Deep-Diagnosis of Arrhythmia using Spectral Features

Osafo Nkansah Isaac
Stanford University
757 Campus Drive, Stanford, CA 94305-2004
ike1997@stanford.edu

Abstract

Arrhythmia is a medical condition in which the electrical pulses that control the beating of the heart malfunctions, introducing irregularities in the heart pulsations. These pulsations can be either too slow or too fast, veering away from the normal heart rate of 60-100 beats per minute. A research article, "Cardiovascular Disease Burden, Deaths Are Rising Around the World" by author Dana Kauffman published last year noted that about a 3rd of the global deaths in 2019 were caused by Cardio Vascular Diseases (CVD) [1], most of which can be quickly diagnosed with EKG/ECG data. This paper draws from research findings in the literature on applying machine learning to ECG data to investigate how much of an impact spectral features can have on multi-class Arrhythmia classification task. Is using only spectral features sufficient to produce outstanding performance on our validation set or a combination of both temporal and spectral features? The paper concludes that using the Discrete Cosine Transform (DCT) of our input signals allows us to achieve marginally better results, roughly 2-3% better on metrics such as accuracy, F-1 score etc. compared to using our temporal dataset. More impressively, the paper reveals that when DCT features are used instead of temporal features, we can appreciably reduce our model size while performing comparably better than when using the temporal dataset.

1. Introduction

The advent of smart wearable devices such as Apple Watch Series 4, 5 and 6, Samsung Galaxy Watch 3 and Watch Active 2 etc., to track heart beat offers a great opportunity to investigate efficient embeddable ML models to help with early detection of cardio-vascular diseases. Over the past couple of years, Machine Learning techniques have been applied to temporal data series of EKGs/ECGs to classify whether an EKG/ECG reading predicts a certain class of arrhythmia or not. 5 major groups of arrhythmia have been considered in the literature on applying Machine Learning to arrhythmia prediction. These 5 classes, labelled N, S, V, F, and Q, will also be used in our experimentation. Further details on these 5 groups are as shown in figure 1. All the models this paper highlights take in

Category	Class
N	• Normal beat (N) • Left and right bundle branch block beats (L,R) • Atrial escape beat (e) • Nodal (junctional) escape beat (j)
S	• Atrial premature beat (A) • Aberrated atrial premature beat (a) • Nodal (junctional) premature beat (J) • Supraventricular premature beat (S)
V	• Premature ventricular contraction (V) • Ventricular escape beat (E)
F	• Fusion of ventricular and normal beat (F)
Q	• Paced beat (/) • Fusion of paced and normal beat (f) • Unclassifiable beat (U)

Figure 1. Different Classes of Arrhythmia

a dataset that contains samples/signals from the different classes of Arrhythmia and aims to perform a multi-class classification on the dataset. The paper is dedicated to exploring the significance of using spectral features, if any, in the hopes of developing a high performing, low memory embeddable model that rivals state-of-the-art models on single-lead ECG data. The rest of the paper explores this in further depth.

2. Related Work

The current state-of-the-art performance on Physionet's MIT-BIH dataset, as documented in a paper titled "SE-ECGNet: A Multi-scale Deep Residual Network with Squeeze-and-Excitation Module for ECG Classification" by authors Haozhen et al.[2] notes an F1 score of 99.2% on the dataset. Though the authors recognize that heartbeat signals are periodic, they only incorporate this fact, howbeit subconsciously, in the preprocessing stage of their experimentation. Though it cannot be quantified how much performance improvement the FFT (Fast Fourier Transform) transformation of the input signals contributed in their final F1-scores and other performance metrics, given that other papers on the same task of Arrhythmia classification pro-

duced slightly lower performance results [3,4], we can hypothesize that the FFT transformation may have contributed to the gains in performance/predictive accuracy and the high F-1 score recorded in their paper. Please refer to figure 2 below for more details on their results.

TABLE I
EVALUATION RESULTS

Model	MIT-BIH dataset			Alibaba dataset		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Seq2Seq	87.7%	87.7%	87.7%	33.8%	33.8%	33.8%
ResNet1D-34	97.4%	97.4%	97.4%	85.7%	85.7%	85.7%
ECGNet	97.2%	97.2%	97.2%	86.2%	86.2%	86.2%
SE-ECGNet	**99.2%**	**99.2%**	**99.2%**	**89.8%**	**89.8%**	**89.8%**

Figure 2. Haozhen et. al.: results on MIT-BIH dataset

Haozhen et. al. argue that their model "is able to optimally leverage the multi-lead information from the ECG signals of the MIT-BIH dataset, using a mix of 2D CNNs and 1D CNNs". The results above apply to multi-lead ECG data, which though very interesting, might not be very practical in the space of embedded devices. Embedded devices usually have memory constraints which limit them from storing too much data at a time. Having noted this, given the amazing results from Haozhen et. al's paper, the goal of this paper will be to use Haozhen et. al.'s paper as an inspiration on single-lead ECG signals to achieve similar performance on metrics such as F1-score and precision.

3. Dataset

We will be using Shayan Fazeli's pre-processed MIT-BIH dataset on Kaggle. This dataset contains single-lead ECG signals, which were extracted from the multi-lead MIT-BIH dataset provided by Physionet. The preprocessing was essentially cleaning up the dataset to retain only one of the signals from the two leads of the ECG. The distribution of the dataset is as given below in the table:

Dataset	N	S	V	F	Q	Total
MIT-BIH Arrhythmia	90,462	2,777	7,223	802	8,027	109,291

Figure 3. MIT-BIH single-lead dataset distribution

Shayan Fazeli's dataset is split by a predefined ratio into a training and a testing/validation set, which I intend to use as is unless I eventually find that there is a need to re-split the data. Also, we note how unevenly distributed the dataset is. Explorations in later sections will explore plausible ways to rectify this uneven distribution in a preprocessing stage before we train on the final result.

4. Pre-processing and Baseline setting

Our initial experimentation was to find intelligent ways of supplementing the dataset through various resampling and data augmentation techniques to mitigate the issue of unbalanced dataset. The approach here was to first find a good baseline model we can use to determine how useful our augmentation techniques were compared to just using the original dataset. The baseline model we used was inspired by Amritvir Singh's submission on Kaggle for this same problem of single-lead ECG data multi-class classification. The model Amritvir Singh used is outlined below:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 187, 64)           448
_________________________________________________________________
batch_normalization (BatchNo (None, 187, 64)           256
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 94, 64)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 94, 64)            24640
_________________________________________________________________
batch_normalization_1 (Batch (None, 94, 64)            256
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 47, 64)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 47, 64)            24640
_________________________________________________________________
batch_normalization_2 (Batch (None, 47, 64)            256
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 24, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 1536)              0
_________________________________________________________________
dense (Dense)                (None, 64)                98368
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 325
=================================================================
Total params: 153,349
Trainable params: 152,965
Non-trainable params: 384
_________________________________________________________________
```

Figure 4. Amritvir Singh's kaggle submission model (Model 1)

With this model, and just using the raw dataset (without balancing the data distribution across different class), we trained a model using ADAM optimizer and categorical_crossentropy loss for 15 epochs and yielded the results below:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	18118
1	0.85	0.83	0.84	556
2	0.97	0.96	0.96	1448
3	0.80	0.79	0.79	162
4	1.00	0.99	0.99	1608
accuracy			0.99	21892
macro avg	0.92	0.91	0.92	21892
weighted avg	0.99	0.99	0.99	21892

Figure 5. Model 1's results using raw dataset

We then used a resampling (with replacement) technique to increase the count of all low-count classes while using a downsampling method to reduce the count of class 0 (which was disproportionately largest) but we observed that the higher the sampling, the closer we got to the results of the raw dataset, provided in figure 5. The reason for this observation was that downsampling class 0 yielded distinct examples while resampling (with replacement) from the other classes with lower counts (especially classes 1 and 3) resulted in a lot of duplicates of the same samples. Relative to the raw training case, the high number of duplicated data for the other classes (especially 1 and 3) reduced the model's ability to generalize for those classes. Additionally, given that we were resampling from classes with low counts, some of the original samples in the low-count classes were missing from the final dataset we trained on. Given the decreased uniqueness of data samples for classes other than class 0, it is expected that the model will

not be able to learn well for those classes. In general, the conclusion here was to try another augmentation technique: applying high frequency noise to the dataset. ECG signals from the leads are subject to a lot of different noise from sources such as the power line, the electrode contact points, baseline drifts/wander etc. Categorically, these noise can be either low or high frequency. We mainly experimented with high frequency noise, to perturb the data for the low data count classes in order to generate more data (only for our training dataset). The noising scheme, given any data sample, S, was to generate $S^{'}$ using the expression below:

$S^{'} = S + \frac{G}{||G||_2^2}$, G is the random Gaussian noise.

Using this new balanced dataset, we retrained the same model for the same number of epochs (15). This method, though it produced better results than what Amritvir Singh recorded in his Kaggle submission, was also suboptimal compared to just training the model using the raw dataset. We even proceeded to use a deeper model, due to the relatively low accuracy on both training and validation sets we observed. This turned out to only be marginally helpful. A summary of this result is listed out in figure 7 below.

	precision	recall	f1-score	support
0	0.99	0.99	0.99	18118
1	0.82	0.86	0.84	556
2	0.96	0.96	0.96	1448
3	0.73	0.85	0.79	162
4	0.99	0.99	0.99	1608
accuracy			0.98	21892
macro avg	0.90	0.93	0.91	21892
weighted avg	0.98	0.98	0.98	21892

Figure 6. Model 1's results using Gaussian noised samples

We note interestingly that the macro average skews towards the recall. We observed a 1% increase in recall, at the expense of precision and hence the F-1 score metric (when considered relative to the raw dataset case). The conclusion from this section was essentially to use the original unbalanced dataset (which we note here was also proportionally unbalanced in the test set) given its optimal performance on all 3 metrics: precision, recall and F-1 score. In authors Haozhen et. al.'s paper, they performed a pre-processing step on the input dataset using the FFT (Fast Fourier Transform), a DFT transformation which converts the temporal signals into spectral information. Given the better performance of the SE-ECGNet compared to other models on the multi-lead MIT-BIH dataset, it was only natural to try a similar pre-processing step. For any discrete signal a_n, of length N, with $n \in 1, ...N$, the DFT is defined as shown below:

$$A_k = \sum_{n=0}^{N-1} e^{-i\frac{2\pi}{N}kn} a_n$$

A is our N-length discrete output DFT signal, with index $k \in 1, ...N$. This is what was used to generate the complex FFT features, from which the real, imaginary and absolute value features are extracted as inputs into our various models. In our exploration, we tried out two primary spectral transformations, i.e. the FFT and the DCT (Discrete Cosine Transform). We used the DCT-II transform, defined as shown below:

$$y_k = 2\sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi k(2n+1)}{2N}\right)$$

x_n is our N-length discrete input signal, with index $n \in 1, ..., N$ and y_k, also our N-length discrete output DCT-II transform of x_n with index $k \in 1, ..., N$. We visualize these transforms using one of the normal heartbeat signals; we show the original signal, its DCT and the absolute value of its FFT/DFT in figures 7, 8 and 9 respectively.

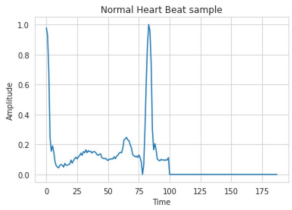

Figure 7. Normal Heart Beat sample

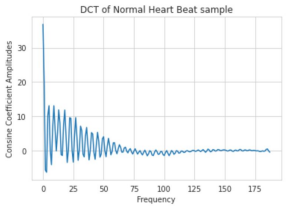

Figure 8. DCT of Normal Heart Beat sample in figure 8

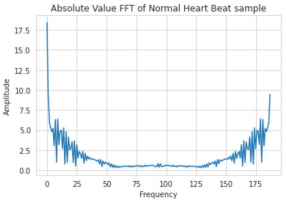

Figure 9. Abs FFT of Normal Heart Beat sample in figure 8

The explorations in the upcoming sections use very similar architectures as used in model 1 (figure 5), together with spectral features to raise performance on our desired metrics. The exploration draws inspiration from both Amritvir Singh's CNN model architecture and the FFT preprocessing steps used by Hao et. al. in their SE-ECGNet paper.

5. Methods and Experiments

In this section, we had 2 primary exploration objectives:

i. investigate whether using only spectral features is sufficient to produce comparable results to what we obtained with the raw dataset

ii. investigate whether a combination of both spectral features and temporal features offer better performance results on our validation set

For objective 1, we extracted the DCT features of our training and validation dataset to use in our training and validation steps respectively. Here, we used the same model as in figure 5, except we used a deeper network. The summary of the model is given below:

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv1d_3 (Conv1D)	(None, 187, 64)	448
batch_normalization_3 (Batch	(None, 187, 64)	256
max_pooling1d_3 (MaxPooling1	(None, 94, 64)	0
conv1d_4 (Conv1D)	(None, 94, 64)	24640
batch_normalization_4 (Batch	(None, 94, 64)	256
max_pooling1d_4 (MaxPooling1	(None, 47, 64)	0
conv1d_5 (Conv1D)	(None, 47, 64)	24640
batch_normalization_5 (Batch	(None, 47, 64)	256
max_pooling1d_5 (MaxPooling1	(None, 24, 64)	0
conv1d_6 (Conv1D)	(None, 24, 64)	24640
batch_normalization_6 (Batch	(None, 24, 64)	256
max_pooling1d_6 (MaxPooling1	(None, 12, 64)	0
flatten_1 (Flatten)	(None, 768)	0
dense_3 (Dense)	(None, 64)	49216
dense_4 (Dense)	(None, 64)	4160
dense_5 (Dense)	(None, 64)	4160
dense_6 (Dense)	(None, 5)	325

Total params: 133,253
Trainable params: 132,741
Non-trainable params: 512

Figure 10. Deeper Model (version of model in figure 5)

We compiled our model using an ADAM optimizer with categorical_crossentropy loss. We trained this model for different epochs and realized that an epoch of 25 was sufficient. This was judged based on where our validation accuracy plateaued during the training phase. For investigation 2, we used the same model above, but modified the input shape of our first 2D convolution layer appropriately to accommodate different 2D input "images" of our signals. For investigation 2, we had multiple options; we could append the transforms as extra channels of the input signal or as extra rows to construct a "2D image" of our signals. We opted to go for the latter. We investigated four different "2D images"; in all four different cases, the original signal itself occupied the first row of the "image". For our actual training, we used a transpose of the aforementioned "2D images" (using rows as columns and columns as rows) due to how pre-processing was done. The image transposition was the same for the testing and training data so our results would be the same whether we used the "2D images" as aforementioned or their transposes. As such, we describe the "2D images" in a (H, W, C) format but in our model you'll find a (W, H, C) instead. A summary of the input "2D image" shapes has been outlined below:

a. "2D image" of shape (2,187,1) with the 1st row as the original signal and the 2nd row as the DCT transform of the signal

b. "2D image" of shape (3,187,1) with the 1st row as the original signal, the 2nd row as the DCT transform of the signal and the 3rd row as the absolute value of the FFT of the signal

c. "2D image" of shape (5,187,1) with the 1st row as the original signal, the 2nd row as the DCT transform of the signal, the 3rd row as the real component of the FFT of the signal, the fourth row as the imaginary component of the FFT of the signal and the last row as the absolute value of the FFT of the signal

The architecture we used for the 2nd part of the investigation is provided below in figure 12. Similarly before

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 187, 3, 64)	640
batch_normalization_3 (Batch	(None, 187, 3, 64)	256
average_pooling2d (AveragePo	(None, 63, 1, 64)	0
conv2d_1 (Conv2D)	(None, 63, 1, 128)	73856
batch_normalization_4 (Batch	(None, 63, 1, 128)	512
average_pooling2d_1 (Average	(None, 21, 1, 128)	0
conv2d_2 (Conv2D)	(None, 21, 1, 128)	147584
batch_normalization_5 (Batch	(None, 21, 1, 128)	512
average_pooling2d_2 (Average	(None, 7, 1, 128)	0
conv2d_3 (Conv2D)	(None, 7, 1, 64)	73792
batch_normalization_6 (Batch	(None, 7, 1, 64)	256
average_pooling2d_3 (Average	(None, 3, 1, 64)	0
global_average_pooling2d (Gl	(None, 64)	0
dense_3 (Dense)	(None, 64)	4160
dense_4 (Dense)	(None, 64)	4160
dense_5 (Dense)	(None, 5)	325

Total params: 306,053
Trainable params: 305,285
Non-trainable params: 768

Figure 11. 2D Deeper Model (version of model in figure 5)

training, we compiled our model using an ADAM optimizer with categorical_crossentropy loss. Additionally, we note that instead of a flatten layer, we used a global average pooling layer. This switch borrowed directly from Hao et. al.'s model in their paper on SE-ECGNet.

6. Results and Discussion

Under investigation 1, we found that using only spectral features was more than sufficient to get similar and sometimes even better results when we change hyperparameters such as the optimizer, the epochs etc. Experimenting with the FFT and the DCT features of the signal, we realized that the DCT features performed better in the same number of

epochs compared to the absolute value FFT transform features. We hypothesize that taking the absolute value of the FFT of our signal, which is but one of many ways we could have combined the real and imaginary components of our transform, limits expressivity. DCT-II produces real value coefficients for the series of cosine functions that can be used to capture the spectral information within our signals and imposes no constraints on these spectral coefficients. This allows us to feed the output of the DCT transform directly into our model, allowing for better expressivity, which explains the marginally better results we observed with the DCT features. We have juxtaposed the results on both the raw dataset, the absolute value of the FFT features and the DCT features below in figures 12, 13 and 14.

	precision	recall	f1-score	support
0	0.99	0.99	0.99	18118
1	0.85	0.83	0.84	556
2	0.97	0.96	0.96	1448
3	0.80	0.79	0.79	162
4	1.00	0.99	0.99	1608
accuracy			0.99	21892
macro avg	0.92	0.91	0.92	21892
weighted avg	0.99	0.99	0.99	21892

Figure 12. Trained Model results with raw dataset

	precision	recall	f1-score	support
0	0.99	1.00	0.99	18118
1	0.89	0.83	0.86	556
2	0.96	0.96	0.96	1448
3	0.88	0.79	0.83	162
4	1.00	0.99	0.99	1608
accuracy			0.99	21892
macro avg	0.95	0.91	0.93	21892
weighted avg	0.99	0.99	0.99	21892

Figure 13. Trained Model results with DCT features

	precision	recall	f1-score	support
0	0.98	0.99	0.99	18118
1	0.87	0.73	0.80	556
2	0.95	0.94	0.94	1448
3	0.87	0.73	0.79	162
4	0.99	0.97	0.98	1608
accuracy			0.98	21892
macro avg	0.93	0.87	0.90	21892
weighted avg	0.98	0.98	0.98	21892

Figure 14. Trained Model results with Absolute value FFT features

