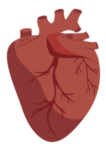
AI Neural Network for ECG Heartbeat Categorization



Bradley Culligan (CLLBRA005) | Michael Scott (SCTMIC015)

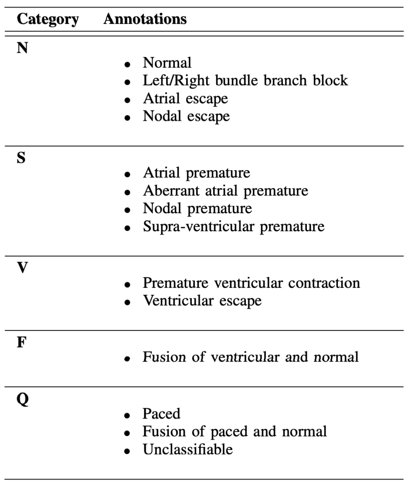
1. Problem Formulation

1.1 Overview

This project aims to develop a neural network for the classification of arrhythmia types from electrocardiogram (ECG) readings. An ECG can be used to test your heart's rhythm and electrical activity [1]. This testing can be used to monitor the functioning of a person's cardiovascular system [2]. An ECG can detect arrhythmia which itself can be further sub-categorised [3]. Thus, we aim to classify arrhythmia types from ECG readings using machine learning.

1.2 Dataset Description

The dataset used comes from research which looked at transfer learning capabilities of ECG classification techniques, so it is already prepared for analysis [2]. The original dataset came from the PhysioNet MIT-BIH Arrhythmia database as a source of labelled ECG records [4]. Each beat was independently annotated by at least two cardiologists [2, 4]. The paper on transfer learning then extracted this data, performed normalisation techniques on it (such as padding and limiting the values to a range of 0 to 1), and created five different beat categories for the data in accordance with the Association for the Advancement of Medical Instrumentation (AAMI) EC57 standard [2, 5]. There are a total of 109 446 samples with a sampling frequency of 125Hz. The five categories can be seen in figure 1.



Category	Annotations
N	• Normal • Left/Right bundle branch block • Atrial escape • Nodal escape
S	• Atrial premature • Aberrant atrial premature • Nodal premature • Supra-ventricular premature
V	• Premature ventricular contraction • Ventricular escape
F	• Fusion of ventricular and normal
Q	• Paced • Fusion of paced and normal • Unclassifiable

Figure 1: Mappings between beat annotations and AAMI EC57 [2]

1.3 Prediction Task

The dataset is used in a supervised, multi-class classification problem that receives an ECG beat in preprocessed form (as per the dataset used in training), and predicts the category of arrhythmia that is present in the input data. The preprocessing is further described in the paper on transfer learning and the code is cited [2, 6]. However, in short, the MIT-BIH dataset contains ECG signals from which beats are extracted. The extraction process splits the continuous ECG signal into 10s windows, selects one of the windows, normalises the amplitudes, finds an appropriate heartbeat period and then pads the selected part with zeros such that there is a predefined, fixed length to the data. An example of the conversion from ECG signal to ECG beat can be seen in figure 2.

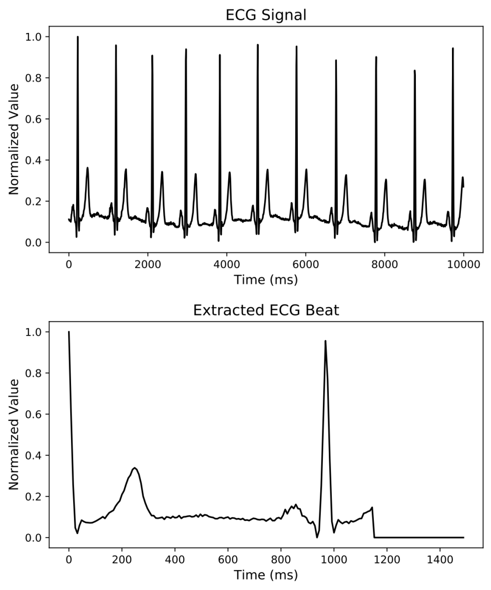


Figure 2: Example of a 10s ECG window with its extracted beat [2]

Thus, the prediction model will analyse an extracted ECG beat of this form and classify the type of arrhythmia present as one of the five categories present in figure 1. These categories are represented in the data as:
[**'N'**: 0, **'S'**: 1, **'V'**: 2, **'F'**: 3, **'Q'**: 4]
N: Normal beat
S: Supraventricular premature beat
V: Premature ventricular contraction
F: Fusion of ventricular and normal beat
Q: Unclassifiable beat
Figure 3 presents a visualisation for a random sample from each class.

