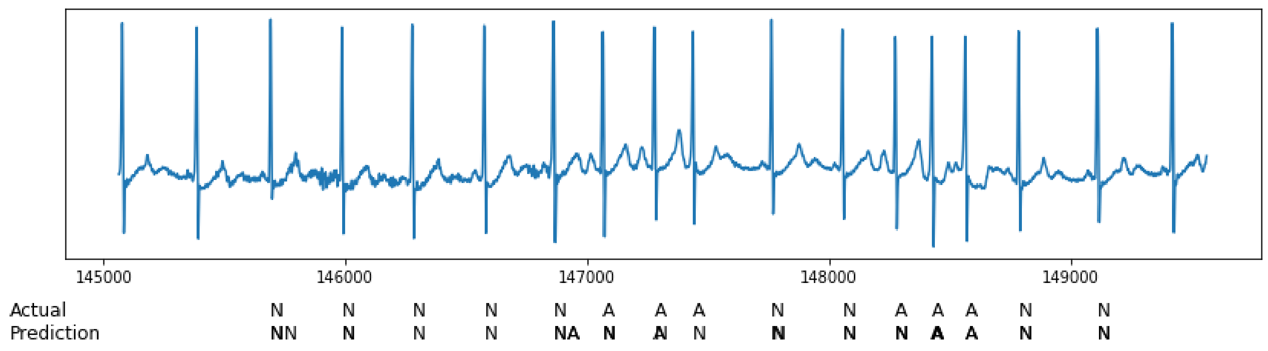
The predictions were made by extracting beat frames of 784 values, each shifted by value of 2 i.e. 5.5 ms and then fed to the model to make predictions. As can be seen this particular model predicts beat at moment of the original beat and around it – that is way there is a bold letter **N**. This stems from the fact that non beats were generated by shifting a given beat frame by 6-135 points.

All presented plots relate to the simplest NN autoencoder: 784 input neurons, 128 output neurons and on top very simple FC layer with 3 neurons. More detailed models were investigated. Still none of the NN autoencoder was capable of correctly classifying 'A' type beats. This was understandable given the fact that 'A' beats are 1% of all other types.

Next convolutional autoencoders were investigated. I was surprised when I saw that models started correctly classifying premature atrial beats - plot 16 . Also CNN networks started to localize precisely the moment of a beat.



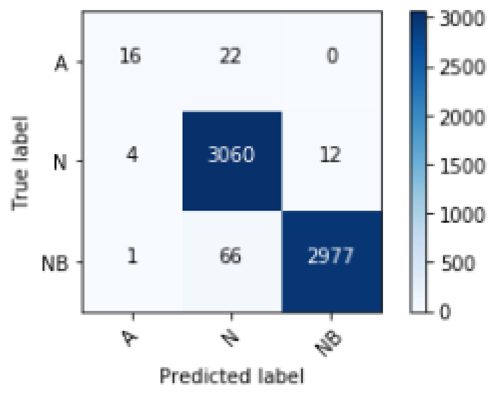
Plot 16: Confusion matrix. AE configuration [32, 16, 16], FC configuration [1024,3]



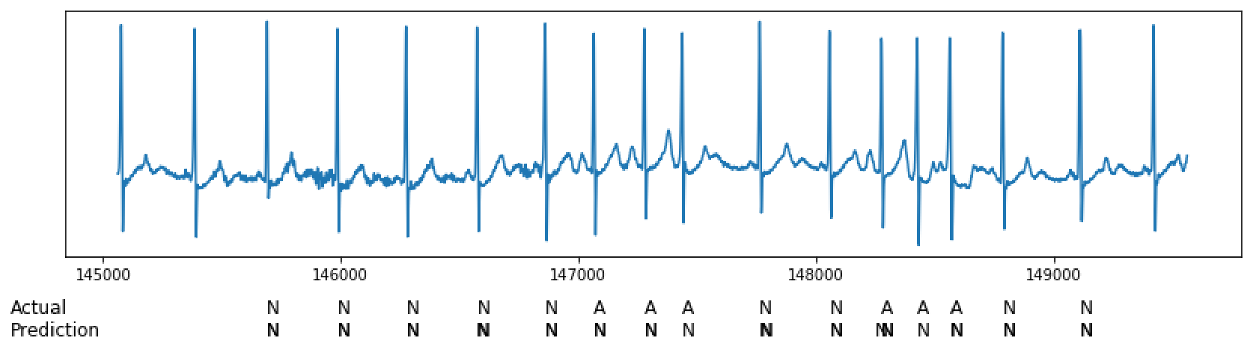
Plot 17: ECG validation diagram. AE configuration [32, 16, 16], FC configuration [1024,3]

This network was build with the following building blocks: convolution layer followed by pooling layer 2x2 followed by batch normalization [11]. The convolution layers have 32, 16, 16 filters respectively. On top there is the FC layer with 1024 neurons followed by layer with 3 neurons.