From the results above, we see that training with DCT features instead of the raw dataset or the FFT features helps the model to perform better, especially on datasets with low data counts. For classes 1 and 3, we observe between 2% - 4% increase in F-1 score when we use the DCT features of our input signal. This increase in performance is attributed to the fact that the model is able to learn on the fundamental constituents of our signals, the cosine functions that defines our periodic heartbeat signals. These constituents are what compose any general heartbeat signal and hence the model's ability to learn directly on these transformed features, gives it the marginal upper hand, as far as better precision, recall and f1-score are concerned. The FFT/DFT transformation assumes that the input signal is periodic. While the DFT/FFT features allows the model the same capability, the FFT features are constrained in the input layer, which we hypothesize to be one of the causes for the low performance when using FFT spectral features. As far as exploration 2 was concerned, we didn't delve deeper into tgis exploration because we realized quickly that the combination of these features produced suboptimal results. This was primarily attributed to redundancy of information the model was trained on. In essence, either temporal features or spectral features are able to completely capture the content of any periodic signal (in theory). Combining these features leads to redundancy that our model wasn't able to exploit well to raise predictive accuracy, precision, recall etc.

7. Conclusion and Future Work

We generally observed that using spectral features, particularly the DCT features of our ECG signals produced comparable results on recall and better performance on metrics such as precision, per-class F-1 score and the overall F-1 score across all the different classes in our multi-class task. This is hypothesized to be due to the model not being able to learn directly from the fundamental constituents of the heartbeat signals: the coefficients of the cosine functions that define the spectral properties of our signals. The primary point here is the fact that DCT features help with improved performance of our model on our validation set, howbeit marginally. While this finding may be helpful, it prompts further research into why using these DCT features are better than using the raw signals, given that our DCT-II transformation can be lossy. Additionally, taking a DCT can be resource intensive and computationally expensive, which adds an extra overhead which is undesirable in the context of edge computing and/or embedded systems. Further research will be needed in this area to examine whether a compromise between using DCT features, model size, sample series length can be achieved such that the overhead of using DCT features is mitigated by requiring a smaller model, smaller-length signal sequences etc. We started delving into this at the later part of our investigation, where we trained a smaller model for just 15 epochs on the DCT features and realized that it achieved comparable results to using the raw dataset.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	18118
1	0.93	0.78	0.85	556
2	0.96	0.96	0.96	1448
3	0.86	0.78	0.82	162
4	0.99	0.98	0.98	1608
accuracy			0.99	21892
macro avg	0.95	0.90	0.92	21892
weighted avg	0.98	0.99	0.98	21892

Figure 15. Smaller Model results with DCT features

Overall, further research is still needed to confirm what tradeoffs can be made. Nonetheless, we consider the investigations carried out so far a success.

References

[1] Kauffman, Dana. "Cardiovascular Disease Burden, Deaths Are Rising Around the World." American College of Cardiology, 9 Dec. 2020, www.acc.org/about-acc/press-releases/2020/12/09/18/30/cvd-burden-and-deaths-rising-around-the-world

[2] H. Zhang, W. Zhao and S. Liu, "SE-ECGNet: A Multi-scale Deep Residual Network with Squeeze-and-Excitation Module for ECG Signal Classification," 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Seoul, Korea (South), 2020, pp. 2685-2691, doi: 10.1109/BIBM49941.2020.9313548. [2012.05510] SE-ECGNet: A Multi-scale Deep Residual Network with Squeeze-and-Excitation Module for ECG Signal Classification (arxiv.org)

[3] Sajad Mousavi, Fatemeh Afghah, Fatemeh Khadem, U. Rajendra Acharya,ECG Language processing (ELP): A new technique to analyze ECG signals, Computer Methods and Programs in Biomedicine, Volume 202, 2021,105959,ISSN 0169-2607,https://doi.org/10.1016/j.cmpb.2021.105959. ECG Language processing (ELP): A new technique to analyze ECG signals - ScienceDirect

[4] M. Kachuee, S. Fazeli and M. Sarrafzadeh, "ECG Heartbeat Classification: A Deep Transferable Representation," 2018 IEEE International Conference on Healthcare Informatics (ICHI), New York, NY, USA, 2018, pp. 443-444, doi: 10.1109/ICHI.2018.00092. Inter- and Intra- Patient ECG Heartbeat Classification for Arrhythmia Detection: A Sequence to Sequence Deep Learning Approach — IEEE Conference Publication — IEEE Xplore