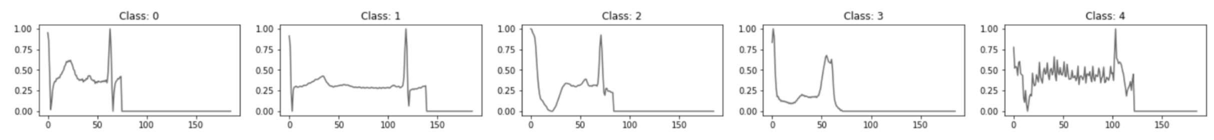


Figure 3: Visualisation of a sample from each class in the data [7]

1.4 Usefulness of the Problem

An ECG is used to assist the diagnosis and monitoring of conditions affecting the heart. It is usually performed to investigate symptoms of a heart problem, such as chest pain, palpitations, dizziness and shortness of breath [1]. For someone already diagnosed with a heart condition or taking medication which may affect the heart, ECG readings may be taken over time to assist with monitoring.

ECGs can detect arrhythmias which is when the heart beats too slowly/quickly or irregularly [1]. Arrhythmias affect all age groups, though some categories of arrhythmias are more common in older people [3]. Arrhythmia risk is increased if you drink excessively, are overweight, or if your heart tissue is damaged due to an illness, such as a heart attack or severe COVID-19 [3]. Further triggers can include tobacco, changes in posture, drinks with caffeine, as well as some prescribed, over-the-counter, and recreational drugs [3]. Thus, it is clear that many people are at risk. The impact of arrhythmia is that your chances of having a stroke are increased by five times and you can be a victim to sudden cardiac death [3]. Yet, with the widespread applicability and harsh impact of arrhythmias, if it is diagnosed early enough, an impacted person may continue to live a normal life [3].

A manual analysis of ECG signals is difficult to detect and categorise different waveforms and morphologies, thus making the process time-consuming and prone to errors [2]. Cardiovascular diseases are the cause of death for approximately one-third of all deaths, globally [8]. This urges the importance of a low-cost and accurate system for diagnosis of arrhythmia. Hence, this project aims to deliver a neural network that is able to deliver the goals of such a system.

There is also an increased usage of wearables which can perform ECG readings. Development of a system which can analyse these readings on the wearable device and provide further classification details of what the reading achieved is desirable. A comprehensive break-down of such a classification is beyond the scope of this project, but the neural network developed for this multi-class classification problem can be used to perform the arrhythmia classification on-device. This goes beyond sending the reading data to a professional and allows a user to freely check their health.

1.5 Potential Difficulties

Before working with the data in building our model, we performed some basic exploratory data analysis (EDA). In doing so, we noticed that the number of samples in each category were vastly different. Hence, we are working with an imbalanced dataset. This is visualised in figure 4. The category with the most weight is the *normal* category with more than 80% of the samples, whilst the smallest category is the *fusion* category. This makes it relatively easy to achieve a high accuracy in the model by simply choosing the most dominant class at all times. However, this is a poor model as it will never predict any of the other classes. Therefore, we shall use the F1-score (a function of the model's precision and recall) when evaluating our models in order to take this into account.

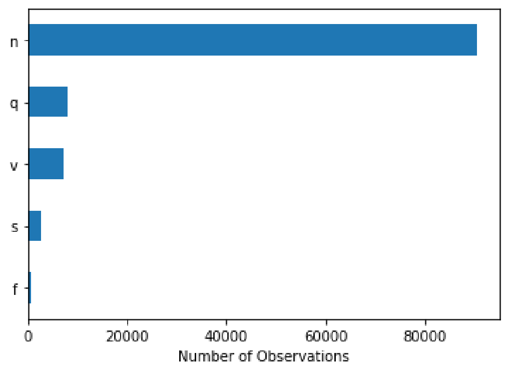


Figure 4: Distribution of classification samples in the dataset

While the full dataset size is sufficient for training a neural network, we notice that the smallest category does not have enough sample points to train an accurate model if all categories were limited to that sample size. This is important because of our imbalanced dataset. One approach we could take to fix the imbalance would be to limit the number of samples in each category to the number in the smallest category. This would cause the problem of a small sample size for training. Instead, we apply re-sampling techniques in order to balance our dataset and enlarge the smaller categories with class sizes of 8000 samples.

