Classification of arrhythmias from single ECG signals using Deep Neural Networks

Alessio Guarachi

05 December 2024

Contents

Abstract

I developed a Convolutional Neural Network(CNN) to classify 5 rhythm classes using 109,446 single-lead ECGs from 57 patients who used a single-lead ambulatory ECG device. When validate against an independent test dataset annotated by a consensus committee of cardiologists, the CNN achieved an average area under the receiver operating characteristic curve (ROC) of 0.97. The average F1-score, which is the harmonic mean of precision and recall, was 0.83. These findings demostrate that a deep learning approach can classify a range of distinct arrhythimas from single-lead ECGs with high diagnostic performance.

Chapter 1

Introduction

1.1 Electrocardiogram

The electrocardiogram (ECG) is important in the clinical cardiology domain since the features of ECG can provide diagnostic bases for cardiac arrhythmia. ECG can reflect changes in the electrical activity of the heart.

1.1.1 ECG cycle details

As shown in Figure 1.1, a typical ECG consists of different waves, that represent the depolarization and repolarization of specific parts of the heart.

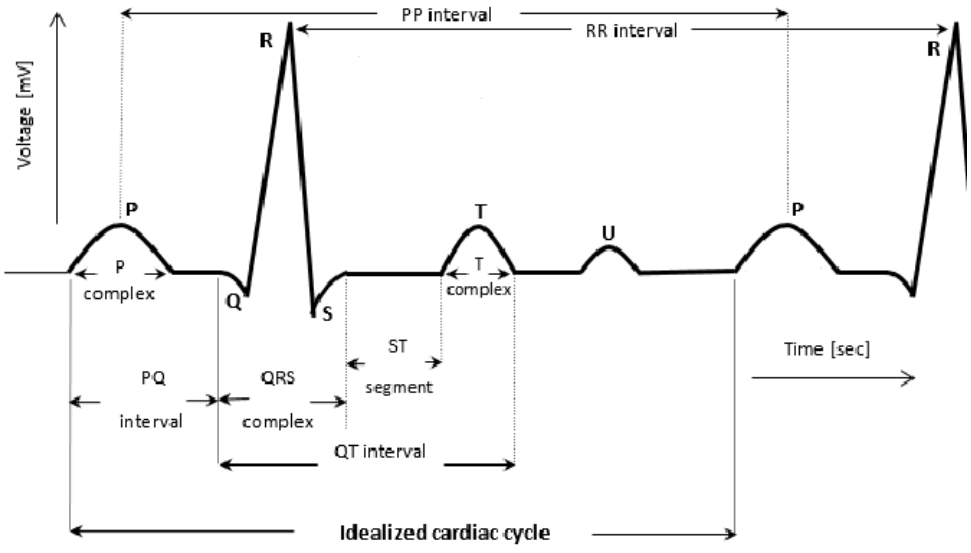

Figure 1.1: ECG cycle details.

1.2 Cardiac Arrhythmia

Cardiac arrhythmias refere to the abnormality and irregularity of the rhythm of the heart. Severe arrhythmias may cause irregular heart contraction which can rising the risks of stroke or death.

1.2.1 Supraventricular ectopic beats

Extra beats that arise from the top chambers of the heart (atria) are called supraventricular ectopic beats (or premature supraventricular ectopic beats). [3]

1.2.2 Ventricular ectopic beats

Extra beats arising from the lower chambers (ventricles) are known as ventricular ectopic beats (or premature ventricular ectopic beats). [3]

1.3 Automatic interpretation of ECG

Computer-aided interpretation has become increasingly important in the clinical ECG workflow since its introduction over 50 years ago, serving as a crucial adjunct to physician interpretation in many clinical settings1. However, existing commercial ECG interpretation algorithms still show substantial rates of misdiagnosis.

1.4 Project Overview

The two main aims of the project are:

1. To provide the base for the hyperparameter tuning of the model.
2. To compare the performance of the model with the existing models.

Chapter 2

Models and Methods

2.1 Dataset

This project utilizes the MIT-BIH Arrhythmia Dataset[4], which contains ECG recordings collected by the BIH Arrhythmia Laboratory between 1975 and 1979. The dataset contains 48 half-hour two-channel ECG recordings, each sampled at 360 Hz. The recordings were digitized at 11-bit resolution over a $10mV$ range. Two or more cardiologists independently annotated each record, and disagreements were resolved to obtain the final set of annotations. The dataset contains 109,446 annotated beats from 48 recordings, with 5 classes of beats:

1. Normal (0)
2. Supraventricular ectopic (1)
3. Ventricular ectopic (2)
4. Fusion of ventricular and normal (3)
5. Unclassifiable beats (4)

In figure 2.1 is shown the distribution of the classes in the dataset. From the figure it is possible to notice that the dataset is imbalanced.

