ECG Signal Classification: KNN vs. CNN

Lucas Raab
Department of Electrical and Computer Engineering, Temple University
lucas.raab@temple.edu

Introduction: In medical diagnosis's, accurate classification of electrocardiogram (ECG) signals is crucial for identifying various cardiac conditions. In this report, I will discuss two methods for building machine learning systems to classify this data: one classification algorithm and one neural network. The data consists of six separate labels (1dAVB, RBBB, LBBB, SB, ST, and AF). Each data file contains an 8-channel signal sampled at 300 Hz, with a total of 2200 samples. The two algorithms I employed were K-Nearest Neighbors (KNN) and a Convolutional Neural Network (CNN). To accomplish these tasks, I utilized the scikit-learn and TensorFlow libraries.

K Nearest Neighbor (KNN): In my initial approach, I used a K-Nearest Neighbors (KNN) algorithm to classify the data. My primary focus in designing the system was to identify effective features. Initially, I considered feeding the raw waveforms into the algorithm, but I soon discarded this idea, thinking that it would produce suboptimal results. It was noted that some others had used this approach and achieved promising results, albeit with significantly prolonged training times. Instead, I opted to utilize various statistical measures of the data as features. These included mean, standard deviation, variance, the five most prominent signals in the Fourier transform, and wavelet energy. With a total of 11 features and 8 channels, each data file would encompass 88 features. During testing, I discovered that all chosen features significantly enhanced the accuracy rate, including seemingly basic measures such as mean. Determining the optimal K value involved a search within the range of 1 to 50 K's. It was found that a K value of 6 yielded the best results, although the accuracy rate stabilized quickly over various K values, with minimal impact on accuracy. While the results of this approach were not optimal, it became apparent that improving them would likely necessitate the incorporation of additional features into the design. Despite my lack of prior knowledge regarding ECG signals during the design phase, I surmise that collaboration with experts versed in signal classification could enable the identification of features representing pertinent quantities. Although the accuracy was modest, the performance was commendable, with the entire system completing execution within a few minutes, requiring minimal optimizations.

Convolutional Neural Network (CNN): In my second approach, I utilized a CNN model for data classification, which offers several advantages over conventional algorithms. Firstly, there's no need for feature extraction as the model inherently extracts features. This eliminates the challenge of defining features manually. In my I chose a CNN model over other neural network architectures, because while doing my research, I saw a lot of CNN's being used in digital signal classification tasks. My CNN system's architecture, I used 6 sperate models each for a binary classification problem for each label. I chose this method, since in my initial testing this gave me better results than classifying each label together. My CNN model comprises four 1-dimensional convolutional layers, each with identical filter and kernel sizes. While my choice of four layers lacked explicit rationale, I observed improved accuracy with increasing layer numbers. In hindsight, I would have experimented with larger kernel and filter sizes. Following convolution, the results underwent max-pooling and dropout layers to enhance model generalization. Subsequently, they were passed through a fully connected layer with 256 nodes. Increasing the node count in this layer enhanced performance for certain labels like '1dAVB', but had minimal impact on already well-performing labels such as 'RBBB'. In future iterations, I'd adjust the dense layer sizes tailored to each label's characteristics. Notably, my model had a parameter count of 16 million per model, totaling 96 million, a figure potentially reducible without compromising performance. Adjusting the pooling and convolutional layers while customizing dense layer sizes for each label could optimize both parameter count and performance.

Results:

KNN	/train	/dev	/eval
Macro Accuracy	0.9099	0.9009	0.9013
Macro Precision	0.7456	0.3948	0.4351
Macro Recall	0.2282	0.1890	0.1907
Macro F1	0.2862	0.2243	0.2276

CNN	/train	/dev	/eval
Macro Accuracy	0.8860	0.8518	0.9044
Macro Precision	0.8490	0.8834	0.7048
Macro Recall	0.6325	0.5564	0.5700
Macro F1	0.7031	0.6491	0.6066

The two tables above display the performance metrics for systems on my train, dev, and eval datasets. My KNN results were rather poor, with consistently low F1 scores, precision, and recall. Conversely, my CNN outperformed the KNN, achieving F1 scores in the 60s. One intriguing observation is that despite the CNN model's superior precision, recall, and F1 scores, the overall accuracy remains similar. Additionally, it's worth noting that across both systems, precision greatly exceeded recall, indicating that my models were more prone to false negatives than false positives.

Conclusions: In conclusion, while both the K-Nearest Neighbors (KNN) and Convolutional Neural Network (CNN) approaches showed promise in classifying electrocardiogram (ECG) signals, the CNN method was noticeably more effective. CNN had greater metrics in almost all fronts compared to my KNN algorithm. Notably, both models exhibited a tendency towards greater precision than recall, suggesting a bias towards false negatives. If I had more time to work on this problem, I would likely try to refine my CNN architecture. I believe if I could tweak individual layers I could improve the models performance, and decrease training times. Overall, this report highlights the potential of machine learning in ECG signal classification and with further refining models like those I discussed could prove very useful.