Beyond these considerations, the data has already been cleaned and normalised. There are no nulls or anomaly data points. Thus, the dataset is already set up for use in the development of a machine learning model. All that remains is to split the data into training, validation, and testing subsets which we apply in a 60-20-20 split.

1.6 Ethics

The MIT-BIH Arrhythmia database is an open access dataset licenced by with the Open Data Commons Attribution Licence v1.0 and distributed through PhysioNet - a repository of freely available medical research data, managed by the MIT Laboratory for Computational Physiology [4, 9]. Thus, use of the dataset for training of our model was allowed.

The use of this neural network for its purpose does pose an ethical discussion, though these concerns can be mitigated. Ultimately, the data used when applying the model is medical data so it should be treated with confidentiality. We make the assumption that its use in a professional medical setting means that medical, ethical clearance to use the data would have been obtained. Further, use of the model on a wearable device should require that classification happens on the device for data confidentiality reasons. Any sending of data should also be encrypted.

Finally, medical classifications that can have life altering effects should be handled carefully. An incorrect classification may have a negative health impact on individuals. Thus, it is important that the model chooses an appropriate evaluation metric and that it more frequently chooses to tell the user to validate their results with a medical professional in order to mitigate the risk of incorrectly classifying a user’s arrhythmia as benign when this could negatively impact their health. In general, this tool should be used to assist professional, doctoral analysis of the results and not only be solely used by the user.

2. Baseline

We now look at the simplest possible approach to the problem in order to set some baseline metrics which we can later compare with our model. Two possible baselines are chosen: for a naive or heuristic baseline, we can always predict the highest frequency class; for a shallow ML baseline, we can use naive Bayes.

2.1 Highest Frequency Class

Due to the imbalance of the dataset, the simplest prediction model would be to always predict the highest frequency class - the *normal* class. This would result in a high accuracy metric, but the model would never predict the other arrhythmia classes. As a result, we would obtain a low F1-score. This metric is more important to verify the validity of our model. Indeed, after running this baseline model, we see that the results achieved are in line with what we predicted. Running this baseline model outputted the following results (rounded to two decimal places):

Baseline Validation Accuracy: 82.68%
Baseline Validation F1 Score: 18.10%
Baseline Test Accuracy: 82.94%
Baseline Test F1 Score: 18.13%

2.2 Naive Bayes

As a shallow machine learning baseline, we can apply a naive Bayes classifier to our dataset in order to predict one of the five arrhythmia classes. We expect this to perform poorly because the naive Bayes makes the assumption that every pair of features being classified is independent of each other. Here, the features are the time series electrical readings which aren't all independent. Indeed, after running our naive Bayes model, we obtain the following results (rounded to two decimal places):

Naive Bayes Validation Accuracy: 71.74%
Naive Bayes Validation F1 Score: 38.73%
Naive Bayes Test Accuracy: 71.62%
Naive Bayes Test F1 Score: 37.94%

3. Model Design

Here we discuss the input representations and the various model architectures that were considered in designing our model.

The input data is a set of voltage readings at the sampling rate, i.e. time series data. There are 187 samples in an input, though this may be a padded result with trailing zeroes. This was done to ensure a consistent input size. Similarly, the amplitude values are normalised to be between 0 and 1. For our neural network, we choose to have each sample point represented as its own neuron in the input, resulting in 187 input neurons. We also have five different output classes which we represent in our model with 5 output neurons.

The model architecture and hidden layer structure require a deeper analysis. The time series nature of our data might imply that one would want to choose a Recurrent Neural Network (RNN) architecture. However, this isn't entirely appropriate because the model operates on segmented heartbeat samples. This would use a bidirectional RNN to process the whole segment at one time, resulting in a large number of parameters and training times. Indeed, an ECG classifier was found comparing the performance of using an RNN and a Convolutional Neural Network (CNN) for this classification problem [10]. While the performance was comparable, the training times and parameters required meant that the model was less ideal than alternatives.