2.2 Pre-processing

The following pre-processing was performed:

1. The dataset was split into training, validation, and testing sets.

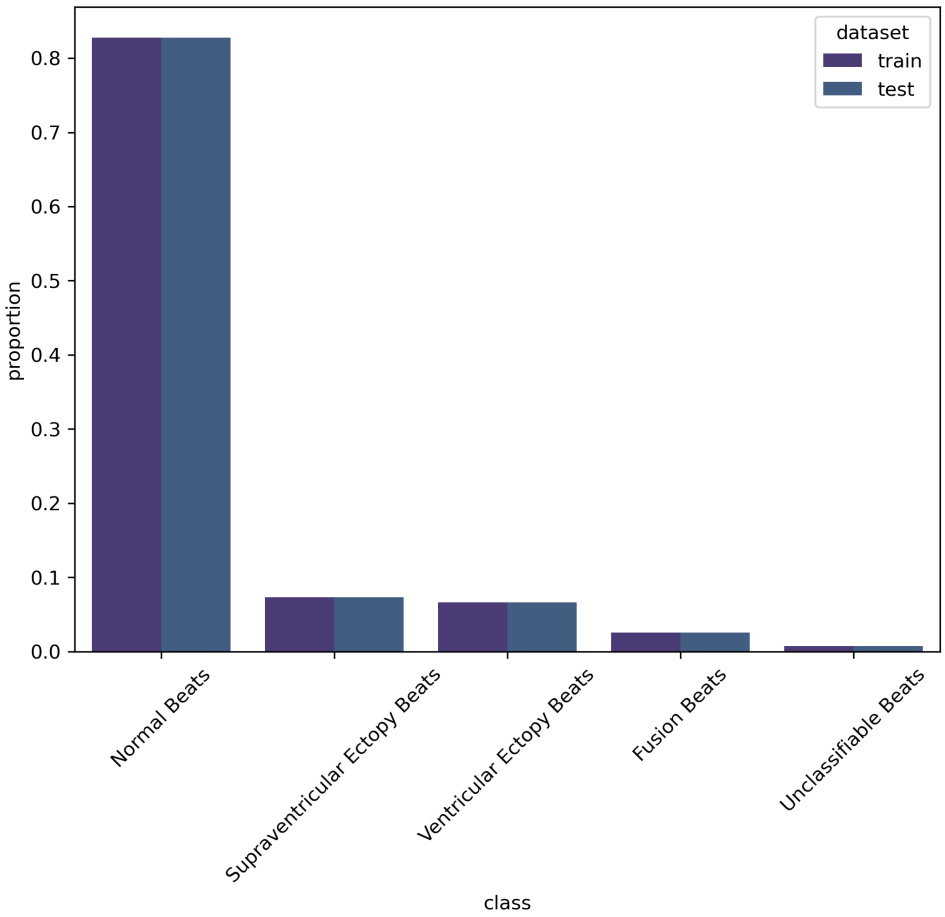

Figure 2.1: Distribution of samples across classes.

2. Normalized the ECG signals.

The splitting of the dataset was done in a manner to ensure the same distribution of classes in each set. The validation set was obtained from the test set, and has the same number of samples.

The ECG signals were normalized using the StandardScaler from the scikit-learn library, which scales the data to have a mean of 0 and a standard deviation of 1. This can improve the performance of the model by ensuring that the features are on the same scale.

2.3 Convolutional Neural Network

The Convolutional Neural Network (CNN) is a type of neural network that is well-suited for image classification tasks, but can also be used for time-series data such as ECG signals.

2.3.1 Model Architecture

The considered CNN architecture consists of the following layers:

1. three 1D convolutional layers with ReLU activation functions,
2. three max pooling layers,
3. two dropout layers,
4. two dense layers with ReLU activation functions,
5. a final dense layer with a softmax activation function.
6. a flatten layer.
7. a batch normalization layer.

2.3.2 Model training

The proposed CNN model is implemented and evaluated on Keras 2.3.0 based TensorFlow 2.1.0 while running on a T4 GPU provided by Google Colab and Kaggle.

To address the imbalanced dataset, the model was trained using a weighted categorical cross-entropy loss function, ensuring that underrepresented classes were given more importance during training.

Early stopping with a patience of 7 epochs was employed to prevent overfitting, halting training if no improvement in validation loss was observed for 7 consecutive epochs. The model was trained with a batch size of 128 samples over a maximum of 45 epochs.

2.3.2.1 Training details

The model was trained using the Adam optimizer with a learning rate of 0.001. To improve convergence and training stability, learning rate scheduling was applied.

2.3.2.2 ReduceLROnPlateau

After seeing that the loss over the epochs were unstable only using Adam I decided to use the ReduceLROnPlateau callback.

1. factor = 0.5, 0.7
2. patience = 3
3. min_lr = 1e-7

2.3.3 Hyperparameter Tuning

The hyperparameters of the CNN model were tuned using the Weight and Biases library [5], which allows for easy tracking of hyperparameters and metrics. The hyperparameters that were tuned include the number of filters in the three convolutional layers, number of neurons in the Fully-Connected layer, learning rate, and dropout rate. The hyperparameters were tuned using the Bayesian Optimization algorithm, which is a probabilistic model-based optimization algorithm that uses a surrogate model to approximate the objective function.

We defined the following search space for the sweep:

- **Layer 1 Size:** [64, 128, 256, 512]
- **Layer 2 Size:** [64, 128, 256, 512]
- **Layer 3 Size:** [32, 64, 128, 256]
- **Fully Connected Layer Size:** [64, 128, 256, 512]
- **Number CNN layers** [4, 6, 8]
- **Number of filters from the 4th layer** [16, 32, 64, 128]
- **Dropout Rate:** [0, 0.1, 0.25, 0.5]
- **Learning Rate:** log-uniform distribution between 1×10^{-6} and 1×10^{-3}

The goal was to minimize the validation loss during the training process.

In total were performed 40 iterations of hyperparameter tuning, with the best model being selected based on the validation loss. The best model was then evaluated on the test set to obtain the final performance metrics.

For the results of the hyperparameter tuning, see the Appendix section A.2.

2.3.4 Final training

Once obtained the best hyperparameters, the model was trained on the training set. The training wasn't mixed with the validation set since the number of epochs wasn't fixed, due to the use of early stopping.

2.4 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a special class of neural networks that are commonly adopted to process time series data or sequential data. To process sequential data,

RNNs contain an internal memory to store information of previous inputs that combine with the current inputs for generating the next output of the sequence.

The different sets were reshaped to have 3D tensors.

2.4.1 Model Architecture

The model is composed by the following layers:

1. LSTM layer
2. Dropout
3. Dense layer

2.4.2 Initial experiments

In the initial phase of model development, I started with a simple architecture consisting of two LSTM layers. The primary motivation for this choice was to capture temporal dependencies in the sequential data, as LSTM (Long Short-Term Memory) networks are well-suited for this task.

To prevent overfitting, I added Dropout layers after each LSTM layer. Dropout is a regularization technique that helps to reduce the likelihood of the model overfitting to the training data by randomly setting a fraction of input units to zero during training.

I also introduced the Batch Normalization layer after the LSTM layers to normalize the activations of the previous layer at each batch. This can help to stabilize and accelerate the training process.

2.4.3 Model Training

The model was trained using the Adam optimizer and the ReduceLROnPlateau callback to reduce the learning rate when the validation loss stopped improving. The model was trained with a batch size of 128 samples over a maximum of 45 epochs.

2.4.4 Hyperparameter Tuning

For the hyperparameter tuning of the RNN model, I can use a fixed number of layers, as done in the CNN model, or I can use a sweep to tune the number of layers and the number of units.

We defined the following search space for the sweep:

- **Number of Units:** [16, 32, 64, 128, 256]
- **Number of Layers:** [1, 2, 3, 4, 5, 6, 7, 8]
- **Fully Connected Layer Size:** [64, 128, 256, 512]
- **Dropout Rate:** [0, 0.1, 0.25, 0.5]
- **Learning Rate:** log-uniform distribution between 1×10^{-5} and 1×10^{-2}
- **RNN Layer Type:** [LSTM, GRU]

The goal was to minimize the validation loss. We utilized Bayesian optimization for the sweep to efficiently explore the hyperparameter space.

2.4.5 Final Training

The model was trained using the best hyperparameters obtained from the sweep. The training wasn't mixed with the validation set since the number of epochs wasn't fixed, due to the use of early stopping.

Chapter 3

Results

The classification performances can be comprehensively evaluated by confusion matrix, precision, recall, F1-score, Precision-Recall curve, and area under the curve (AUC). See Appendix A for more information regarding those metrics.

These evaluated measures are calculated by the following equations:

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{3.1}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{3.2}$$

$$F1score_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{3.3}$$

TPi and TNi represent the number of correctly predicted positive and negative samples, respectively. On the other hand, FP and FN are the values of false prediction for positive and negative samples separately.

3.1 Convolutional Neural Network

3.1.1 Confusion matrix

See figure 3.1. for the confusion matrix of the CNN model.

It is possible to notice that ....

3.1.2 Classification report

The classification report is a report showing the main classification metrics. As shown in Figure 3.2, the CNN model achieved an average F1-score of 0.82. The F1-score for

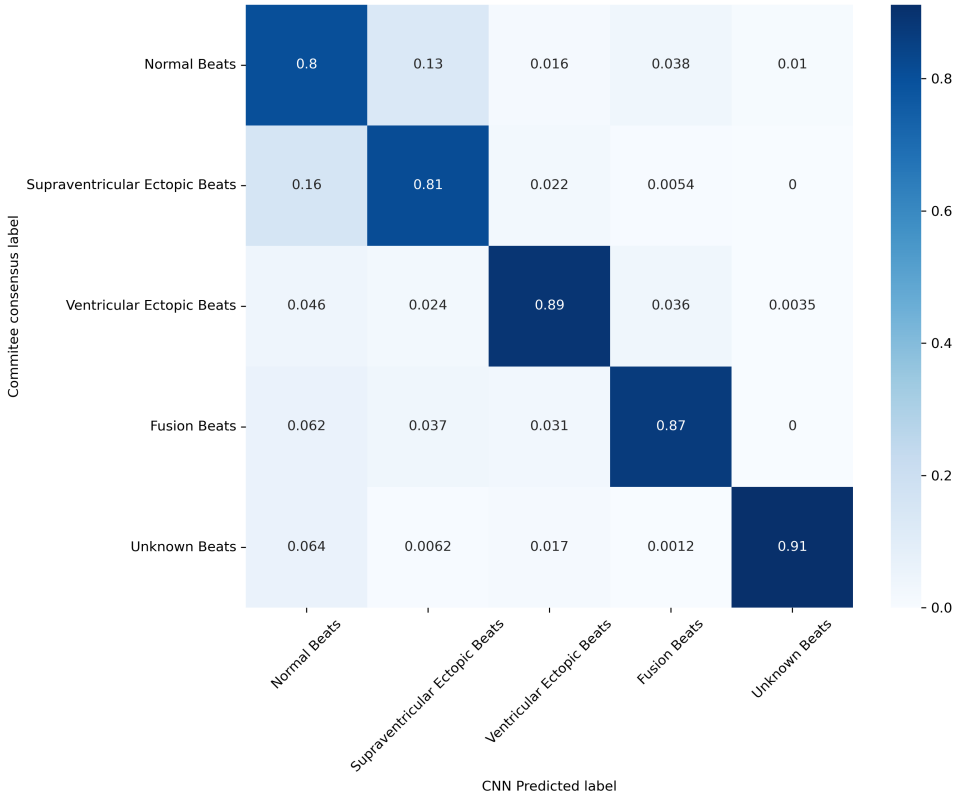

Figure 3.1: Confusion matrix of the CNN model. The percentage of all possible records in each category is displayed on a color gradient scale.

each class is shown in the figure. The model achieved the highest F1-score for the class 'Normal' (0.88) and the lowest F1-score for the class 'Supraventricular ectopy beats' (0.26).

3.1.3 Precision-Recall curve

Considering that the dataset is umbalanced, the Precision-Recall curve is a better metric to evaluate the model performance. The Precision-Recall curve is shown in figure

The model achieved an average AUC of 0.85.

3.1.4 Error Analysis

In order to understand the mistakes of the model, the limitations of the model, nine signals were randomly selected from the misclassified signals. The misclassified signals are shown in Figure 3.3. This figure can allow us to understand the limitations of the model and the potential improvements that can be made. Future work will consider also the contribution of cardiologists, to better analyze the mistakes of the model.

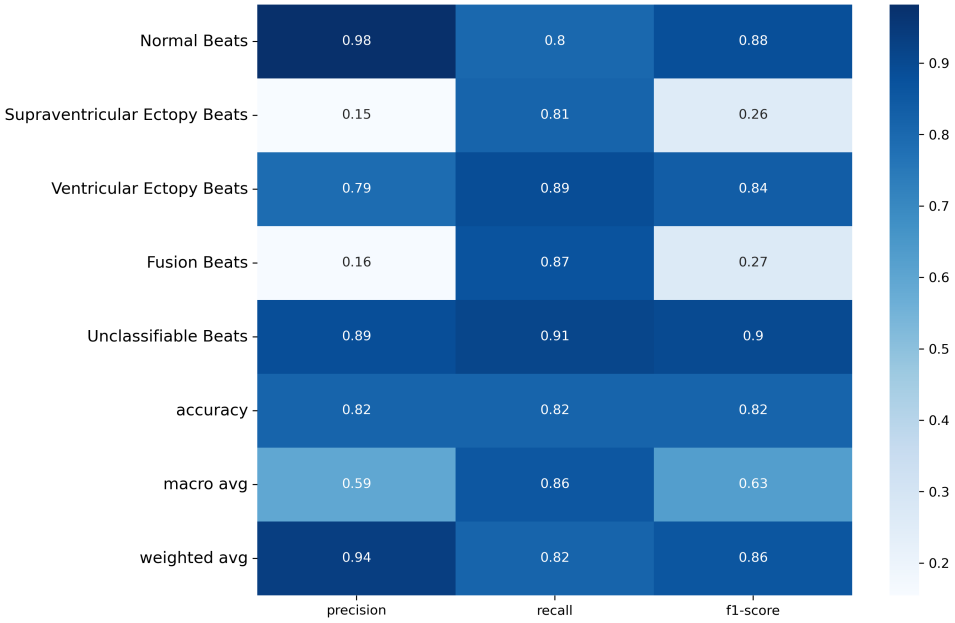

Figure 3.2: Classification report of the trained CNN model.

3.2 Recurrent Neural Network

3.2.1 Confusion matrix

3.3 Comparison

Considering that the trained model could be used in a wearable device to detect arrhythmias. In this case the cost of False positive is lower with respect to False Negative, the classification metric that has the highest importance is Recall.

The trained CNN and RNN model were compared with the existing models.

1. F1 score across the classes
2. Precision across the classes
3. Recall across the classes
4. average AUC of the Precision-Recall curve

3.3.1 Recall

From Figure 3.4, it is possible to notice that the CNN model achieved the highest recall for the class 'Supraventricular Ectopy Beats' (0.84) and the lowest recall for the class 'Supraventricular ectopy beats' (0.26). The RNN model achieved the highest recall for the class 'Normal' (0.94) and the lowest recall for the class 'Supraventricular ectopy beats'

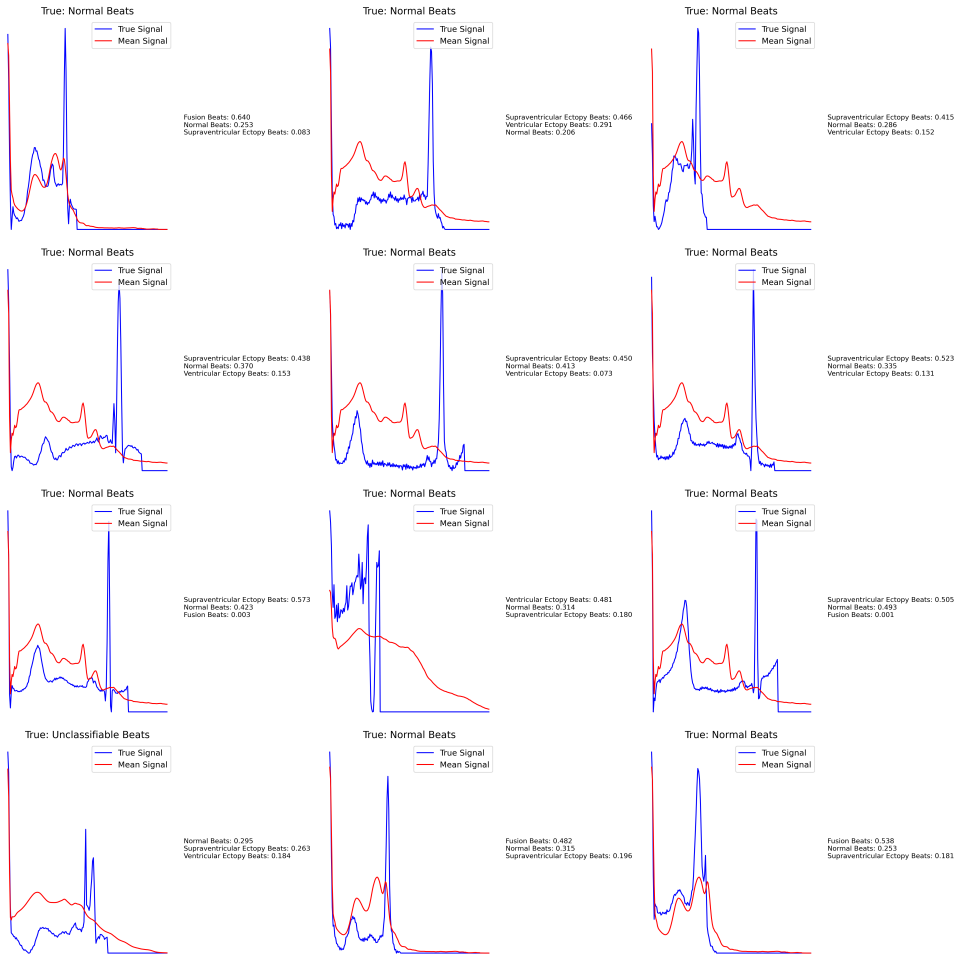

Figure 3.3: Nine misclassified signals provided by the trained CNN model.

(0.26). The existing model achieved the highest recall for the class 'Normal' (0.94) and the lowest recall for the class 'Supraventricular ectopy beats' (0.26).

3.3.2 Precision

A low precision for an arrhythmia detection model signifies that a significant proportion of the instances that the model predicts as arrhythmias are actually not arrhythmias. In other words, there are many false positives.

3.3.3 F1 score

See figure 3.6 for the comparison of F1-scores across the different classes.

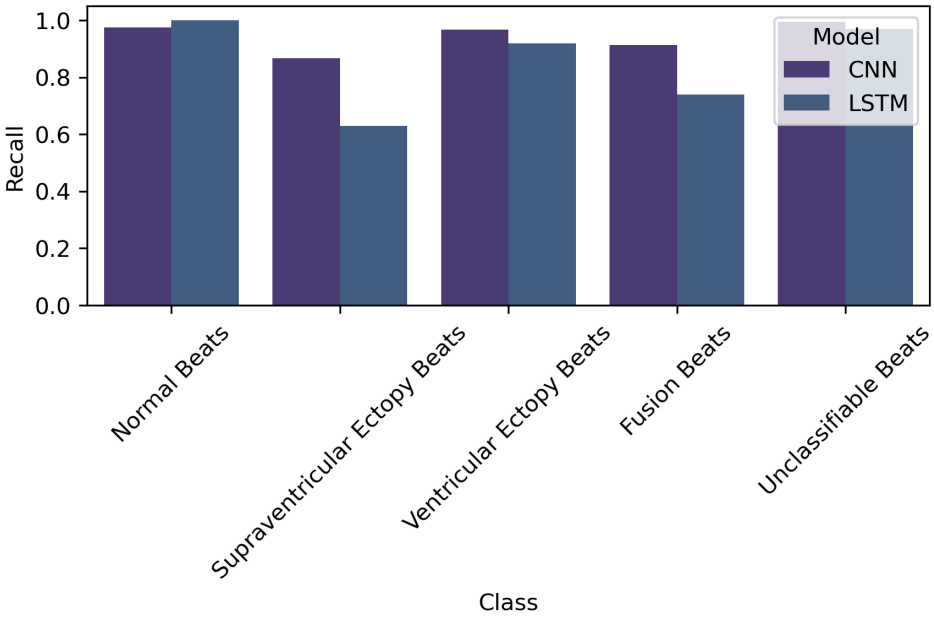

Figure 3.4: Recall score comparison.

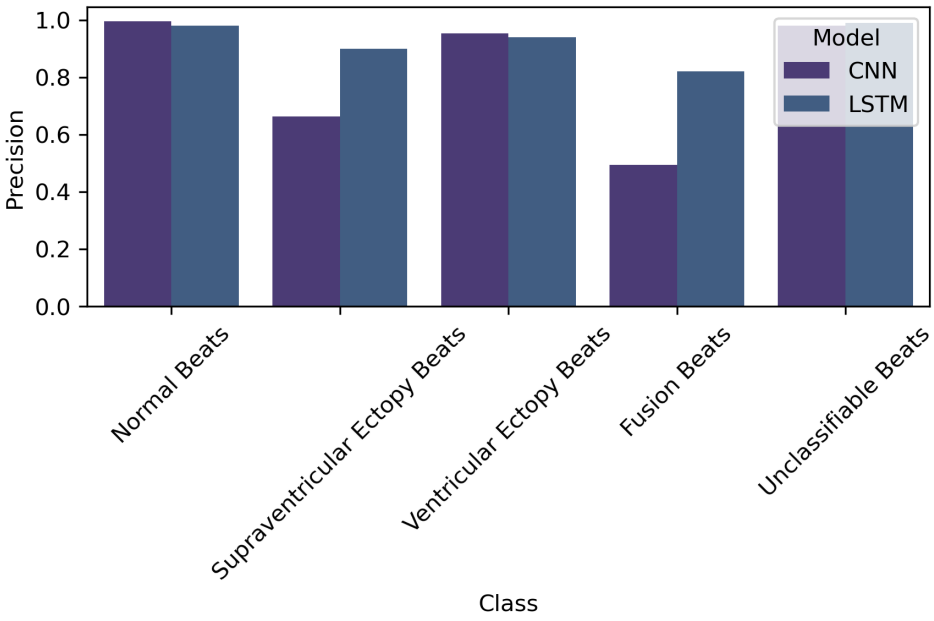

Figure 3.5: Precision score comparison.

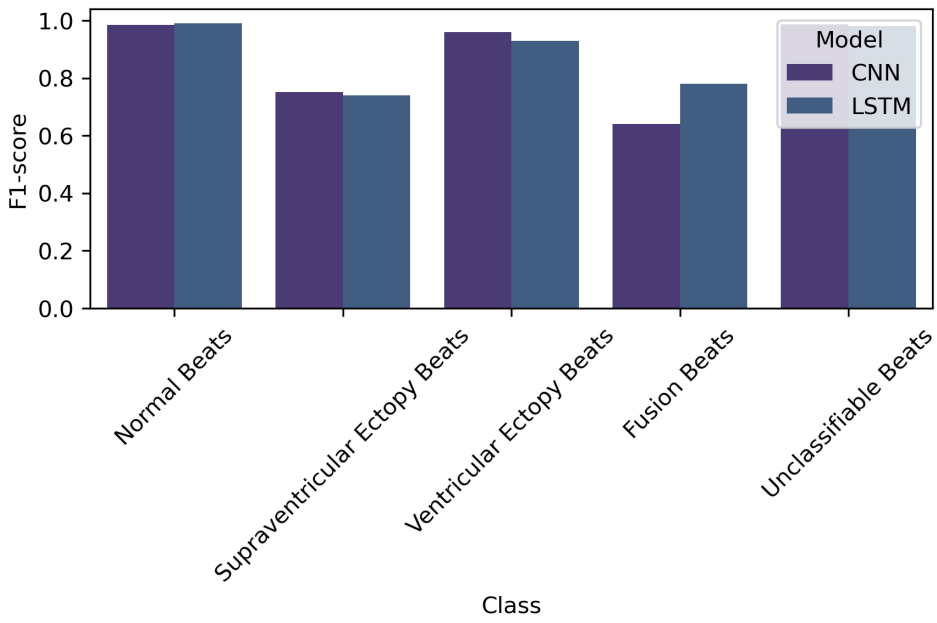

Figure 3.6: F1 score comparison.

Chapter 4

Discussion

The results of this study suggest that a deep learning approach can classify a range of distinct arrhythmias from single-lead ECGs with high diagnostic performance. The CNN achieved an average area under the ROC curve of 0.97 and an average F1-score of 0.83 when validated against an independent test dataset annotated by a consensus committee of cardiologists. These findings suggest that deep learning models can be used to classify a range of distinct arrhythmias from single-lead ECGs with high diagnostic performance.

Given that more than 300 million ECGs are recorded annually, high-accuracy diagnosis from ECG can save expert clinicians and cardiologists considerable time and decrease the number of misdiagnoses. Furthermore, we hope that this technology coupled with low-cost ECG devices enables more widespread use of the ECG as a diagnostic tool in places where access to a cardiologist is difficult.

4.1 Potential Limitation

There are a few potential limitations in the works of this project:

1. The number of classes is limited to 5, which may not be sufficient to cover all possible arrhythmias.

2. The model was trained on a single-lead ECG, which may limit the generalizability of the results to multi-lead ECGs.

3. The model was trained on a single dataset, which may limit the generalizability of the results to other datasets.

4. The model was trained on a single type of ECG device, which may limit the generalizability of the results to other types of ECG devices.

4.2 Future Work

The work presented in this project pose several opportunities for future research, the most crucial of which will be discussed below.

A significant area for future work is to use other datasets with more types of arrhythmias, more patients, and detailed demographics and gender information. This could improve the model's performances by allowing us to split the dataset and address potential biases more effectively.

Another is the use of an independent test set, to evaluate the generalization ability of the models.

1. Data Augmentation for the minority classes
2. In future work also main distinguishing criteria will be used, provided by doctors, to validate the results obtained by the model.
3. Try different architectures like attention and also Transfer Learning, for example using a ResNet
4. Anomaly detection... (suggested by prof. Ferrara)
5. A lot of different combinations can be tried, for example, I could compare the results achieved splitting in different manners the sets.
    (a) Adam with weight decay

4.3 Closing Words

The project presented an exploration of deep learning models for the classification of arrhythmias from single-lead ECGs. The project provided improvements in the performance of the models. Automatic ECG arrhythmias detection have the potential to prevent stroke or sudden death caused by cardiac arrhythmia. The prevention and treatment of cardiac arrhythmias will continue to be improved in the future, according to the author, by the use of more deep-learning-based ECG detection methods, allowing for effective and timely treatments.

Appendix A

Additional Information

A.1 Model training

For both the CNN and RNN the dimension of the input of the model is (187, 1), (timesteps, features).

A.1.1 LR schedule

At the beginning I used a LR schedule. Specifically, an exponential decay schedule with a warmup period and a decay rate of 0.1 was used, defined by the following equation:

$$LR = \begin{cases} \text{initial\_lr} \cdot (\text{epoch} + 1) & \text{if epoch} < \text{warmup\_epochs} \\ \text{lr} \cdot \text{decay\_rate}^{(\text{epoch}/\text{warmup\_period})} & \text{otherwise} \end{cases} \tag{A.1}$$

A.1.2 ReduceLROnPlateau

Using a patience of 3 and a factor of 0.5 which means that the learning rate will be reduced by half when the validation loss does not improve for 3 epochs.

Where warmup epochs=5 and decay rate=0.1.

A.1.3 Convolutional Neural Network

Same padding was used in the covolutional layers to preserve the dimensions of the input. Considering that each signal has 187 samples, I carefully considered only 3 Max Pooling layers to avoid reducing the size of the input too much. The number of filters in the convolutional layers was increased to capture more features in the input signal.

The decision of increasing the number of convolutional layers, and so of increasing the

deepnes ot the network was taken after seeing the results of the hyperparameter tuning. From the last two parallel coordinate plots.

At the beginning I used Adam with a LR scheduler, but at the end I've decided to just use Adam. Then seeing the results I decided to use ReduceLROnPlateau, since the loss over epochs was a little bit unstable.

A.1.4 Recurrent Neural Network

I've tried to use Adam and ReduceLROnPlateau but the results were not that good, so I decided to use only the Adam optimizer.

A.2 Results Hyperparameter Tuning

A.2.1 Convolutional Neural Network

Before starting disccusing about the results obtained I should mention that they can be grouped, based on the splitting of the datasets. At the beginning I extracted the validation set from the training set, then I noticed that since the number of samples in the test set I could extract the samples from it.

In figure A.1 is shown the parallel coordinate plot of the hyperparameters of the CNN model.

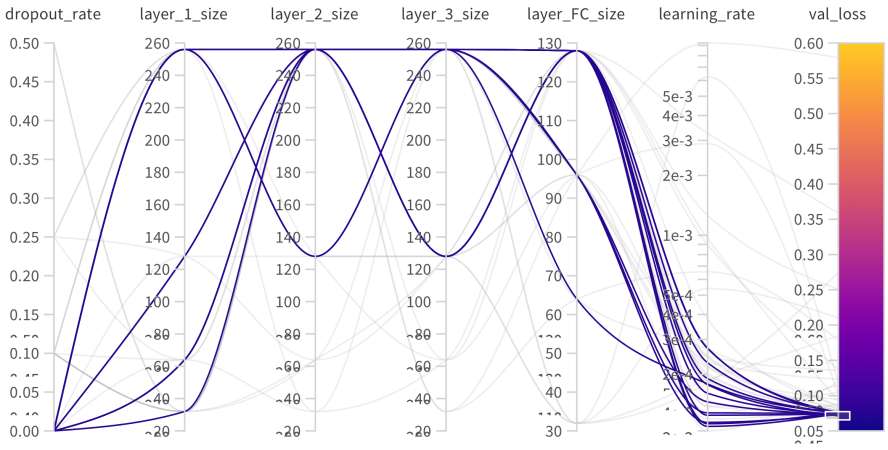

Figure A.1: Parallel Coordinate Plot of Hyperparameters of the trained CNN model.

From the Figure A.1 is possible to notice that the best model was obtained with the following hyperparameters:

- Number of filters in the first convolutional layer: 64

- Number of filters in the second convolutional layer: 128
- Number of filters in the third convolutional layer: 256
- Number of neurons in the Fully-Connected layer: 512
- Learning rate: 0.001
- Dropout rate: 0.5

However, you should notice that the values of most of the hyperparmeters reached the extremes, this suggests that I should enlange the search space of the hyperparameters.

A.2.1.1 Recurrent Neural Network

The first thing reshaping the signals data, I added a single dimension For our dataset, we will transform each sample from a 2D array of shape (samples, timesteps) to a 3D array of shape (samples, timesteps, features).

A.3 Previous implementations

A.3.1 LR schedule

A warm-up learning rate schedule is often used to stabilize training in the initial epochs. The learning rate gradually increases during the warm-up period, and then after reaching the initial value it decreases exponentially, this can help improve model convergence.

The choice of this learning rate schedule is based on quick experiments that showed its effectiveness in stabilizing the training process. Without the warm-up phase, the loss exhibited unstable behavior and did not converge to the optimal solution. By implementing the warm-up schedule, the model's training became more stable and converged more reliably. (Maybe I saw this behavior because I was using incorectly the LR scheduler, at the beginning I was using two of them and maybe they were conflicting with each other, and also with Adam)

Then implemented properly the warmup period and the exponential decay but I didn't see that much of improvement so I decided to spend my time improving other hyperparameters.

At the end I decided to remove it since I am using Adam optimizer, which is an adaptive optimizer.

A.4 Future implementation

A.4.1 Hyperparameter tunining

1. Number of LSTM units
2. Attention units (multihead attention)
3. L1/L2 regularization
4. Type of RNN layer: LSTM/ GRU (I've been inspired from the Sweep example in the website)

Appendix B

Additional Theory

B.1 Model training

Usually the model performs the update every batch. So the model is trained on a batch of data, then the model parameters are updated based on the loss calculated on that batch. This process is repeated for all the batches in the dataset. This is called an epoch.

Number of batches = Total size of the set / batch size

B.2 Hyperparameter tuning using Sweep WandB

Training a machine learning model normally requires multiple iterations. Furthermore, it might be unclear which Hyperparameter combination to use for a given training run.

Weights and Biases Sweeps [5] can be used to create an organized and efficient way to automatically search through combinations of hyperparameters values.

In order ot find hyperparameters there are three search methods: grid, random and Bayesian search. After specifying it, a metric to optimize must be defined, as instance the validation loss.

B.2.1 Bayesian Optimization

What Is Bayesian Hyperparameter Optimization? With Tutorial. WandB

Unlike grid and random search methods, which try every possible combination blindly, Bayesian optimization uses a smart and efficient approach to guide its search, informed by previous evaluations.

B.2.2 Visualize Sweep Results

B.2.2.1 Parallel Coordinate Plot

This plot maps hyperparameters values to model metrics. It is useful to focus the attention on combinations of Hyperparameters that led to the best model performance. See figure B.1 for an example of a parallel coordinate plot.

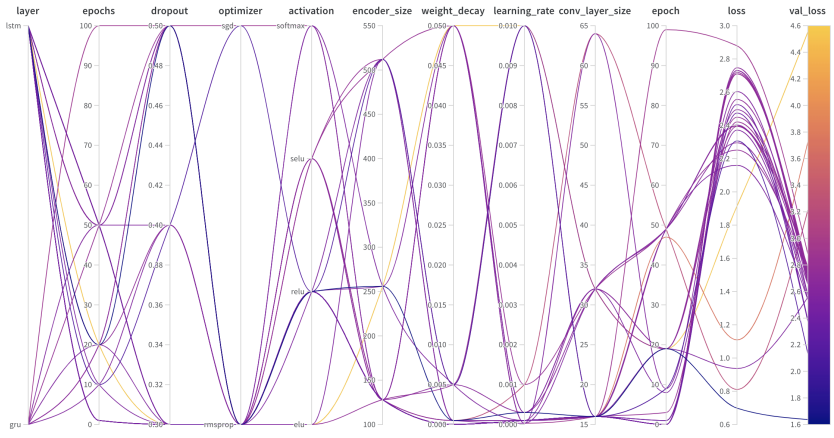

Figure B.1: Example of Parallel Coordinate Plot of Hyperparameters.

B.3 Classification Metrics

Classification metrics are crucial for evaluating the performance of a machine learning model. This section covers some fundamental metrics used in classification tasks.

B.3.1 Confusion Matrix

A confusion matrix is a table that is used to describe the performance of a classification model on a set of data for which the true values are known.

A typical confusion matrix has four main components

1. True Positive (TP)
2. True Negative (TN)
3. False Positive (FP)
4. False Negative (FN)

B.3.1.1 Confusion matrix analysis

The diagonal elements of the confusion matrix represent the counts of correctly predicted samples for each class.

The off-diagonal elements represent the misclassifications made by the model.

Confusion matrix can be used to calculate various performance metrics such as accuracy, precision, recall, and F1-score which can help in evaluating the model's effectiveness and identifying areas of improvement.

B.3.2 Accuracy

Very simple value, and not suitable in case of class imbalance. I computed as the total number of correctly classified istances over the total number of istances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{B.1}$$

B.3.3 Precision

Precision is the ratio of true positive predictions to the total number of positive predictions made by the model. It is a measure of the model's accuracy when predicting positive classes. Precision is calculated as follows:

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{B.2}$$

Precision is important in scenarios where the cost of false positives is high. Because when FP is high we Precision is low.

B.3.4 Recall

Recall is the ratio of true positive predictions to the total number of ACTUAL positive instances in the dataset. It is a measure of the model's ability to identify all positive instances. Recall is calculated as follows:

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{B.3}$$

Recall is crucial in situations where the cost of false negatives is high. Because when FN is high Recall is low.

B.3.5 F1-score

Harmonic average of precision and recall.

B.3.6 ROC curve

The ROC curve is a visual representation of model performance across all thresholds. The long version of the name, receiver operating characteristic, is a holdover from WWII radar detection.

The ROC curve is drawn by calculating the true positive rate (TPR) and false positive rate (FPR) at every possible threshold (in practice, at selected intervals), then graphing TPR over FPR.

B.3.6.1 Area Under the Curve (AUC)

The area under the ROC curve (AUC) represents the probability that the model, if given a randomly chosen positive and negative example, will rank the positive higher than the negative. It is a single scalar value that summarizes the performance of the classifier over all thresholds. AUC is a useful measure for comparing the performance of two different models, as long as the dataset is roughly balanced. (See Precision-recall curve, above, for imbalanced datasets.) The model with greater area under the curve is generally the better one.

B.3.7 Precision-Recall curve

1. Trade-off between precision and recall

AUC and ROC work well for comparing models when the dataset is roughly balanced between classes. When the dataset is imbalanced, precisio-recall curves (PRCs) and area under those curves may offer a better comparative visualization of model performance.

Precision-recall curves are created by plotting precision on the y-axis and recall on the y-axis across all thresholds.

Motivated by the following statement: "ROC has become an entrenched evaluation tool for assessing the performance of classifiers and risk scores in the medical arena. However, when the data is highly imbalanced, ROC can provide a misleading optimistic view of the performance of the classifiers. In such circumstances, it is imperative to employ PRC to precisely evaluate the prediction of the minority class." I've decided to obtain the Precision-Recall curve, since the dataset is highly umbalanced, and in particular in this

project I will extract the Area Under the curve for the different classes. (Maybe I will not compute, only for lack of time)

B.4 More on Cardiac Arrhthmias

Cardiac arrhythmias are a typical and serious group of cardiovascular diseases (CVD), leading to various complications and high mortality rates. Early detection and proactive treatment of cardiac arrhythmia are extremely essential for averting worse consequences.

Bibliography

[1] Awni Y. Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H. Tison, Codie Bourn, Mintu P. Turakhia, and Andrew Y. Ng.

[2] Awni Y. Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H. Tison, Codie Bourn, Mintu P. Turakhia, and Andrew Y. Ng. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine*, 25(1):65–69, January 2019.

[3] Dr Sagar Mahida. Ectopic beats. Accessed: 2024-12-08.

[4] PhysioNet. Mit-bih arrhythmia database, 2005. Accessed: 2024-12-05.

[5] Weights and Biases. Wandb sweeps documentation.