However the best results were obtained using CNN sequential network. All metrics were improved. Also training such model is quicker than a model with autoencoder.



Plot 18: Confution matrix. Configuration filters: 16, 32, 64, FC 2048, 3, dropout: 0.5



Plot 19: ECG validation diagram. Configuration filters: 16, 32, 64, FC 2048, 3, dropout: 0.5

This particular CNN was created with building block of: convolution layer, pooling layer 2x2, batch normalization. Each block had 16, 32, 64 filters respectively. Next there was a dropout layer [12] attached with ratio 0.5. On top there are two fully connected layers with 2048, 3 neurons.

The following tables present obtained results during various models evaluation. In bold best results in given family of architectures.

Autoencoder configuration	FC configuration	Validation accuracy	Precision	Recall	F1 score
[784, [], 128]	[3]	0.92	0.61	0.62	0.61
[784, [], 128]	[64, 3]	0.95	0.63	0.64	0.63
[784, [], 128]	**[128, 3]**	**0.95**	**0.63**	**0.64**	**0.64**
[784, [256, 128], 64]	[3]	0.94	0.63	0.63	0.63
[784, [256, 128], 64]	[64, 3]	0.94	0.63	0.63	0.63
[784, [256, 128], 64]	[128, 3]	0.95	0.63	0.64	0.63

Table 2: NN autoencoder + FC layers metrics

Autoencoder configuration	FC configuration	Validation accuracy	Precision	Recall	F1 score
[16, 8, 8]	[3]	0.94	0.63	0.63	0.63
[16, 8, 8]	[64, 3]	0.96	0.74	0.68	0.70
[16, 8, 8]	[128, 3]	0.97	0.89	0.72	0.75
[16, 8, 8]	[1024, 3]	0.97	0.84	0.73	0.76
[32, 16, 16]	[3]	0.95	0.64	0.64	0.64
[32, 16, 16]	[64, 3]	0.97	0.84	0.72	0.75
[32, 16, 16]	[128, 3]	0.97	0.86	0.73	0.76
[32, 16, 16]	**[1024, 3]**	**0.98**	**0.87**	**0.79**	**0.82**

Table 3: Metrics of models: CNN autoencoder + FC layer

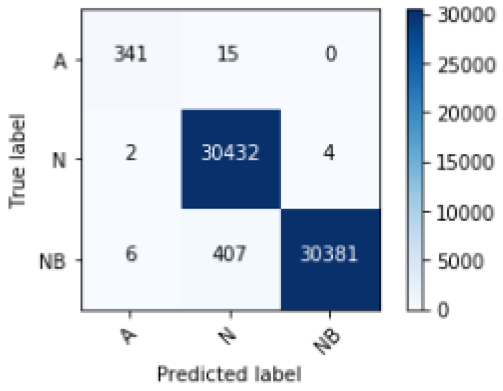
CNN filters	FC units	Dropout	Validation accuracy	Precision	Recall	F1 score
[4, 8, 16]	[1024, 3]	0.5	0.98	0.91	0.80	0.84
[8, 16, 32]	[1024, 3]	0.5	0.98	0.92	0.75	0.80
[16, 32, 64]	**[1024, 3]**	**0.5**	**0.99**	**0.92**	**0.82**	**0.85**
[16, 32, 64]	[2048, 3]	0.5	0.98	0.91	0.80	0.84

Table 4: Metrics of CNN sequential models

IV. Results

Model Evaluation and Validation

Based on the previous research outcome the CNN model with 16, 32, 64 filters was selected. It has two fully connect layers of 1024, 3 neurons. Dropout is 0.5. This model was trained on full train set and validated on full test set. Number of epochs was steadily increased until loss function reached plateau. The obtained results are astonishing (20).



Plot 20: Confusion matrix of the final model

The model had only 8 false positives and 15 false negatives for ‘A’ type beat. Is also scored very high in all metrics shown in table 5.

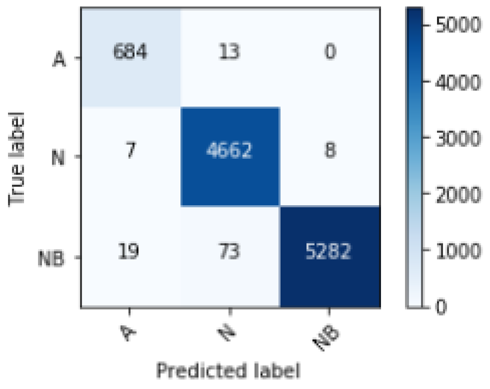
CNN filters	FC units	Dropout	Validation accuracy	Precision	Recall	F1 score
[16, 32, 64]	[1024, 3]	0.5	0.99	0.99	0.98	0.98

Table 5: Final model metrics

Having seen this result I double checked if the test and train data do not intersect. Also I wanted to reproduce model achievements using totally different dataset. Three persons with total of 697 type ‘A’ beats were in validation set, rest of patients were in training set. Table 6 shows beats distribution. I evaluated model and obtained similar high metrics (21, 7).

Beat type	Train	Validation
A	2542	697
N	79257	4677
NB	81799	5374

Table 6: Alternative dataset

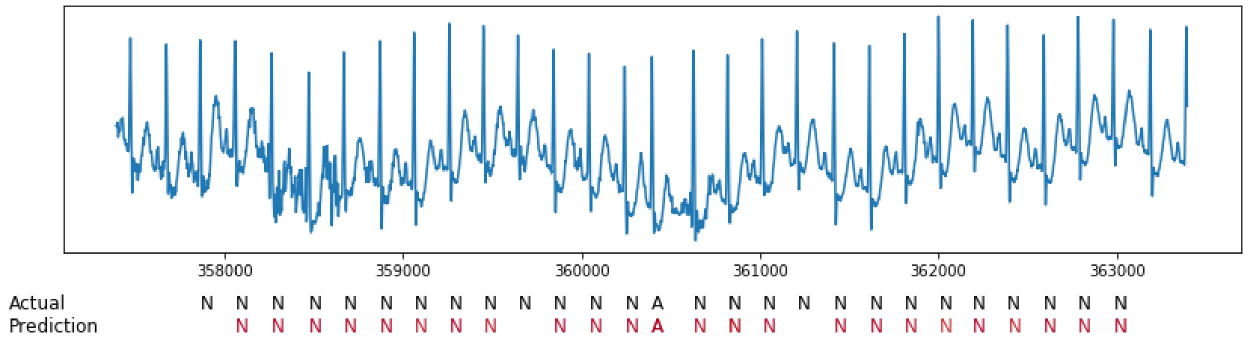


Plot 21: Confusion matrix for alternative dataset

CNN filters	FC units	Dropout	Validation accuracy	Precision	Recall	F1 score
[16, 32, 64]	[1024, 3]	0.5	0.99	0.98	0.99	0.98

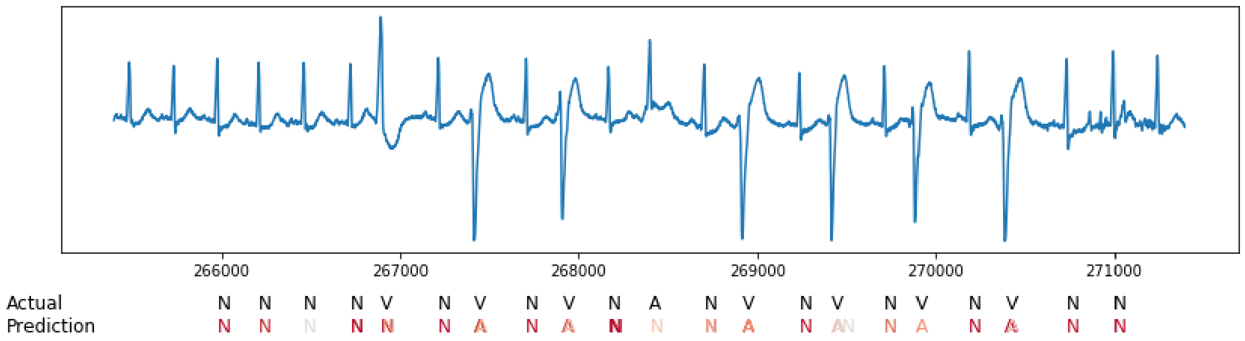
Table 7: Final model metrics on alternative dataset

In order to see how the model behaves in terms of prediction and classification I used ECG validation diagrams similar to previous one but beat type was colored according to prediction probability. Strong red means probability closer to 1, faint red means probability closer to ½. Plot 22 shows ECG with known beat types. Model performed very well misclassifying only 3 'N' beats.