Instead, we focus on the CNN architecture chosen in the transfer learning paper originally used when attaining the dataset [2]. A CNN is an appropriate architecture choice because it can detect features from sequences of the 1D observations and map these internal features to different classification types. Importantly, this is one of the reasons for extracting the single beat reading from the full ECG. The CNN structure then enables spatial generalisation along the time dimension.

To design the hidden layer structure, we made mixed use of the heuristics from various Kaggle notebooks which looked at ECG classification with CNNs, as well as the model originally proposed in the transfer learning paper [2, 7, 11, 12, 13]. Together, these models allowed us to compare various architectural choices without spending time training the models ourselves. It also allowed us to gain confidence in our final architectural choice.

Our architectural choice was to have the input neurons connect to three 1D convolutional layers with filter and kernel size combinations of (32, 3), (64, 3), and (128, 5). This was connected to a 1D max pooling layer with a pool size of 3 and stride size of 2. The output is then flattened before we add the final three layers that are densely connected with 512 and 1024 neurons, before the output layer with the 5 output neurons as stipulated at the start of this section. Finally, we used ReLU activation on all hidden layers for our non-linearity activation function and to handle the vanishing gradient problem. The output layer used a softmax activation function to obtain a probability distribution over the output classes.

This model architecture is in-line with what would be expected of the dataset - the data has very little noise and a relatively complex structure for a 1D object. This would imply that a deep model would be designed which correlates to the model we have chosen. The split between having a CNN at the start of the network and then a multi-layer perceptron at the end is also appropriate because we are able to extract structural features of the data and then apply classification on the extracted features.

While this model architecture has made use of various prior CNNs in comparing different architectures, it is still a baseline neural network for the problem. The model was run to see how performance fared, but a full hyperparameter search is still in place to optimise the model. Further, the model's evolution will also be compared to our previous baseline models of the naive Bayes classifier and the model which always chose the highest frequency class.

4. Model Validation

In designing our network model and looking at various architecture choices, we analysed various models that had already been designed for the classification of ECG arrhythmia. This allowed us to compare the architectural and feature choices available without having to spend time manually training the various models. The design we chose, as stipulated in the model design section, provided high enough evaluation metrics that we decided to not search alternative structures further. This would only add to training time which we instead dedicated to only our hyperparameter search.

We employed a grid search to find the optimal set of hyperparameters. Again, to reduce training times, a grid search is sufficient to enumerate a given set of values for each hyperparameter without requiring a full search over all hyperparameter choices. The grid search takes a sample of hyperparameter combinations, trains the network with those hyperparameters, and then records the achieved results. Importantly, we want to maximise the evaluation metric of the validation F1-score. To achieve this, the recorded results from each trained network include the accuracy, precision, and recall. We also used early stopping criteria on the grid search to try to reduce the training times required, as well as to avoid overfitting. Table 1 outlines the search space for each hyperparameter that was involved in our grid search.

Hyperparameter	**Values Tested**
Activation function	[‘ReLU’]
Dropout	[0, 0.1, 0.25, 0.5]
Learning rate	[0.01, 0.005, 0.001, 0.0005]
L2 penalty	[0, 0.001, 0.01, 0.1]

Table 1: Hyperparameters grid search parameter and value choices

Google Cloud Services allowed us to train and compare this vast number of networks in a timely manner. The model with the highest validation F1-score was chosen as our optimal model and saved for later use. We load in this model for use in the rest of the project, but having a separate save also assists with reproducibility so the reader may load in the model themselves. In general, the model had already attained relatively good accuracy without hyperparameter searching. However, as discussed in the ethics section, the model is being used for medical classification which urges the importance of an accurate model. Thus, it was still important to look at the various hyperparameter configurations in order to aim for a neural network which would minimise the risk of misclassification, and hence better our F1-score.

5. Evaluation

This section provides details on the evaluation metric(s) that were used, the choice of training/validation/test split of the data, and we report the results achieved.

5.1 Evaluation Metrics

A popular evaluation metric for classifiers is *accuracy*. This supplies the percentage of classifications that were correct. However, due to the imbalance in our dataset, it would not be appropriate to solely rely on accuracy as our evaluation metric. This is because a high accuracy value may be due to the model more commonly choosing the higher frequency classes. Such an approach would allow the model to achieve a high accuracy because it consistently classifies the majority of classes correctly. However, the less represented classes fail to be classified and, thus, this model is poor.

To solve this, we can instead use the F1-score as our evaluation metric. The F1-score is a function of the *precision* and *recall* metrics. Precision attempts to answer the question: "What proportion of positive identifications was actually correct?", whilst recall attempts to answer the question: "What proportion of actual positives was identified correctly?" [14]. The function of these produced by the F1-score can provide a better understanding of how our model is performing with regards to the imbalanced dataset. The F1-score is calculated for each class, with the overall model performance then achieved by taking a weighted average of each class' F1-score. We aim for this value to be equal to 1.

5.2 Dataset Split

The original dataset was provided already in a split form of approximately 70% training data and 30% testing data. However, we did not like this dataset split and wanted a validation set to use for our hyperparameter training. To achieve this, we joined the original datasets together and then performed a 60-20-20 split for training, validation, and testing, respectively. This means that the majority of our data is used for training with equal size splits for validation and testing. Validation data was used to test the trained model on each hyperparameter configuration - emulating how the model would later be tested but with a common dataset that would allow us to compare performance differences without causing data leakage. The test dataset then simulates an unseen dataset test to get an unbiased indication of how the model performs on unseen data and how well it is able to generalise - i.e. it is used to test how well the model has learnt.

It should be noted that the validation and testing datasets kept their original class distribution in order to simulate the imbalanced nature of the data for out-of-sample applications of our model. However, the training dataset used resampling techniques in order to better balance the model's exposure to the various classes that it should predict. This allows the model to strengthen its view of feature extraction to all classes.

5.3 Results

The choice of hyperparameters from our hyperparameter search is as follows:
Activation function: ReLU
Dropout: 0
Learning rate: 0.0005
L2 penalty: 0

Looking at the results from our hyperparameter search shows us that the model was mostly affected by the L2 penalty, then the learning rate, and then the dropout values. This can be seen when sorting the search values by the validation F1-score which we used to decide on the optimal set of hyperparameters. All of the better performing models had an L2 penalty of 0 or 0.001 which were our two lowest options. Similarly, the learning rate values that performed better were also our lowest values - 0.0005 and 0.001. Finally, the dropout values generally fluctuated which shows that they didn't have as significant of an effect compared to the L2 penalty and learning rate, but we can still notice a trend of lower dropout values matching to better performance when paired with the L2 and learning rate values.

Table 2 provides the accuracy and F1-score achieved in validation for each of our designed models - baseline (highest frequency class), naive Bayes, unoptimised (original) and optimised (using the found hyperparameter values) neural network (NN).

	Baseline	**Naive Bayes**	**Original NN**	**Optimised NN**
Accuracy (%)	82.68	71.74	97.12	97.25
F1-Score (%)	18.10	38.73	29.19	87.38

Table 2: Validation accuracy and F1-score for the four models tested

From these results, we can see that although the accuracy in the baseline model was higher than the naive Bayes model, the F1-scores are dramatically different. This shows that the naive Bayes didn't simply rely on the highest frequency class to ensure a high accuracy, but actually applied some learning in order to double its F1-score over the baseline highest frequency model. The neural networks then both improved their accuracy when compared to the two baseline models but each neural network has very different F1-scores. The original network had a poorer F1-score than the naive Bayes model, even though its accuracy was increased. This is largely due to its poor precision which weighed down the F1-score. Comparatively, the model had a low precision and high recall so it returned many positive results, though often incorrect. In general, this needed work but applying such a model under doctoral supervision means that the doctor would more likely be requested to validate the results. After performing our hyperparameter search, we achieved a model with relatively similar accuracy, but the F1-score increased significantly and performed the best of all of the models compared. This was an important gain for our model to make in analysing medical data as it implies that the model should now perform much better in making correct classifications, even when applied to infrequent classes.

6. Model Performance Analysis

We now analyse the final model that was designed and trained by looking at the results achieved on the test set. The final model uses the hyperparameters found in the hyperparameter search. We also provide any shortcomings and conclusions of the project.

6.1 Tested Model Results

The test dataset provides an indication of how the final model will perform when applied to unseen data. Regarding the test dataset, the final model achieved an **accuracy** of **97.19%**, and an **F1-score** of **86.84%**. These values are in-line (slightly lower but not statistically significant) with the validation metrics achieved which implies that the test and validation datasets are fair comparisons, i.e. the spread of class classifications were similar allowing for all classes to be involved in testing and validation. Further, as discussed in the evaluation of the validation metrics for this model, these values show that the model has indeed learnt to identify structure in the data and can be used to assist with analysis.

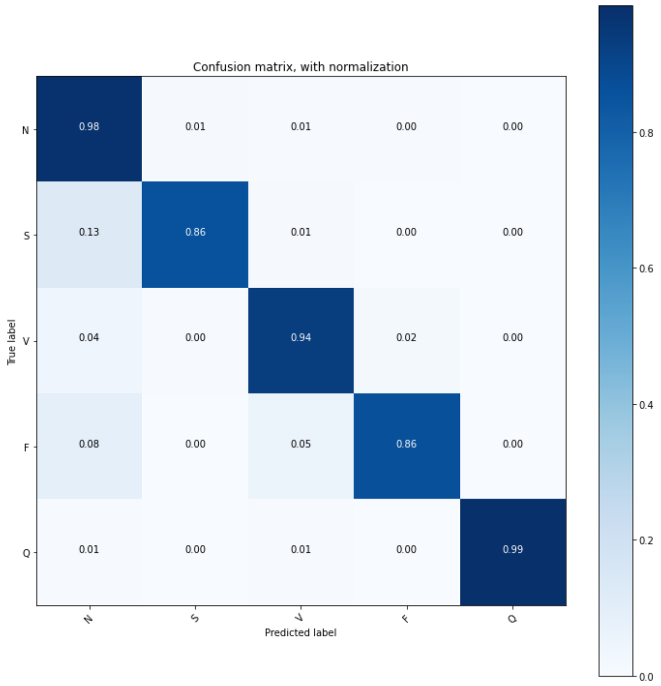


Figure 5: Confusion matrix for test set of arrhythmia classification using a neural network

Figure 5 presents the confusion matrix achieved with our final model. This is a graphical representation of the percentage comparisons of each true label and predicated label proportions achieved. In other words, it presents the percentage that the true label was predicted along the main diagonal, and presents the percentage that an alternate class was predicted incorrectly for a true label. Noticeably, the main diagonal has relatively high percentages showing that the model typically predicts the true label correctly. The F and S classes had lower percentages compared to N, V, and Q. This could be due to the fact that the F and S classes had the smallest size, thus, when resampling, repeated input values would have been taken which meant the training data was less varied for the model to learn how to extract a larger range of features. It could also be due to the fact that the F and S classes are related to other classes. For example, the fusion class is a mix of the ventricular (V) and normal (N) beat classes. To further strengthen this argument, notice that the true label F row would have incorrect predictions for the N and V classes which it is defined to be a mix of. The N column also shows that the model favoured making predictions of the N class when the true label was something else. This may be due to the large variety of the N class inputs which were originally the largest class in the imbalanced dataset. Thus, the model has learnt to favour predicting class N because of its high accuracy metric achievable when acting this way.

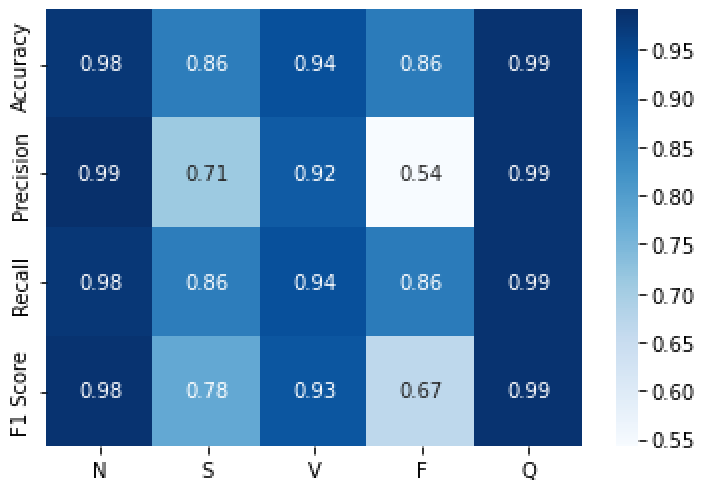


Figure 6: Heatmap of evaluation metrics per classification class

Figure 6 now further breaks down the model's performance on the test dataset. Here we can analyse the accuracy, precision, recall, and F1-score of each class individually to see how they each affected the model's performance. Importantly, we can visualise the precision and recall separately which constitute the F1-score and, thus, allows us to more accurately understand why certain F1-scores were achieved.

Similar to the confusion matrix, we notice in general that the majority sized classes (and therefore the classes with the most variety of input after resampling) had the overall better performing metrics. The F and S classes decreased the model's overall F1-score due to their own low F1-score values. These classes both had low F1-scores due to their low precision values. As can be seen in figure 5, the model would often falsely predict the F and S classes to be of class N. This is of concern because it means that the model is telling the user that their heartbeat is normal when they actually have a type of arrhythmia. With this in mind, it is clear that the model (at this stage) should only be used to assist with professional, medical analysis and not be used on its own. This allows a doctor to notice these mis-classifications and rectify the situation. The high recall values show that we have correctly handled the imbalanced dataset.

6.2 Shortcomings and Conclusions

Based on the analysis of our final model with the test dataset, a few shortcomings have come to notice. Even though we had undertaken resampling measures to account for the imbalanced dataset, we still obtained a situation where the model would more frequently predict class N (the more common class in the original dataset) and would more commonly classify classes F and S (the least common class in the original dataset) incorrectly. With the high recall values that we were able to achieve, we are led to believe that the only way to mitigate this issue is to obtain more data on the less common classes, F and S. In general, the amount of data available was large, but after reducing the size to handle the class imbalance, it proved that more data was needed.

We also notice that although the recall values are relatively high, they definitely have room for improvement. Further hyperparameter searching can assist with this concern. Noticeably, our final model's hyperparameters were ultimately set to values to ignore regularisation. This may indicate that more optimal parameters exist but we didn't find them because we had to perform a sparse hyperparameter grid search due to the amount of training time available. With more time, we could employ a deeper set of hyperparameters for searching and also apply various model architecture designs with that hyperparameter search.

To conclude, the model designed achieves an overall good performance. It is able to generalise well to unseen data and typically makes the correct classification. With the shortcomings discussed, there is a way to attempt at bettering the model so it may be used as a sole classification mechanism. However, in its current state, the model is not able to perform the classification accurately enough alone and should instead be used to aid medical professionals in arrhythmia classification.

7. Software

This project was developed in Python, making use of various Python packages to assist with development. We also used Google Cloud Services with the N2 machine series to train the neural networks, and Jupyter notebooks to quickly run code and analyse the results. Python packages used included: NumPy for numerical computations, Pandas for data manipulation and analysis, Matplotlib for visualisations, Sci-Kit Learn for many of the metric and utility functions as well as the baseline classifiers. Tensorflow and Keras were used for the actual neural network development. Additionally, some code excerpts from the cited Kaggle notebooks provided visualisations of the metrics which we chose to use in this project to represent our findings.

8. Reproducibility

To reproduce the results obtained in this paper, the full set of data and working code have been attached. This can be used to ensure that the experiments and results claimed in this paper are reproducible. Random seeds were used where necessary to ensure a consistent randomness in this project for reproducibility. The final, trained model produced has also been saved so that it can be imported and tested against test data. This eases the process of checking reproducibility without having to wait for the full training process. Test results and hyperparameter configurations have been reported in this paper. Finally, a README is attached with this document to assist with running the full process, as well as running the testing suite with the saved model.

References

1. https://www.nhs.uk/conditions/electrocardiogram/
2. https://arxiv.org/pdf/1805.00794.pdf
3. https://www.nhs.uk/conditions/arrhythmia/
4. https://www.physionet.org/content/mitdb/1.0.0/
5. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6260536/
6. https://github.com/koen-aerts/ECG_ML/blob/master/02_import_mitdb_data.ipynb
7. https://www.kaggle.com/code/l33tc0d3r/ecg-heartbeat-categorization-using-cnn
8. http://www.who.int/mediacentre/factsheets/fs317/en/
9. https://www.physionet.org/
10. https://github.com/dave-fernandes/ECGClassifier
11. https://www.kaggle.com/code/gregoiredc/arrhythmia-on-ecg-classification-using-cnn
12. https://www.kaggle.com/code/coni57/model-from-arxiv-1805-00794
13. https://www.kaggle.com/code/l33tc0d3r/ecg-heartbeat-categorization-using-cnn/notebook
14. https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall