Notebook

September 21, 2021

1 Detecting Atrial Fibrilation with 1-D CNNs

So I recently bought an Apple Watch. It was on sale for a pretty good deal and I've been wanting one for quite a while, so I just had to get it. Like I normally do with all of my recently purchased gadgets, I played around with it in the middle of the night to figure out all of its features. This led to me discovering the ECG feature which intrigued me, so I set out to recreate a similar system to detect Atrial Fibrillation (AFib) like Apple's implementation does.

The thing is, I had no prior experience with ECGs. Most of what I know is focused around computers, so the biomedical domain isn't really my specialty. However, I thought this project would serve as a good exploration into the domain.

TLDR; I got an Apple Watch. It can do ECGs. I wanted to replicate that and learn more about ECGs.

1.1 Data

The first thing I had to do was to source a good ECG dataset. Luckily, I quickly found the MIT-BIH Atrial Fibrillation Database. It contains annotated ECG data from 23 unique patients each with two simultaneous 10 hour ECG signals. The annotations detailing the rythym at a given time are given in a record's `.atr` files. There are 4 different types of annotations for hearth rythyms: Atrial Fibrilation (AFIB), Atrial Flutter (AFL), AV Junction (J), and "all other rythyms" (N). The dataset mostly captures AFIB and N rythyms while AFL and J only make up around 1% of the data.

```
<IPython.core.display.HTML object>
```

Again, I am going into this project with little to no background knowledge on how ECGs work and what characteristics each rhythm tends to display. So using the data, I'm going to hypothesize the characteristics of each rhythm then compare my hypotheses with existing information.

1.2 Time Domain Analysis

FFirst, I've curated a small random sample that has decent variation which I have plotted below for a quick visual comparison. While I don't really notice any obvious differences between the different rhythms, I did notice that they see to follow the same pattern of having some sort of lead-up activity, a giant spike in activity and then some follow-up activity. This is extremely prevalent in record 08455.

{% include 1x4.png %}

I later learned that these were the P, R, and T waves of the heart rhythm. However, I missed a couple other waves. According to this diagram, it should appear like this for normal sinus rhythm (Wikipedia).

{% include SinusRhythmLabels.png %}

1.3 Frequency Domain Analysis

Since nothing really stood out between the different rythms, perhaps another view of the data might prove more informative. Using the same sample, I plotted their Discrete Fourier Transforms and noted some observations below.

AFIB

AFL

J

N

• Hard to notice distinct harmonic banding • Noisy †

• Clear harmonic banding • High frequency fundamental

• Sometimes strong harmonic banding • Low frequency fundamental • Slightly noisy †

• Occasionally strong harmonic banding • Records 05091 and 04936 are a little noisy † • Record 08455 has 60 Hz noise(probably a product of the data capture process)

† I should be careful when I say "noisy". If you actually look at the signals they derived from, they aren't so noisy per say. However, in the frequency domain, it is hard distinguished harmonic spikes like we see in the other signals.

{% include 1x4_dft.png %}

1.4 Aggregated FDA

To do a more general comparison, I took 200 samples for each class, applied a Discrete Fourier Transform, and averaged their values to generate the plot below.

{% include mean_ecg_dft.png %}

The only real distinction I can make is sharpness in the banding for each rythym. However, I believe this doesn't really inform us much about the characteristics of each class. We can expect banding as the result of the harmonics produced by the beat rythym. The sharpness of this banding is likely a result of the variance in each label. AFL and J will have low variance due to the limited number of samples available for them while AFIB and N have higher variance since they have a larger sample pool. Thus, AFL and J appear "sharper" while AFIB and N are more "fuzzy".

There is a slight contradiction though. AFIB and N do not follow this pattern. N has slightly more sample availability than AFIB, having about 20% more data available by duration and appearing in more unique occasions that last longer than 5 seconds. Still, AFIB appears to have more variance making it hard to distinguish any clear harmonics. Meanwhile, N has some pretty evident harmonic banding.

One explanation could be due to AFIB appearing in 23 records while N only appears in 21. If this is the case, then I theorize that N's appearance in this plot will become more fuzzy as I increase the sample size for each class. I tested this at two other sample sizes. At 1,000, I observed slightly more variance in both N and AFIB, but harmonic banding remained distinguishable in N. At 10,000 there was no noticable difference than that at the 1,000 level. From this experiment, I can't conclusively say that this isn't the case.

Another explanation could be that AFIB rythyms simply have more variation in BPM resulting in this plot appearing less sharp.

One final explanation might be more clear if we reference the previous frequency domain plot. Recall how I noted that the AFIB plots had indistinguishable spikes. This may indicate the AFIB signals are more eratic resulting in their aggregates appearing as they do.

1.5 Detection Models

Now, my main goal: Creating machine learning models to detect AFib. I actually had a lot of trouble with this initially. I thought that by having such a large dataset, I could just generate random samples during training and validation. However, this system yielded unreliable results, no matter the sample sizes I used. After struggling for a while, I referred to Detection of Atrial Fibrillation Using 1D Convolutional Neural Network (Hsieh, 2020) for some guidance which lead to my final data loading system. After making these changes, I immediately saw better results. Here's the details:

1.5.1 ETL Pipeline

To split the data into 10-second labeled samples with 3-fold cross-validation, I extracted each unique occurance of an annotation and noted the record it came from, when the occurance began, and when it ended. Then I discarded any occurances less than 30s (3x the expected length) and split each occurance into 3 smaller, equally-sized signals and randomly one-to-one mapped each to a fold. From here, the subsamples were further split into 10 second slices with a 50% overlap between each, discarding any excess. This resulted in 54,989 samples (22,020 AFIB, 32,969 N) per fold.

1.5.2 Models

When considering what type model to apply to this problem, I immediately jumped to Convolutional Neural Networks. CNNs have proven themselves as very capable signal classifiers in various other tasks, so I thought that they should be my go-to answer for this problem. However, the exact architecture of a CNN can vary widely so I've compared various designs in this project. The only common elements for the models is that they each take 10-second, 2-lead ECGs as their input and output a prediction of Normal Sinus Rhythym (0) or AFib (1).

To generate a baseline I used two models: First a 1-D variation of the Pytorch MobileNetV2 implementation and second the model described in Hsieh et. al, 2020. I also created a self-made CNN (though admittedly I have very little experience with them). To train a model, I held out one fold for validation and trained on the remaining data and repeated this for each model and fold.

1.5.3 Ensembles

Lastly, I grouped each fold by architecture into ensembles by averaging their outputs (without performing any further training). I then evaluated the ensembles on the entire dataset to determine if averaging outputs was an effective approach for merging the various models together.

```
<IPython.core.display.HTML object>
```

1.6 Conclusion

Coming soon!