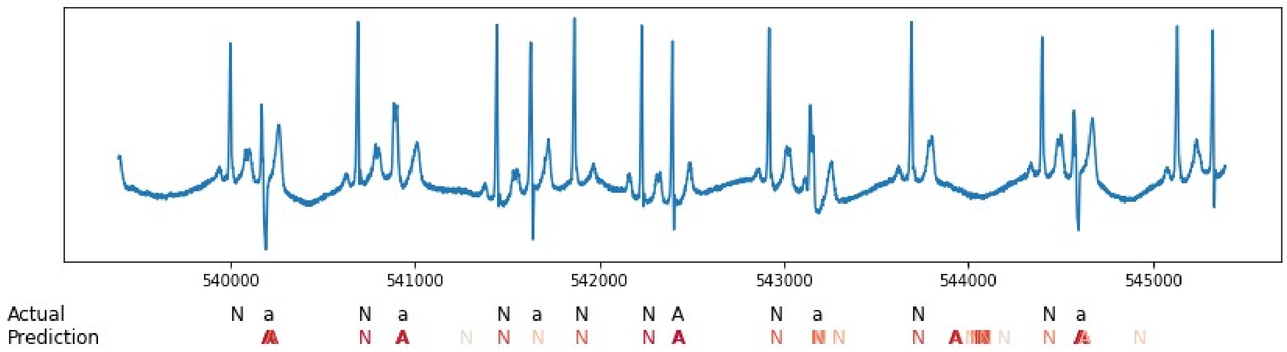
Plot 22: ECG validation diagram for 'N' and 'A' beats

Plot 23 shows ECG with 'V' type beat which is unknown to the model. This type of beat is mostly classified as 'A'. All 'N' beats are correctly localized and classified.



Plot 23: ECG validation diagram with unknown beat type 'V'

Plot 24 shows highly irregular ECG data and some unknown beats. Model correctly localizes true 'N' beats but it has some false positives in regions where the signal has curvy U shape.



Plot 24: ECG validation diagram with highly irregular data

Justification

The final model metrics are extraordinary. For reference point table 8 presents dummy classifier metrics, table 5 final model metrics.

Validation accuracy	Precision	Recall	F1 score
0.35	0.33	0.34	0.27

Table 8: Dummy Classifier metrics

The model deals very well classifying unseen data. It only struggles during localization task, where it has to classify unknown beat types.

V. Conclusion

Reflection

I found this project really challenging. I decided to use several ideas at once which no one has done before like no signal cleaning/transformation, squashing 1D signal into 2D for CNN. Also I shifted heartbeat frames to learn model to localize beats. At the beginning I was not sure if autoencoder will learn anything useful. I had to gradually create environment in which I could make experiments and validate different parts of the network and different architectures.

What surprised me the most was the fact that CNNs are able to localize heartbeats with such precision. Initially I thought that CNN autoencoder with FC layers will be optimal. I was surprised to find that CNN sequential model performed even better. It is also much quicker.

The other thing that astonished me was the fact that CNN is able to detect ‘A’ type beats with such precision and recall. It is only 1% of data. Interesting is also the fact that the solution is able to generalize prediction for unseen data.

Lastly I was amazed when I saw top model in action in ECG validation diagrams. Overall I found this project highly rewarding.

As a side note I would like to share my experience with the problem of performance especially of convolution autoencoders. My laptop does not have Nvidia GPU so all calculations are done on CPU. I tried running code on Google Compute Engine equipped with GPU but there are extra charges applied and one has to remember to shutdown virtual machines when they are no used. I found many opinions suggesting decent, modern GPU with at least 4GB RAM is required for doing experiments with CNNs. Still I was curious so I checked workstation with Nvidia GTX 750 GPU with 2 GB of ram and 512 cuda cores. It turned out it supports cudnn and works pretty well with Tensorflow-gpu. It is slightly slower than Google Compute Engine but 20 times faster than my laptop’s core i5. The bottom line is for simple CNN such configuration will do.

Improvement

The presented model could be extended to classify also other heartbeat types. It would be beneficial to have better dataset though as some beats are really rare in terms of frequency or occur only in small number of patients.

1 https://en.wikipedia.org/wiki/Electrocardiography#History

2 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.495.4675&rep=rep1&type=pdf

3 http://www.sciencedirect.com/science/article/pii/S0031320304002766

4https://www.researchgate.net/profile/Serkan_Kiranyaz/publication/285493884_Convolutional_Neural_Networks_for_Patient-Specific_ECG_Classification/links/565e999f08aeafc2aac90822/Convolutional-Neural-Networks-for-Patient-Specific-ECG-Classification.pdf

5 https://physionet.org/physiobank/database/mitdb/

6 ODC Public Domain Dedication and License v1.0

7 https://www.physionet.org/physiobank/annotations.shtml

8 https://en.wikipedia.org/wiki/Autoencoder

9 http://cs231n.github.io/convolutional-networks/

10 http://scikit-learn.org/stable/modules/model_evaluation.html#dummy-estimators

11 https://arxiv.org/abs/1502.03167

12 https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf