duanlingxiao1219@sina.com
https://askwitioniary.github.io

Contribution Title*

Lingxiao Duan, Zihan Jiang, and Shengsheng Zhuang

[1] Washington University in St Louis
[2] Peking Union Medical College Hospital

Abstract. With the development of computing power marching on rapidly, machine learning and deep learning has begun to shine in the past decade. As a tool, it has been applied in various circumstances making our lives much easier and sometimes even fancier. In China, a country with one fifth of the population on earth, we only have 1.8 [**?**] licensed doctors among 1000 population. We want to free our doctors from redundant and repeating work so that on one hand, they can spend more time on patients with more complicated conditions and on the other hand, more doctors can dedicate into research so that we can hopefully cure more diseases in the future. In this work we focus on 8 different abnormal Electrocardiography(ECG) changes. We are able to not only distinguish normal ECGs with abnormal ones, but also label each abnormal ECGs with all the abnormalities it has, with an averaged accuracy of nearly 90% using machine learning and/or deep learning methods.

Keywords: Automated ECG labeling · Machine Learning · Deep Learning

1 Before we start

I would like to talk a little bit about the scope of our work before we dive in. Like I mentioned above, we are only dealing with 8 abnormal ECG changes, as shown below,

Table 1. List of abnormal ECG changes and abbreviations

Name	Abbrev.
Atrial Fibrillation	AF
First-degree atrioventricular block	FDAVB
Complete right bundle branch block	CRBBB
Left anterior fascicular block	LAFB
Premature ventricular contraction	PVC
Premature atrial contraction	PAC
Early repolarization	ER
T wave change	TWC

* Supported by Liangma data co., Ltd.

along with the normal category. It is obvious that comparing with more than 154 [**?**] of ECG abnormalities, this is just a tip of the iceberg. However, there is no so-called "golden tool" for any problem. The key of solving scientific problems is to always talk business in a confined "environment". We must be aware that the general problem will only be possible to solve when its sub-problem is solvable.

2 Getting to know our problem

2.1 Data description

Thanks to Tsinghua University and the data provider, we are able to get our hands on 6500 labeled 12-lead ECGs as training and validating data, including I, II, III, aVR, aVL, aVF, V1, V2, V3, V4, V5, V6 along with gender and age information of the patient. The data is in the form of time series while each data point represents the voltage of the timestamp. The sampling frequency is fixed at $500Hz$. The length of each data entry is variable while the length of 12 leads in one data entry is the same. Below is a length distribution of all 6500 data. We can see that most ECGs have length between 4000 to 6000 inclusively while there is no very short($<= 3000$) entries but some very long ones(> 10000).**(todo: get a graph of the length distribution and justify the statement)** The class distribution of the training data is as follows. Note that this statistics counts multi-labeled data more than one times. Say we have a case with abnormalities 3 and 4, then it is counted in both class 3 and 4.

```
count
```

```
{'T波改变 TWC': 2156,
 '一度房室传导阻滞 FDAVB': 534,
 '完全性右束支传导阻滞 CRBBB': 826,
 '室性早搏 PVC': 654,
 '左前分支阻滞 LAFB': 180,
 '心房颤动 AF': 504,
 '房性早搏 PAC': 672,
 '早期复极图形改变 ER': 224,
 '正常 Normal': 1953}
```

Fig. 1. Class count of training data

2.2 Problem description

The goal of this problem is to come up with a model using the provided 6500 training data to label an ECG that lies within the 8 categories mentioned above with all of its abnormalities, or of course normal if one has none.

2.3 A deeper look

Knowing the fact that most abnormal ECG changes can co-exist, we came up with 3 different methodologies to solve this problem.

- Single-label classification problem of every possible combinations of 8 abnormalities
- Multi-label classification/labeling problem
- Multiple 2-classes classification problem

However each data entry can theoretically have 1 to 8 labels(neglecting abnormalities that are naturally contradicting), making possible class pool huge($2^8 + 1 = 257$). The provided data only has 107 different combinations and more than half of which has less than 5 data entries. Therefore treating each combination as a separate class is not wise.

For multi-label classification, we not only have the problem of training data hardly spans half of the possibilities, the very uneven distribution of each class can also be problematic.

Since this is a real world problem and we have been using human intelligence to solve this problem for decades, it is hard to be ignorant about the experience doctors built and the rules researchers came up with. After all, we are dealing with a tiny subset of the real problem: labeling 8 out of more than 100 ECG changes[**?**]. Thus we finally used the last methodology, treating the problem as 8 two-class classification sub-problems, because we want to be able to easily meddle with our techniques for each different abnormalities so that we can add human experience and knowledge to our model.

3 Preprocessing

Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn [**?**]. When we as human-beings look at information around us, we subconsciously "preprocess" the information and come up with a conclusion. This is why we can read text in various fonts and easily capture objects from a very noisy picture. Our model on the other hand, have very limited ability to extract key information from data so it is tremendously important for us to preprocess our data before feeding it into our model.

3.1 Detecting QRS complex

There are many open source tools and algorithms available online for detecting QRS complex. It is the most important preprocessing of data in our work since we will be using ECG data of just one beat(one complete cycle of heart beat). We will also try to find R point of each beat to calculate heart rate, to determine Arrhythmia as well as roughly locating P segment, ST segment and T wave.

Here we used the offline version of an open source toolkit on github [**?**] as a base project and modified it to suit our need.

First off we changed the structure of input data. The original project requires both voltage and timestamp data as inputs, however the timestamp data is actually redundant. Thus I modified it to accept 1 dimensional time series data.

Secondly I noticed the toolkit is using *butter* and $lfilter$ method from *sklearn.signal* module to filter out high and low frequency noise from the raw data. I think our primal goal here is to accurately find R point rather than give a nice looking ECG, so I built a slider tool to test and find out the best parameters for high and low frequency cutoff values to magnify wave features in our ECG, i.e. amplitude and slope. **todo: post a screenshot for the simple slider tool and the high, low cut of the final band-pass filter**

The tool then calculates the discrete derivative of the signal and take the square of it. At last it calculates the integral of the result to find out the steepest small interval and thus roughly find out where the QRS complex is. In other words, it is calculating the energy of the electricity of equally spaced small intervals, and in most cases if not all, the QRS complex will have way more energy than other segments of the ECG.

After we get the integrals, aka. energy levels of the lead in the ECG, we use a peak detection method [**?**] using Pan-Tomkins algorithm [**?**]. At this point we have a list of possible R points calculated based on "intervals of the most energy". However, due to the fact that the QRS complexes are very short and depending on how we slice those small intervals, the point we get may not lie right on R point, but on either QR segment or RS segment instead. **get a graph of result not right on R points**.

At least we have a somehow accurate interval, i.e. we know where the QRS complexes are, just not the actual R point. This is good enough for our purpose though. We can find the maximum/minimum point of the small segments near the R points in the result (note that aVF lead is generally drew upside down) and find the minimum/maximum point before and after the previous point to get possible Q,R,S points.

The efficiency of this process is quite good. **todo: give an example of how long it takes to process a certain number of data**

3.2 Slicing ECG based on beats

We obtained a list of QRS complex locations(indexes) in the previous step. We still want to cut the full ECG based on pulses. Note that for each data entry, we can obtain 12 different sets of R points. Given the fact that data in 12 leads are

taken simultaneously, their R points should align pretty good. Though we really only need one set to do the splitting, it is such a waste to disregard the others. What we did was to obtain R points of all 12 leads using the above mentioned technique, then find the most common length within these 12 sets of R points. For instance, if the resulting R points of 12 leads

$$R = \begin{matrix} 23\ 447\ 902\ 1342\ 1898\ 2256\ 2712\ 3123\ 3589 \\ 449\ 904\ 1345\ 1897\ 2252\ 2713\ 3125 \\ 448\ 903\ 1344\ 1898\ 2253\ 2716\ 3124 \\ 445\ 900\ 1340\ 1897\ 2255\ 2714\ 3121 \\ 444\ 900\ 1341\ 1898\ 2254\ 2712\ 3119\ 3592 \\ 441\ 898\ 1338\ 1895\ 2253\ 2712\ 3120 \\ 447\ 902\ 1340\ 1897\ 2255\ 2714\ 3125 \\ 445\ 900\ 1342\ 1897\ 2255\ 2714 \\ 444\ 903\ 1340\ 1898\ 2258\ 2712\ 3121 \\ 445\ 901\ 1340\ 1895\ 2255\ 2714\ 3121 \\ 443\ 901\ 1342\ 1897\ 2253\ 2714 \\ 21\ 445\ 900\ 1340\ 1897\ 2255\ 2714\ 3121 \end{matrix}$$

has length $length = [9, 7, 7, 7, 8, 7, 7, 6, 7, 7, 6, 8]$, we will take $l_{most\_common} = 7$ as the most common length. In reality the above matrix won't align so nicely like this, so we need to find and eliminate outliers ourselves. We take the leads with the most common length, namely the II, III, aVR, aVF, V1, V3, V4 and use the average of each column as the "golden rule". We then find the outliers by looping though all other leads and calculate the minimum difference between each R index and every R index in the golden rule. For example, in lead I, the very first R index 23 is **not close to** any index in the golden rule, thus it is an outlier and needs to be removed. As for the ones that are shorter than the "golden rule", we find out where(which pulse) the it is missing and simply insert with the average of the "golden rule" of that pulse. Now we should have a $NUM_{pulse} \times 12$ matrix that is perfectly shaped so we simply calculates the average of each column to obtain the ultimate R point indexes of the case.

We are now able to locate where each QRS complex is, but it is still hard to accurately locate P and T wave of each pulse because many of abnormalities will result in significant changes with P and T wave. To achieve our goal efficiently, we simple split each pulse 50 points after the middle point of two continuous R indexes. Because P wave is generally shorter than ST-T segment. For example, [449 904] will have splitting points at $\lfloor \frac{449+904}{2} + 50 \rfloor = 726$

This method can eliminate noises that occur in a subsets of leads assuming leads without noise still dominates the data entry. However it still has its limitation. It does not help with the case where every single lead is affected by significant noise. **For example, add the case in the testing data set with tremendous noise**

Here is a real result after applying the technique. Showing the number of pulses, i.e. the length of detected R indices of cases **where the number of pulses detected in 12 leads are not the same**. We can see that there are

