Cardiac Arrhythmia Detector

CS 4641 Project to detect and classify cardiac arrhythmias

View the Project on GitHub dkatragadda3/ml-group-32

Introduction/Background

Our project topic is to classify the several types of cardiac arrhythmias given an EKG reading. To train our models, we will be using the MIT–BIH arrhythmia database. To ensure our predictions will be as accurate as possible, we will compare the results of three different supervised learning models, a convolutional neural network (CNN), a support vector machine (SVM), and a long short-term memory model (LSTM). We have also conducted a literature review for each model that summarizes each model and past implementations.

Literature Review

TCNs

TCNs are deep neural networks that are optimized for use with time series data. In the paper by Ingolfsson et al., they explore the detection of arrhythmias using a TCN on single lead ECG readings. The results demonstrate that the model matches an LSTM-based system on accuracy and improves balanced accuracy by approximately 16.5% [2]. The TCN model is concluded to be much more generalizable than the LSTM [2].

CNNs

CNNs are networks that apply a convolutional filter to data for feature extraction. Tun and Khine used a CNN that "demonstrates the results of accuracy that are efficient for irregular heartbeats or arrhythmia detection like atrial fibrillation and flutter" [1]. Their results were strong, with a training and validation accuracy of 98.6% and 92%.

LSTMs

LSTM networks are recurrent networks that use gating to remember long sequences [3]. Classifications can be made using the hidden recurrent states of the network. The original paper uses backpropagation through time with SGD for optimization, but we plan to use the Adam optimizer at it appears to work better in practice [4].

SVMs

SVMs are algorithms that attempt to fit a hyperplane to data in order to draw a boundary between two classes. Asl et al. demonstrate that it is possible to classify ECG readings with up to 99% accuracy. The primary benefit of using an SVM over other classification techniques was inference speed, however the model required feature reduction methods and heavy filtering of the input ecg to achieve its speed and accuracy [5].

Dataset Description

The MIT-BIH dataset, found here, contains 48 samples of 30-minute ECG readings taken on patients with a 2-lead setup. This dataset is the industry standard for automated arrhythmia detection.

Problem Definition

Reading EKGs takes a significant amount of time. Also, not all physicians and healthcare professionals are experts at reading EKGs.

We aim to drastically reduce this analysis time and potentially the error rate with automated anomaly detection.

Methods

Preprocessing

Data preprocessing method:

For the LSTM, the EKG signal readings were divided into 10 second segments, this would be enough time to see multiple heart beats. This length for the segments was chosen as here the model would have enough temporal data and not need to denoise the raw signals since it was clean on its own 1. The EKG signal overall is well preserved and 2. Denoising would add some additional latency to our models which we again avoided. We then windowed the data to generate additional data points and add training labels from the dataset to create our dataset.

For the CNN model, we applied multiple different transformations to the data in order to preprocess it. This includes segmenting the heartbeats using the pan_tompkins algorithm and aligning the peaks to the annotations, applying multiple filters (including a high-pass, notch, and low-pass) to remove noise and baseline wander. Additionally, we applied smote to help even out the non-uniform distribution of our dataset.

For the SVM, the initial segmentation was relatively similar to the LSTM methodology. After segmentation, we modified the dataset into a series of single dimension arrays. Due to training results, it was determined that using a single EKG lead was sufficient for accurate detection while also reducing training and inference times. Each reading was zero-padded to ensure consistent dimensions, and the labels were converted to their appropriate classes prior to training. An 80-20 split was used for training and validation separation. We also reduced the number of classes to the 5 AAMI (Association for the Advancement of Medical Instrumentation) classes in accordance with AAMI guidelines.

ML Algorithms

LSTM networks are algorithms typically used for any type of sequence classification tasks, as they are able to remember longer term dependencies they are a great option for our analysis of sequential heartbeats. Our use of Adam optimizer over stochastic gradient descent allowed for improved efficiency due to faster convergence.

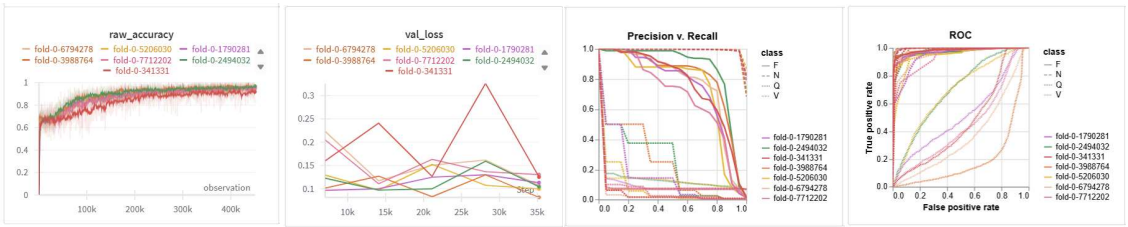
The CNN model, while often used just for images, can easily be applied to our dataset, as heartbeats follow a very structured output. This means that CNN will be able to find and determine patterns in our data quickly and efficiently, which we hoped would lead to more accurate results. Additionally, for the CNN model, we decided to implement SMOTE, which is typically used to help alleviate imbalances in our dataset (which we discovered with LSTM).

SVMs are algorithms used to classify data by generating an n-dimensional hyperplane to find the optimal separation between categories. While they are normally used for binary classification tasks, we can combine multiple models using a one-versus-rest approach to tackle as many classes as necessary. One major issue with SVMs however, is that they can become easily biased with severely imbalanced datasets. This is an issue with the current dataset that may lead to classification issues. It is possible to tune the gamma and C values of the SVM to reduce the bias, which was explored in the training process.

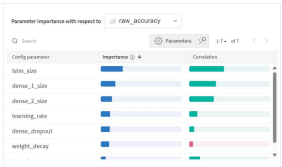
Potential Results and Discussion

Visualizations

We were able to generate various graphs to keep track of our model's performance. We focused on Raw Accuracy, Value Loss, Precision v. Recall, and ROC (False Positives v. False Negatives). See visuals of our top 7 runs below:



Note: We were also able to generate graphs comparing the importance and correlation of different hyperparamters against the runs. While not shown in depth, due to the vast amount of data, here is an example of for the Raw Accuracy Metric:

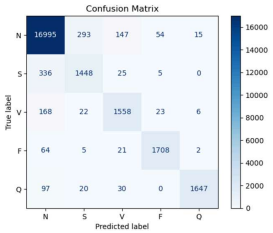


For CNN, in order to showcase the extent to which our data was skewed, we generated some graphs to show the label distribution before and after applying smote to the data. As we can see, we go from a label having a max and min of 82.3 and 0.7% respectively, to 70.7 and 7.3%. This means our models have a better chance of training accurately against the more rare conditions.

	precision	recall	f1-score	support
N	0.96	0.97	0.97	17504
S	0.81	0.80	0.80	1814
V	0.87	0.88	0.88	1777
F	0.95	0.95	0.95	1800
Q	0.99	0.92	0.95	1794
accuracy			0.95	24689
macro avg	0.92	0.90	0.91	24689
weighted avg	0.95	0.95	0.95	24689

```
False Positive Rate (FPR) per class:
N: 0.09
S: 0.01
V: 0.01
F: 0.00
Q: 0.00
False Negative Rate (FNR) per class:
N: 0.03
S: 0.20
V: 0.12
F: 0.05
Q: 0.08
```

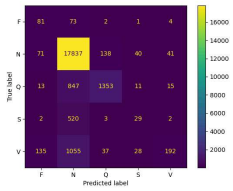
Also, we generated a Confusion Matrix for CNN which shows a high consistency between the true values and the predicted values.



For SVM, we generated multiple tables and matrices to showcase the model's performance. The data included in these tables includes the Weighted Average across classes, the False Positive and False Negatives per class, f1 scores, and the Confusion Matrix.

```
ROC_AUC: 0.8562138852847587
True Positives: [0.50310559 0.98400177 0.68428763 0.05215827 0.13268832]
True Negatives: [0.99012026 0.4333489  0.99112987 0.99635933 0.99705924]
False Positives: [0.00987974 0.5666591  0.00887093 0.00364067 0.00294076]
False Negatives: [0.49689441 0.01599823 0.39571237 0.94784173 0.86731168]
```

	precision	recall	f1-score	support
F	0.27	0.50	0.35	161
N	0.88	0.98	0.93	18127
Q	0.88	0.68	0.72	2239
S	0.27	0.05	0.09	556
V	0.76	0.13	0.23	1447
accuracy			0.87	22530
macro avg	0.61	0.46	0.46	22530
weighted avg	0.85	0.87	0.84	22530



Quantitative Metrics

For LSTM: As we can see from these graphs, for our best runs, our accuracy quickly jumped to the high 90s, with our best accuracy reaching about 98.5%. Furthermore, our validation loss was fairly minimal at around 10%. This was stabilized by about 12k steps. With regards to the precision versus recall graph, we can see that our main 7 runs curve towards the top right corner, meaning our model has a high recall and high precision in general. This is supported by our accuracy percentage. Additionally, our ROC shows this is also true, as our model hovers towards the true positive side while dodging the false positives.

The PR and ROC curves could both be used to evaluate the performance on class Q. On one hand, the LSTM model does show a good ability to discern between true cases of Q and other cases. On the other hand the PR curve shows us that since Q is so rare, the model will need a much better ability to discern Q in order to prevent too many false positives when used at scale.

For CNN: Using just 20 epochs, we were able to achieve a total accuracy of 94.6%, with the highest percent of accuracy being the "Q" class with a precision of 0.99. Furthermore, our validation loss was consistently low at around 13.5% for the 20 epochs. Now, in regards to precision versus recall: CNN was able to get a weighted average across all classes of about 0.95. This is very impressive, as false positive rates were minimized (maxing out at just 0.09 for "N" class and dropping to 0 for "F" and "Q").

However, for the false negative rates, CNN performed slightly worse, reaching 0.2 for the "S" class, which caused a precision score of just 0.81. We also were able to generate the f1-scores, which are often used for imbalance datasets. Our results for CNN show a maximum f1-score of 0.97 for the most popular class ("N") and a minimum f1-score of 0.8 for the less common class ("S").

For SVM: Despite working on a large imbalanced dataset, SVM performed decently. It achieved a total accuracy of about 0.87 and a weighted average across all classes of approximately 0.85. Furthermore, we can see from the results that it only performed well for the 2 most common classes ("N" and "Q"), achieving an f1-score of approximately 0.93 and 0.72 respectively. The other three classes performed significantly worse, achieving an f1 score of less than 0.35 all the way down to 0.09. Based on this, we can see that SVM did not perform well for the uncommon classes whatsoever.

In regards to the False Positive rates, even the most common class ("N") had a poor false positive rate of 0.57 and a false negative rate of 0.01. This means that the model overestimated the number of occurrences of "N". The other classes, while performing negatively for precision (0.27, 0.28, and 0.76 for example), did not get high false positives. This means that for the uncommon classes, the model was more likely to miss the prediction when it occurred over guessing it was that class, incorrectly. Finally, from the confusion matrix, we can see that the model almost solely predicted class "N", which rarely predicted the less common classes.

Let's look at some real life data: According to the study by Cook, Oh, and Pusic, the median accuracy across all experience levels was approximately 54%. This means, the average likelihood someone with medical experience accurately interprets an EKG is a coin toss. At best, cardiologists only make a 74.9% accurate assessment. Given this information, our LSTM and CNN model performs significantly better than the average cardiologists (who are supposed to be experts in their field), while SVM performs just slightly better for the most popular class (and far worse for the less common classes).

Analysis Of Algorithms

The LSTM model showed great potential and ability in capturing sequential and long term dependencies we see with EKG signals. Looking at its accuracy we saw varied results across the classes, with the best runs achieving 95%+ accuracy, and performs very well on two classes F and N. Q appears to perform well on the ROC, but since there is a large class imbalance here, we should also consider the less rosy PR curve. We would likely still be able to detect these, but with a higher false positive rate. Some of the poorer runs could be due to the way we preprocessed the data and could also be fixed from resampling to cover the data imbalance we mentioned. Furthermore, the heart condition, class 'S', had very few instances in our dataset for us to use therefore affecting performance of the category. While too late now, this does show us the importance of having a well balanced dataset when building ML models. Finally most models showed little ability to discern V. We may need to further augment this data or continue training for longer.

The CNN model performed exceptionally well, especially when combined with the smote preprocessing technique. By using smote, which generates more data for the less common classes, our model was able to achieve a weighted accuracy of 0.95. Additionally, it performs well on 3 of the 5 classes (achieving an f1-score of 95%+ for "N", "F", and "Q") while performing decently on the less common classes (0.8 and 0.88 for "S" and "V"). Based upon the FNR and FPR rates, we can see that the main cause of poor performance for those last two classes was simply a large false negative rate. All of these things combined show the power and efficiency of CNN with smote. Despite being a rather simplistic algorithm, we are able to achieve accuracy scores that meet or beat the scores of a similarly implemented model on other datasets.

SVM, in general, is typically used for more binary classification against smaller datasets. Therefore, when we attempted to apply it to our large dataset with many different classes (all of which were very similar), SVM performed 'just okay'. While it does achieve a weighted accuracy of about 0.78, which still beats the experts according to the study, it is clear that this type of ML model is not suited for our data. The primary issue was the SVM's susceptibility to imbalanced datasets, as the hyperplane naturally attempts to maximize the separation between classes. This can result in smaller classes effectively disappearing to increase separation. This was mitigated by increasing our C value (inverse of acceptable margin size) and reducing our gamma value (influence of proximal points) during the tuning process, however the issue was not completely removed.

Overall, especially compared to practicing physicians, we can see that our algorithms performed exceptionally well, despite the unbalanced dataset. If we were to recommend any model going forward, we would likely suggest others to use CNN with smote, since that model had the best accuracy against the less common classes.

Comparison Of Algorithms

When applying ML algorithms to EKG classification, LSTM, CNN, and SVM all have unique strengths and weaknesses when applied. While the LSTM model performed exceptionally well at capturing the long-term patterns within the sequential EKG readings (achieving an accuracy of up to 98.5% and high precision), it was affected by the imbalance dataset. For example, the "S" class, which was very uncommon, led to a higher false positive rate compared to the other classes. However, especially when compared to the other models, had a significantly simpler pre-processing method, opting for simple denoising.

Comparing this to the CNN, the CNN model stands out as the best model due to the ability to use smote during preprocessing. Because of this, it performed significantly better than the other models for the less common classes, reaching an weighted average of 95% and f1 scores all over 0.80. These scores really show the power of smote when working with unbalanced datasets. While not perfect (CNN still struggled with false negatives for the rare classes), CNN performs much better than LSTM for the rare classes, while being slightly less performant overall. This shows that CNN with smote is likely the most robust option in real-world applications, even though compared to the other models, its preprocessing methods were more complicated and time consuming.

Finally, for SVM, even though it technically was able to reach a max accuracy of 87%, it was more heavily affected by the imbalanced dataset compared to the other models. Similarly to LSTM, it seemed to favor predicting the class "N" over the more rare classes, although in a more extreme manner. However, since it performs so poorly on the less common classes, the model is limited in the real-world applications due to the limitations in multiclass problems and imbalance datasets. When put head to head against the other implemented models, SVM was clearly less suited for the scale and complexity of this task, making it less favorable overall.

For direct comparison between the three, we'll list the top stats in their performant order:

- For the total accuracy (unweighted), we have LSTM (98.5%), CNN (94.6%), and SVM (87%)
- For Validation Loss: LSTM (10%), CNN(14%), and SVM (inconsistent)
- For Best True Positive: SVM (98.5%), CNN (97%), and LSTM (97%)
- Best False Positive: CNN (~0%), SVM (0.29%), LSTM (~0.5%) Here, we can see that each model has different strengths and weaknesses when it comes to choosing the specific data we are looking for.

Next Steps

Next steps would be to adjust the preprocessing of the data, resampling to cover the data imbalances, further tune the LSTM hyperparameters and look into some class balancing methods. Furthermore, since we have shown smote to work with great success in the CNN model, we will look into applying this to LSTM and SVM, if possible. We will also consider modifying the hyperparameters for the 2 new models as well to try and achieve even better results. Finally, we would like to look into other databases to see if these models are consistent with the new data.

References

[1] Z. M. Tun and M. A. Khine, 'Cardiac Diagnosis based ECG Images Classification System using Convolution Neural Network', in 2023 IEEE Conference on Computer Applications (ICCA), 2023, pp. 387–392.

[2] T. M. Ingolfsson, X. Wang, M. Hersche, A. Burrello, L. Cavigelli, and L. Benini, 'ECG-TCN: Wearable Cardiac Arrhythmia Detection with a Temporal Convolutional Network', 2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS), pp. 1–4, 2021.

[3] S. Hochreiter and J. Schmidhuber, 'Long Short-term Memory', Neural computation, vol. 9, pp. 1735–1780, 12 1997.

[4] D. P. Kingma and J. Ba, 'Adam: A Method for Stochastic Optimization', arXiv [cs.LG]. 2017.

[5] B. M. Asl, S. K. Setarehdan, and M. Mohebbi, 'Support vector machine-based arrhythmia classification using reduced features of heart rate variability signal' Artificial intelligence in medicine, vol. 44, pp. 51–64, 1 2008

Contribution Table

Contribution Table

Name	Contribution
Dash	Gantt, GitHub, SVM Code
Arjun	Gannt, Report, Results
Robert	Gannt, Report, Analysis
Pranav	Gannt, Report, LSTM Code
Carter	Gannt, Report, CNN Code

Gantt Chart

Overview

				PHASE ONE		PHASE TWO					PHASE THREE			
TASK TITLE	START DATE	DUE DATE	DURATION	Sep 26	Oct 4	Oct 11	Oct 18	Oct 25	Nov 1	Nov 8	Nov 15	Nov 22	Nov 29	Dec 6
Project Proposal	9/26/24	10/4/24	8											
Model 1	10/14/24	11/8/24	24											
Model 2	11/4/24	11/26/24	22											
Model 3	11/4/24	11/26/24	22											
Evaluation	11/27/24	12/3/24	6											

Full Table

					PHASE ONE		PHASE TWO					PHASE THREE			
					Sep 23	Sep 30	Oct 7	Oct 14	Oct 21	Oct 28	Nov 4	Nov 11	Nov 18	Nov 25	Dec 2
TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	M T W R F S U	M T W R F S U	M T W R F S U	M T W R F S U	M T W R F S U	M T W R F S U	M T W R F S U	M T W R F S U	M T W R F S U	M T W R F S U	M T W R F S U
Project Proposal															
Introduction & Background	All	9/26/24	10/4/24	8											
Problem Definition	Dash	9/26/24	10/4/24	8											
Methods	Pranav	9/26/24	10/4/24	8											
Potential Results & Discussion	Carter	9/26/24	10/4/24	8											
Video Recording	Robert+Arjun	10/3/24	10/4/24	1											
GitHub Page	Dash	10/3/24	10/4/24	1											
Model 1															
Model Selection	All	10/14/24	10/14/24	0											
Data Sourcing and Cleaning	Pranav + Dash + Carter	10/7/24	10/11/24	4											
Data Pre-Processing	Pranav	10/14/24	10/18/24	4											
Model Coding	Pranav	10/21/24	11/1/24	10											
Results Evaluation and Analysis	Dash	11/4/24	11/8/24	4											
Midterm Report	Robert+Arjun	11/4/24	11/8/24	4											
Model 2															
Model Selection	All	11/4/24	11/4/24	0											
Data Sourcing and Cleaning	Pranav + Dash + Carter	10/7/24	10/11/24	4											
Data Pre-Processing	Dash	11/5/24	11/6/24	1											
Model Coding	Dash	11/7/24	11/20/24	13											
Results Evaluation and Analysis	Robert+Arjun	11/21/24	11/26/24	5											
Model 3															
Model Selection	All	11/4/24	11/4/24	0											
Data Sourcing and Cleaning	Pranav + Dash + Carter	10/7/24	10/11/24	4											
Data Pre-Processing	Carter	11/5/24	11/6/24	1											
Model Coding	Carter	11/7/24	11/20/24	13											
Results Evaluation and Analysis	Robert+Arjun	11/21/24	11/26/24	5											
Evaluation															
Model Comparison	All	11/27/24	11/27/24	0											
Presentation	Robert+Arjun	11/27/24	12/2/24	5											
Recording	Robert+Arjun	12/3/24	12/3/24	0											

					PHASE ONE	PHASE TWO	PHASE THREE
Final Report	All	11/27/24	12/3/24	6			