Atrial Fibrillation from ECG Data

Background

The dataset used for this project are electrocardiograms (ECG), which are recordings of electrical activity of the heart for a period of time used for detecting abnormal heart rhythms, specifically Atrial Fibrillation (AF) in our case. There are 8,528 recordings of normal, AF, non-AF abnormal, and noisy ecgs, split into a train/test 75/25 (6396)/(2132). To provide a sound model, I took the shortest recording length and cut the rest of the recordings to fit a length of 2714.There were other ways to do this such as taking a random section from each recording or finding a balance of a longer recording and then abandoning the shorter ones to fit against the rest. Each have their own strengths and weaknesses, but I chose my method due to simplicity.

DNN

I started with a dense neural network using a baseline model that can be seen below. When the initial accuracy came back I was surprised at how well only 2 layers had done.

Baseline (Optimizer = Adam, Epoch = 10)

Accuracy	**Layers - Neurons**	**Activation**	**Confusion Matrix**
49.2495%	2 Layers - 4, 10	ReLu - Softmax	[[825 119 377 38] [66 7 30 4] [337 40 162 19] [64 14 27 3]]

The optimized DNN was tuned using the Keras Tuner library which I had previously used in my final project. Once figured out, it's quite easy to work with for DNNs. I decided to try a second optimizer, SGD, as well to see what type of results I could get. It was interesting to see the SGD do so well and get an accuracy of 60%, however when I looked at the confusion matrix I noticed all recordings were only labeled normal readings, somewhat of a "cheating" way for such a high score and also the reason why I included the matrices through this paper. The adam optimizer had not performed as well accuracy wise, but did show labeling for the other 3 categories which makes me more interested in investing into those models.

Optimized DNN (Epoch = 30)

Optmizer	SGD	Adam	
Neurons	832	672	928
Drop Out	False	True	True

LR	0.00860038690486	0.004561896914979461	0.007537119294169523
Confusion Matrix	[1292 180 596 64] [0 0 0 0] [0 0 0 0] [0 0 0 0]	[[1032 157 476 48] [37 6 23 2] [205 15 91 11] [18 2 6 3]]	[[935 117 431 46] [78 14 33 2] [270 47 128 14] [9 2 4 2]]
Percentage	100%, 0%, 0%, 0 %	79%, 3.33%, 15.3%, 4.7%	72%, 7.7%, 21.5%, 3.1%
Accuracy	0.6060037523	0.5309568480300187	0.5060975609756098

Honestly, the number of neurons didn't change things as much as I thought it would, nor the number of epochs (though I only increased it from 10 to 30 rather than something largely significant). The biggest changes were optimizers, and sometimes the drop out which might've helped balance and reset the model between. But the change from adam to sgd was important to see since I shouldn't get caught up in only focusing on accuracy.

The matrices also helped identify how poorly the model picked up on the AF and noisy rhythms. This is probably because of how unbalanced the number of those entries are. The percentage row just shows the accuracy of each label to its own category.

CNN

Multiple issues arose from trying to hypertune the CNN. I can't even begin to explain the 9 hour headache of trying to fix the hypertuner only to find the library had a bug. Not to mention almost every adjustment I made would end up with the same confusion matrix.

Neurons	32	672, 672
Filter, kernal_size	1, 1	32, 3
Optimizer	Adam	SGD
Confusion Matrix	[[1287 178 592 64] [0 0 0 0] [5 2 4 0] [0 0 0 0]]	[[1225 169 559 61] [0 0 0 0] [67 11 37 3] [0 0 0 0]]
Accuracy	0.606003752345215	0.5919324577861164

DNN really pulled through over CNN, I also tried an RNN using LSTM but I ran out of time before it could finish fitting. The baseline of that rnn didn't show great results though 🙁