34 out of the first 300 cases does not have consistent number of pulses detected, while the most common number of pulses technique works for all of them. Thus it performs relatively stable and is giving us what we need.

```
([15, 16, 16, 16, 15, 16, 16, 16, 16, 16, 16, 16], 'TRAIN101.mat')
([10, 10, 11, 10, 10, 11, 10, 11, 10, 10, 10, 10], 'TRAIN110.mat')
([11, 11, 12, 12, 11, 12, 12, 12, 10, 10, 11, 12], 'TRAIN112.mat')
([13, 15, 15, 16, 15, 15, 15, 15, 15, 15, 15, 15], 'TRAIN116.mat')
([11, 12, 11, 11, 11, 12, 11, 12, 12, 12, 11, 11], 'TRAIN118.mat')
([10, 11, 4, 11, 11, 4, 11, 11, 11, 11, 11, 11], 'TRAIN120.mat')
([12, 12, 12, 12, 12, 12, 11, 12, 12, 12, 12, 12], 'TRAIN121.mat')
([13, 12, 13, 13, 13, 13, 14, 14, 14, 1, 13, 13], 'TRAIN123.mat')
([12, 12, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12], 'TRAIN128.mat')
([10, 10, 10, 10, 10, 10, 11, 10, 10, 10, 10, 10], 'TRAIN130.mat')
([10, 11, 11, 10, 11, 11, 11, 11, 11, 10, 10, 10], 'TRAIN142.mat')
([13, 1, 1, 1, 1, 1, 3, 10, 13, 13, 13, 13], 'TRAIN158.mat')
([13, 13, 13, 13, 9, 13, 12, 13, 13, 13, 13, 13], 'TRAIN176.mat')
([9, 9, 8, 9, 8, 9, 9, 9, 9, 9, 9, 9], 'TRAIN179.mat')
([10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11], 'TRAIN181.mat')
([13, 13, 13, 13, 13, 7, 14, 14, 14, 14, 13, 13], 'TRAIN184.mat')
([15, 15, 15, 14, 2, 15, 15, 15, 14, 15, 15, 15], 'TRAIN186.mat')
([10, 11, 3, 11, 10, 11, 10, 10, 10, 11, 11, 10], 'TRAIN197.mat')
([12, 12, 12, 8, 4, 12, 12, 10, 12, 12, 12, 12], 'TRAIN198.mat')
([12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13], 'TRAIN201.mat')
([12, 12, 12, 12, 11, 12, 12, 12, 12, 12, 12, 12], 'TRAIN208.mat')
([8, 7, 9, 8, 8, 7, 8, 8, 8, 8, 8, 8], 'TRAIN218.mat')
([13, 13, 13, 13, 12, 13, 13, 13, 13, 13, 13, 13], 'TRAIN225.mat')
([13, 13, 13, 14, 13, 14, 15, 14, 14, 14, 13, 14], 'TRAIN233.mat')
([14, 13, 14, 13, 14, 13, 14, 14, 13, 14, 13, 13], 'TRAIN237.mat')
([10, 10, 13, 10, 13, 13, 13, 13, 13, 13, 13, 10], 'TRAIN246.mat')
([19, 19, 19, 19, 19, 17, 19, 19, 19, 19, 19, 19], 'TRAIN248.mat')
([9, 9, 3, 9, 9, 9, 9, 9, 9, 9, 9, 9], 'TRAIN253.mat')
([13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 12], 'TRAIN254.mat')
([10, 12, 12, 12, 12, 12, 12, 12, 11, 12, 12, 12], 'TRAIN258.mat')
([14, 13, 14, 13, 14, 14, 13, 13, 13, 13, 13, 13], 'TRAIN260.mat')
([4, 14, 14, 12, 5, 14, 14, 14, 14, 14, 14, 14], 'TRAIN283.mat')
([14, 14, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14], 'TRAIN285.mat')
([15, 14, 15, 14, 15, 15, 15, 15, 15, 15, 14, 14], 'TRAIN296.mat')
```

Fig. 2. Number of pulses in 12 leads

3.3 Enriching data set

This is yet another important technique to improve machine learning performance. Considering the labor and time cost of gathering training data, it is often not viable to gather data that can perfectly cover all possibilities of an abnormality. Human can learn how to distinguish one abnormality from another by comparing very few cases because we are very good at extracting key features from a case. However, this is comparably hard for machines. What we can do to help machines though is to create data that are different in irrelevant features while keeping the key features the same. For instance, if we were to write a program to determine whether a picture has apples in it, we only care about whether at least one apple exists, not its location, or color or orientation. Thus if we can create pictures with both green and red apples locating in every corner of the picture in various orientations, we can train a more general model to determine the existence of apples.

Here I used two basic technique to enrich or data. One is adding minor noise to the data and the other one is messing up the order of each pulse.

After consulting with cardiologist in our team, we decide that an noise ranging $[-0.05, 0.05]mV$ won't change the key features that are deterministic. Thus in addition to original training data, we created considerable amount of data with "minor noise".

Also, for most abnormalities, the order of the pulses doesn't really hurt the diagnoses much. With the ability to split each pulse in a case, I created more data by messing up the order of pulses. For example, a data entry is originally ordered like $\overbrace{pulse_1, pulse_2, ..., pulse_1 0}^{10 pulses}$. We can create a number of data that looks like $\overbrace{pulse_8, pulse_2, pulse_6, ..., pulse_k}^{10 pulses}$

4 Solving the problem

Finally we are in business. As we mentioned in the abstract, we used a number of machine learning/deep learning techniques along with some rule-based methods to label our data. Given the fact that all abnormal wave change are different, we used different models for each abnormality based on their characteristics. However, there are two models appeared in every single cases. I would like to talk about these two first.

4.1 Two base models

5000 Model As the name suggests, our input data has length 5000. In other words, we are modifying our variable length training data and make them all with the of the same length. We choose the length 5000 for the following reasons.

- Majority of data have length close to 5000

– We only have one label for each ECG, so 5000 is a good length to include sporadic abnormalities.

Thus we will be dealing with 3 scenarios explained below.

– $length_{original} = 5000$. This is the trivial case. We simply keep it what it is.
– $length_{original} < 5000$. In this case we append 0s to both the start and the end of the data to forcibly make it has length of 5000.
– $length_{original} > 5000$. This is the tricky case. We calculate the remainder of the length divided by 5000 and see if it is less than 1000.
  – If so, we cutoff the start and the end of the data by half of the remainder to make it dividable by 5000. Then simply divided the data into $\lfloor \frac{length_{original}}{5000} \rfloor$ pieces. The first and the last one second of an ECG are often noisy anyways.
  – If the remainder is greater than 1500 though, we calculate the starting indices of the split as follows An simple example would be a data entry of length 22000 will have starting points at $0, 4250, 8500, 12750, 17000$, or in other words, we will slice the original data into 5 pieces with length 5000 with ranges:

0	5000
4250	9250
8500	13500
12750	17750
17000	22000

Our inputs are now in the right shape. We still have concerns about the data distribution. Recall that we are building a 2 class classifier, we need to balance our data. Also note that we are not distinguishing one abnormality with only the normal ones, but every other data without that abnormality. This makes the number of negative data much more than the number of positive ones. Here we used data enrichment techniques we introduced above to populate positive data to match the size of negative data. We are not truncating negative data here because we want our model to be "familiar" with all possible cases that should not be classified positive.

Now that we have our inputs ready to roll, let's talk about the model. We mainly used $Conv1d$ layer from tensorflow based keras. The reason behind this is that our inputs are one dimensional time series. Our target features lie within the pattern of the time series. Since converting our data into images will result in most pixels being meaningless, we believe that $Conv1d$ would work as well as converting our data to images and using $Conv2d$ layers, if not better.

In general, we used Conv1D of depth 6 followed by fully connected Dense layer of depth 4 along with a output layer. Shown below is just an example of what the model looks like. We have tried a lot of different minor modifications based on how many original data we have for the specific abnormality and how long our target feature is.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 5000, 12)	0	
conv1d (Conv1D)	(None, 4998, 512)	18944	input_1[0][0]
batch_normalization_v1 (BatchNo	(None, 4998, 512)	2048	conv1d[0][0]
conv1d_1 (Conv1D)	(None, 4996, 256)	393472	batch_normalization_v1[0][0]
batch_normalization_v1_1 (Batch	(None, 4996, 256)	1024	conv1d_1[0][0]
conv1d_2 (Conv1D)	(None, 4994, 512)	393728	batch_normalization_v1_1[0][0]
max_pooling1d (MaxPooling1D)	(None, 1664, 512)	0	conv1d_2[0][0]
batch_normalization_v1_2 (Batch	(None, 1664, 512)	2048	max_pooling1d[0][0]
conv1d_3 (Conv1D)	(None, 1662, 256)	393472	batch_normalization_v1_2[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 554, 256)	0	conv1d_3[0][0]
batch_normalization_v1_3 (Batch	(None, 554, 256)	1024	max_pooling1d_1[0][0]
conv1d_4 (Conv1D)	(None, 552, 256)	196864	batch_normalization_v1_3[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 184, 256)	0	conv1d_4[0][0]
batch_normalization_v1_4 (Batch	(None, 184, 256)	1024	max_pooling1d_2[0][0]
conv1d_5 (Conv1D)	(None, 182, 256)	196864	batch_normalization_v1_4[0][0]
global_average_pooling1d (Globa	(None, 256)	0	conv1d_5[0][0]
dropout (Dropout)	(None, 256)	0	global_average_pooling1d[0][0]
dense (Dense)	(None, 256)	65792	dropout[0][0]
dense_1 (Dense)	(None, 128)	32896	dense[0][0]
dropout_1 (Dropout)	(None, 128)	0	dense_1[0][0]
dense_2 (Dense)	(None, 64)	8256	dropout_1[0][0]
dropout_2 (Dropout)	(None, 64)	0	dense_2[0][0]
dense_3 (Dense)	(None, 64)	4160	dropout_2[0][0]
input_2 (InputLayer)	(None, 3)	0	
concatenate (Concatenate)	(None, 67)	0	dense_3[0][0] input_2[0][0]
dropout_3 (Dropout)	(None, 67)	0	concatenate[0][0]
dense_4 (Dense)	(None, 1)	68	dropout_3[0][0]

Total params: 1,711,684
Trainable params: 1,708,100
Non-trainable params: 3,584

Fig. 3. 5000 Model Summary

We inserted normalized age and sex data into the Dense layer since there are some abnormalities that is more common in one gender/elder people and less in the opposite.

Pulse Model Some abnormalities, such as blocks can be found in every single pulse if one has it. We can have much more training data if we consider each pulse as a standalone data entry. Also, doing this can enable us to detect abnormalities in real time.

Here we used pulse splitting method mentioned above to split one case into pieces consisting data of one pulse only. To make the shape of our input data looks nice, we set a fixed length of 500. Pad 0s to the shorts and truncate the longs. We do this on the start and the end of each pulse. Thus the input shape for this model is 500×12.

Everything else besides length is pretty similar to the 5000 model above.

4.2 Abnormality specific tuning

Atrial Fibrillation (AF) Atrial Fibrillation, aka. AF is "characterized by high-frequency excitation of the atrium that results in both dyssynchronous atrial contraction and irregularity of ventricular excitation"[**?**]. The most significant characteristics of AF appear on ECG is that the amplitude of its P wave is very low. Its P wave consists of mostly high frequency low amplitude "trembles" [**?**]. Additionally, AF may not happen in every single pulse.

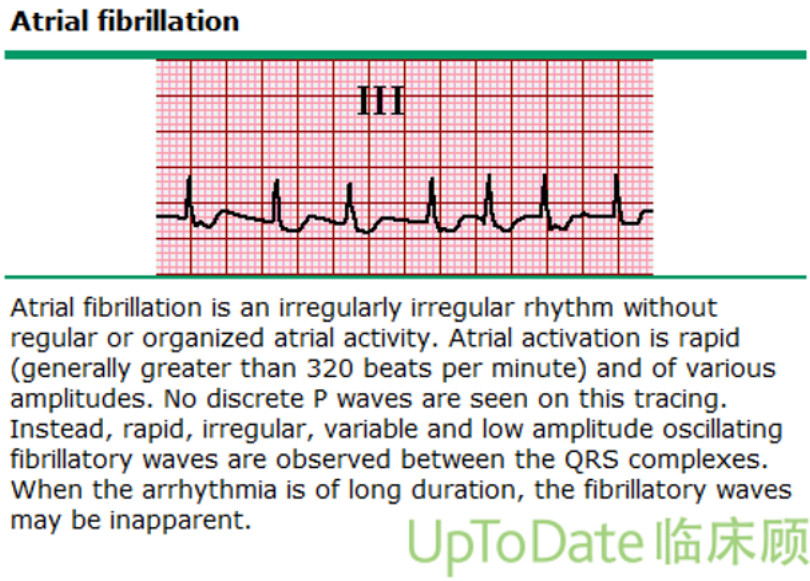


Fig. 4. Atrial Fibrillation. [**?**]

With that being said, we used the 5000 model as our base model and had some minor tweaks to it since the key feature, the P segment is relatively wide. We were able to achieve more than 97% 10 folds validation accuracy offline and got 0.9588% testing f1 score online.

First-degree Atrioventricular Block(FDAVB) This abnormality works pretty well with our base model so we did not make much modification or research deeper into it. The final f1 score for FDAVB is 0.9093%

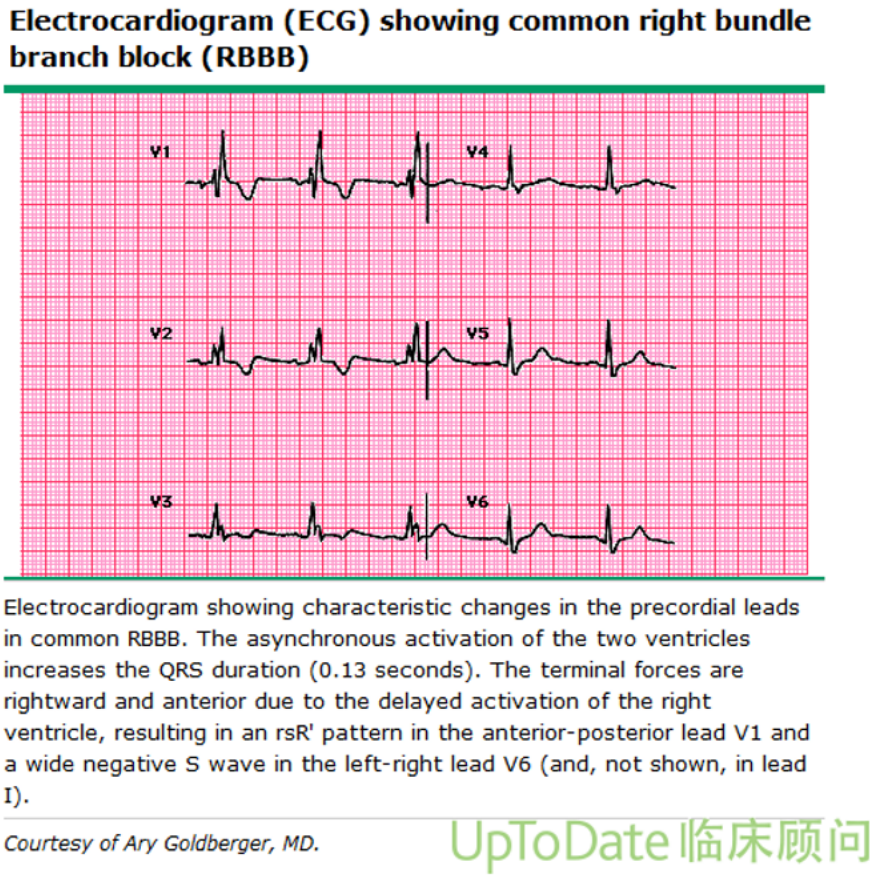


Fig. 5. Complete Right Bundle Branch Block [**?**]

Complete Right Bundle Branch Block(CRBBB) CRBBB has the most obvious feature with its ”M” shaped R wave. No other abnormalities have similar

feature like CRBBB. This might be the easiest among this subset of abnormal ECG changes. We got surprisingly f1 score of 1 on leader board a on our first submit with the base pulse model and final f1 score of 0.9539 on testing data. Offline validation accuracy was more than 99%

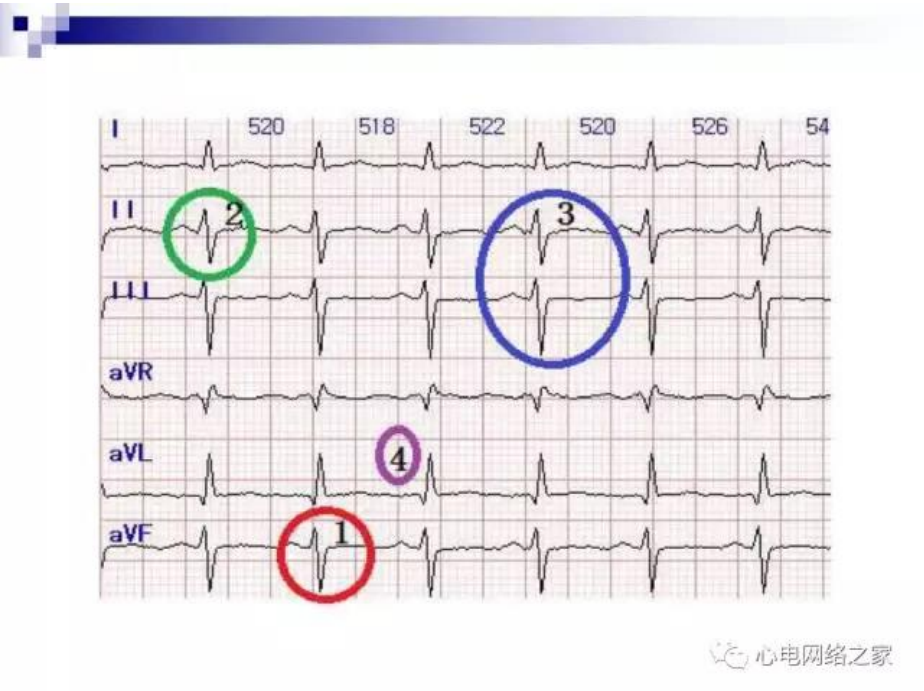


Fig. 6. Left Anterior Fascicular Block [**?**]

Left Anterior Fascicular Block(LAFB) LAFB is a tricky one because convolutional neural network is good at extracting "shape" features. In our case it would be "how voltage is changing". However, LAFB requires something else to be diagnosed.

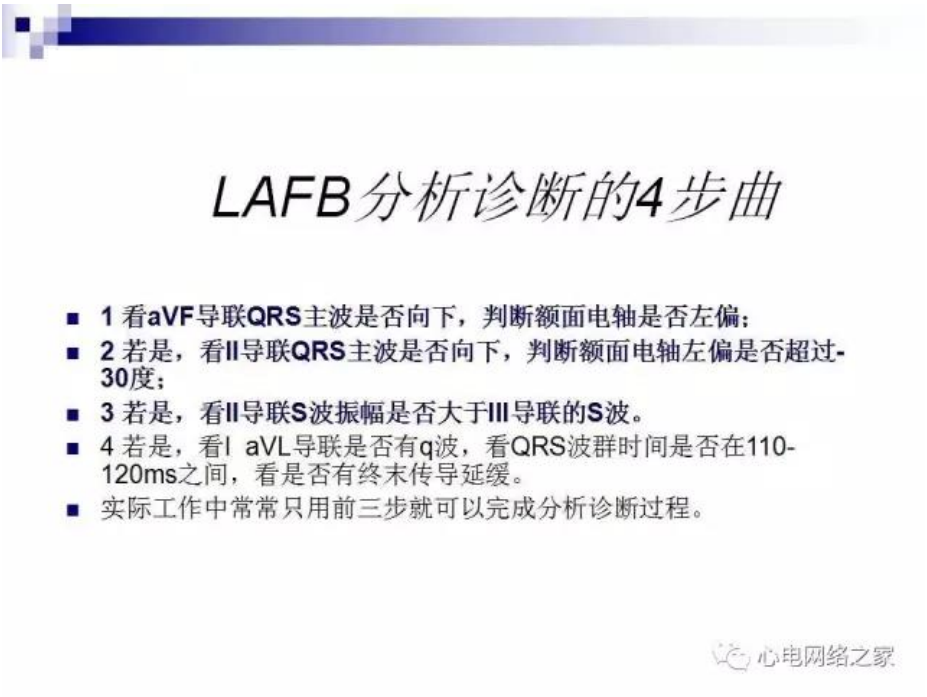


Fig. 7. LAFB diagnoses [**?**]

We see that the key feature here is whether the case has left axis deviation and how much. There is no way we can extract that feature solely from "shape". Thus our convolutional models is very unstable here. We came up with a toolkit to roughly calculate the axis. We also wrote a function to follow the steps shown above to diagnose LAFB with rules. The function works pretty well on training data with accuracy more than 94%. However, it does not work well on the 500 testing data for leader board A. My takeaway here is that we should have trust our method more since 500 is relatively small data set to judge a multi-labeling problem with 9 classes. Our final approach is to use all the tools we had, both the 5000 model and the pulse model, along with the rule-based function we wrote to obtain 3 predictions for each case. Then take the vote of these 3 predictions as our final answer. We set that if 2 models out of 3 votes 1, then we label the case with LAFB.

Further thoughts after the contest is that we could have played with the weights instead of prediction to do the vote. We also should have manipulated the weights of each model. Simply put, we should have trained a mini-model for our voting instead of simply counting the positive predictions.

Our offline accuracy for LAFB was above 90% and f1 score for leader board based on 500 testing data was 0.895. Sadly the final f1 score dropped a lot to 0.71. I think this has much to do with our voting system.

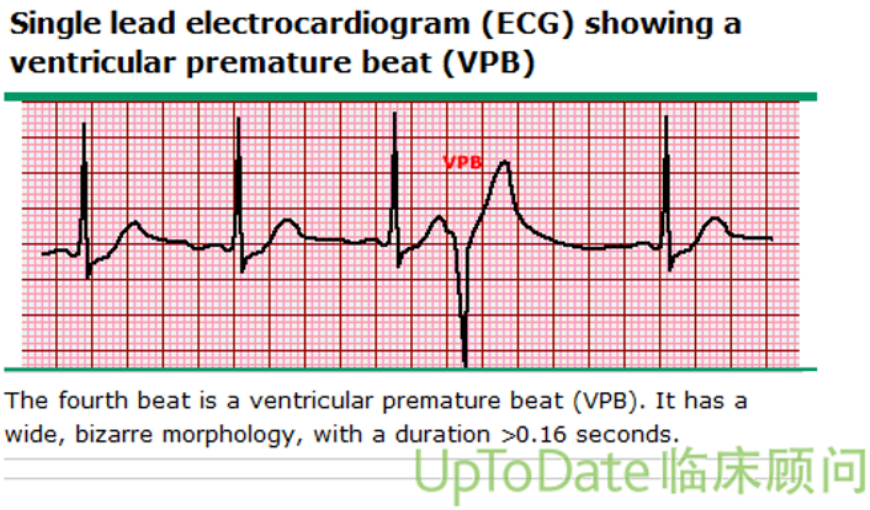


Fig. 8. Premature Ventricular Contraction [**?**]

Premature Ventricular Contraction(PVC) PVC is also sporadic so the pulse model is not well suited. Luckily the characteristics of PVC is very obvious. We used the based model but with shorter width and larger kernel size and achieve about 95% offline validation accuracy. F1 score of the 500 testing data is 0.922 and the final f1 score is 0.92.

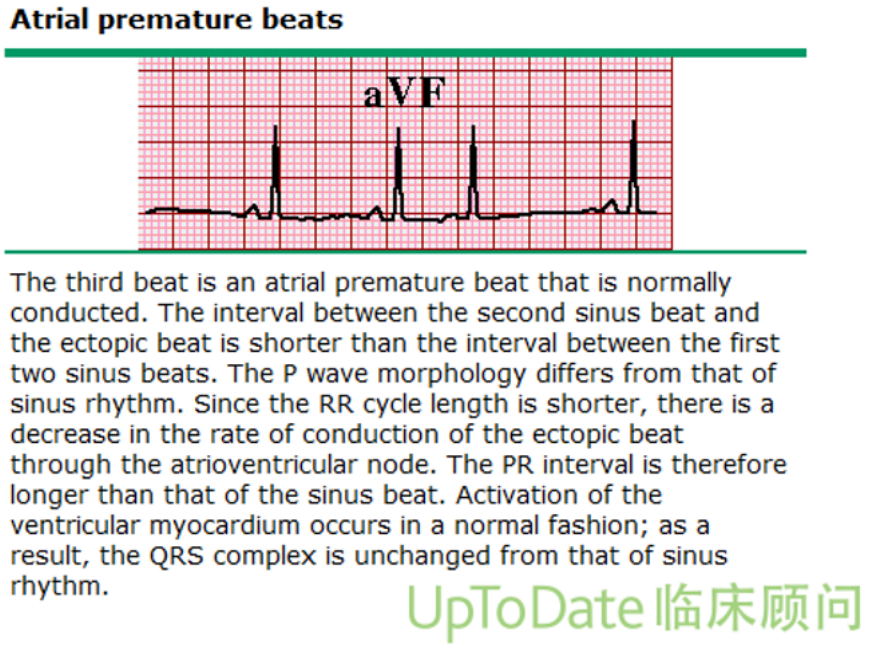


Fig. 9. Premature Atrial Contraction [?]

Premature Atrial Contraction(PAC) This is another naughty abnormality. It was pretty hard for us to improve at around 80% accuracy. The diagnosis of PAC highly relies on heart rate and how its heart rate is irregular. Therefore we calculated more parameters and inserted them into our Dense layers and hoping it would help. We calculated the standard deviation of heart rate, and the maximum difference between RR intervals. It is a pity that we did not realize we can put PAC and AF in one model since they logically do not co-exist. We got approximately 85 validation accuracy offline and f1 score of 0.8 on the 500 testing data. The final f1 score dropped a little to 0.7863.

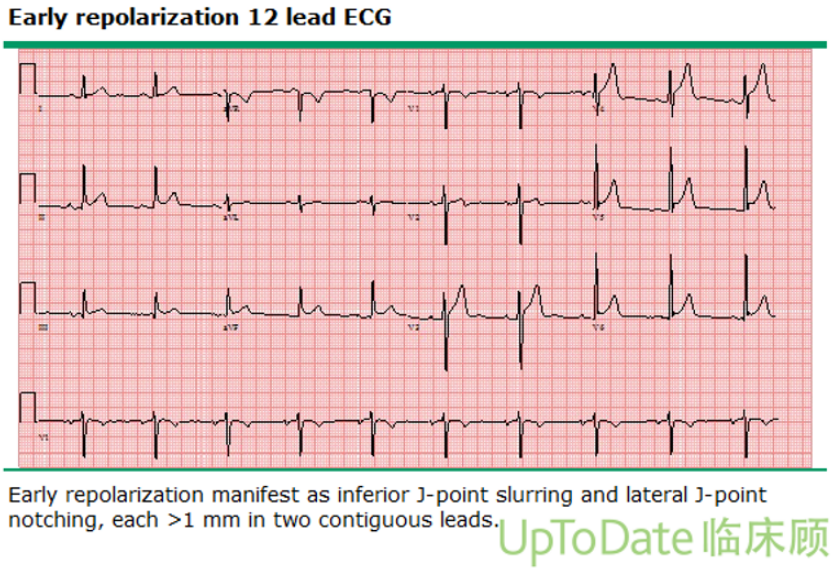


Fig. 10. Early Repolarization [**?**]

Early Repolarization(ER)

T Wave Change(TWC) TWC did not seem to be harder than others. Its characteristics are obvious. T wave is wide and the changes are often significant.

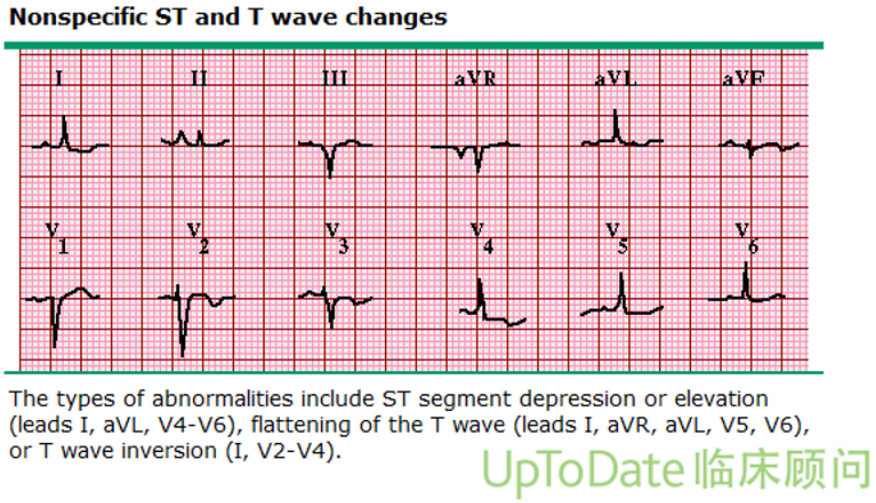


Fig. 11. T Wave Change [?]

We used our based model and achieved more than 90% offline validation accuracy. The f1 score for the 500 testing data was 0.882. However, the final score dropped drastically to 0.7287. After some deeper research, we found that one possible reason why our f1 score dropped a lot is that TWC has a number of sub-categories, as shown below.

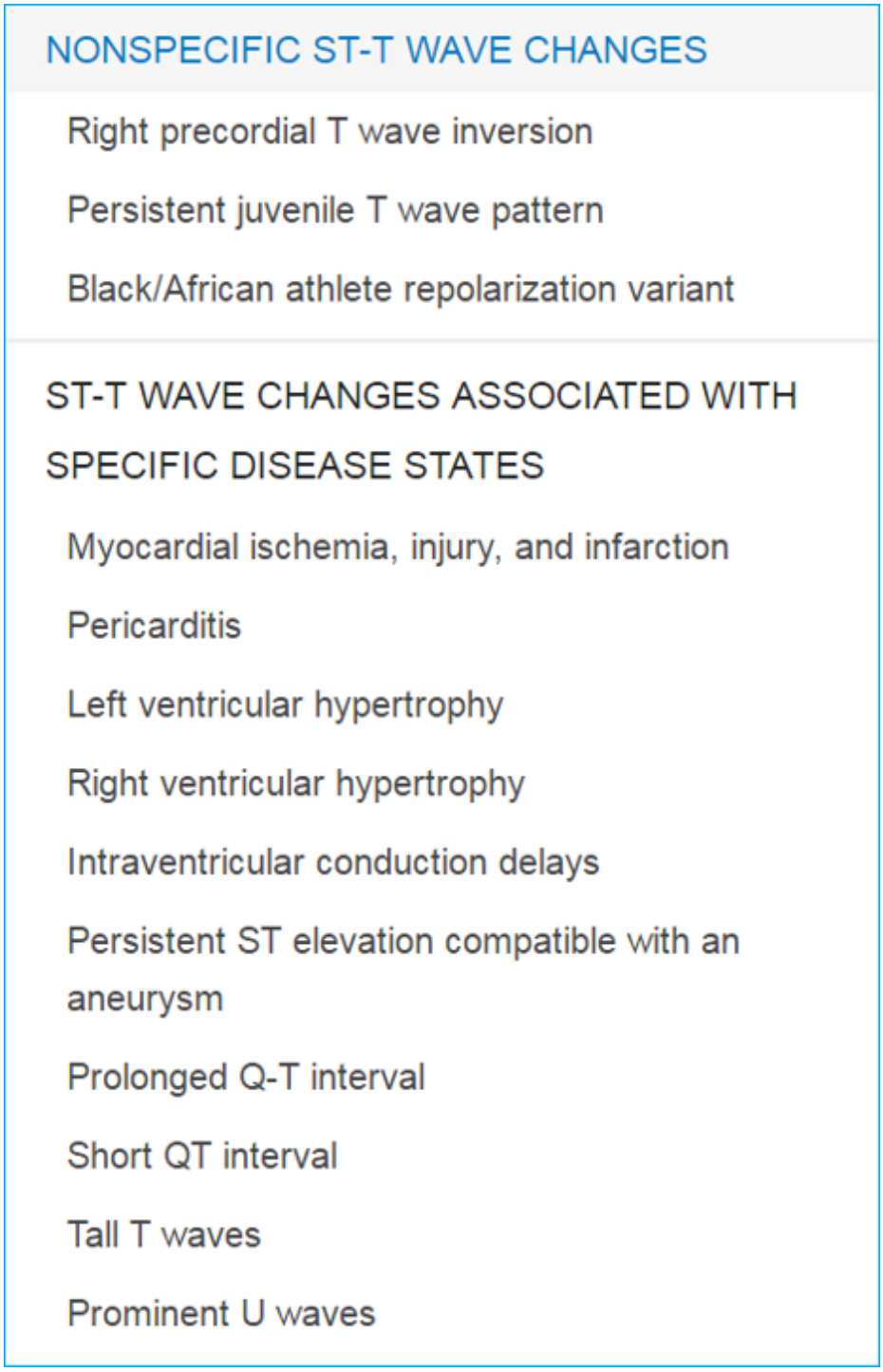
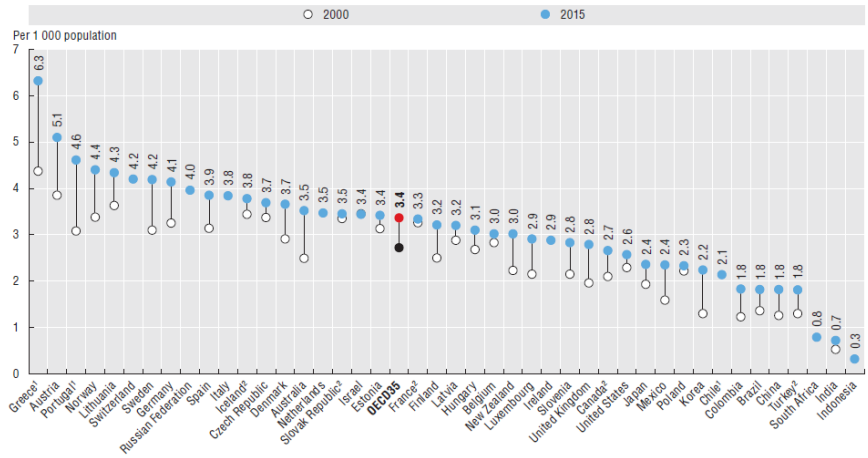


Fig. 12. TWC in details [**?**]

We think our training data did not span every single one of these changes or did not distribute evenly. Also it is hard to say that our validation data was good enough to show how our model perform for the same reasons. In practice, I think we should split TWC into more specific abnormalities to get more accurate models.

A Practicing doctors per 1000 population



1. Data refer to all doctors licensed to practice, resulting in a large over-estimation of the number of practising doctors (e.g. of around 30% in Portugal).
2. Data include not only doctors providing direct care to patients, but also those working in the health sector as managers, educators, researchers, etc. (adding another 5-10% of doctors).

Source: OECD Health Statistics 2017.

StatLink http://dx.doi.org/10.1787/888933604571

Fig. 13. Practicing doctors per 1000 population in 2000 and 2015. [**?**]

B Scoring Rubrics

First of all, define 4 parameters for abnormality j, where $0 \leq j \leq 8$

$$TP_j = |\{x_i | y_j \in Y_i, y_j \in f(x_i), 1 \leq i \leq N\}| \tag{1}$$

$$FP_j = |\{x_i | y_j \notin Y_i, y_j \in f(x_i), 1 \leq i \leq N\}| \tag{2}$$

$$TN_j = |\{x_i | y_j \notin Y_i, y_j \notin f(x_i), 1 \leq i \leq N\}| \tag{3}$$

$$FN_j = |\{x_i | y_j \in Y_i, y_j \notin f(x_i), 1 \leq i \leq N\}| \tag{4}$$

Then calculate *precision*, *recall* and F_1 score

$$Precision_j = \frac{TP_j}{TP_j + FP_j} \tag{5}$$

$$Recall_j = \frac{TP_j}{TP_j + FN_j} \tag{6}$$

$$F_{1j} = \frac{2 \cdot Precision_j \cdot Recall_j}{Precision_j + Recall_j} \tag{7}$$

Finally calculate the average F_1 score of all 9 classes.

$$F_1 = \frac{1}{9} \sum F_1 j \tag{8}$$