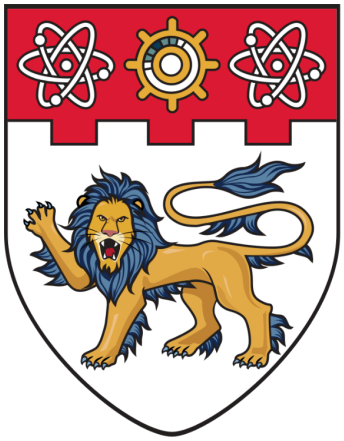

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4001: Neural Networks & Deep Learning

Group Assignment: A. Deep Learning for ECG Heartbeat Classification

Tan Jia Ze (U2122410B)

Tng Meng Kiat (U2122251B)

Xavier Yeo (U2120239E)

College of Computing and Data Science

Introduction

An arrhythmia, or irregular heartbeat, occurs when there is a problem with the heart's rate or rhythm. This can cause the heart to beat too fast, too slow, or with an uneven pattern. Arrhythmias stem from several factors, including structural heart disease, genetic predispositions, electrolyte imbalances, or damage from events like myocardial infarctions [1]. If arrhythmia is not treated, the heart, brain and other organs may be damaged, potentially leading to life-threatening strokes.

Arrhythmias can be classified based on the area of the heart affected and their impact. The main types include Supraventricular and Ventricular arrhythmias. Supraventricular arrhythmias start in the atria or the gateway to the lower chambers whereas Ventricular arrhythmias start in the heart's lower chambers. Ventricular arrhythmias are more dangerous and usually require urgent medical care [2].

Problem Description

An electrocardiogram (ECG) records electrical signals in the heart, which can reveal the health of a person's heart. With a dataset of ECG signals, each sample representing a single heartbeat categorised into different classes based on their underlying rhythm (normal or various types of arrhythmias).
Develop a deep learning model to accurately classify ECG heartbeat signals into different classes.

Literature Review

With recent advancements in computational power, the availability of large labelled datasets and developments in algorithms, deep learning has become more efficient. Furthermore, with architectures such as CNNs for image recognition, RNNs and transformers for sequential data and hybrid models, the accuracy and adaptability of deep learning have improved significantly.

Various research works on classifying arrhythmia from ECG signals have been done using different model architectures and techniques to process the ECG signals. Some examples include CNNs [3], RNNs (and its variants) [4] and hybrid models [5].

Exploratory Data Analysis

We will be working with the MIT-BIH arrhythmia data [6][7]. The data set is a collection of ECG recordings by 47 subjects. The train data consists of 87553 samples while the test data has 21891 samples. Each sample consists of 188 columns, Columns 1 to 187 are the QRS complex for each ECG signal, while the last column is signal classification. The signal is either cut short or padded with zeros to ensure all heartbeat records contain exactly 187 values. The sampling frequency is 125Hz. We first analysed the dataset. From Figure 1, we can observe that the distribution of the classes is heavily skewed for both the train and test data. Figure 2 shows an example of a heartbeat ECG signal from each class.

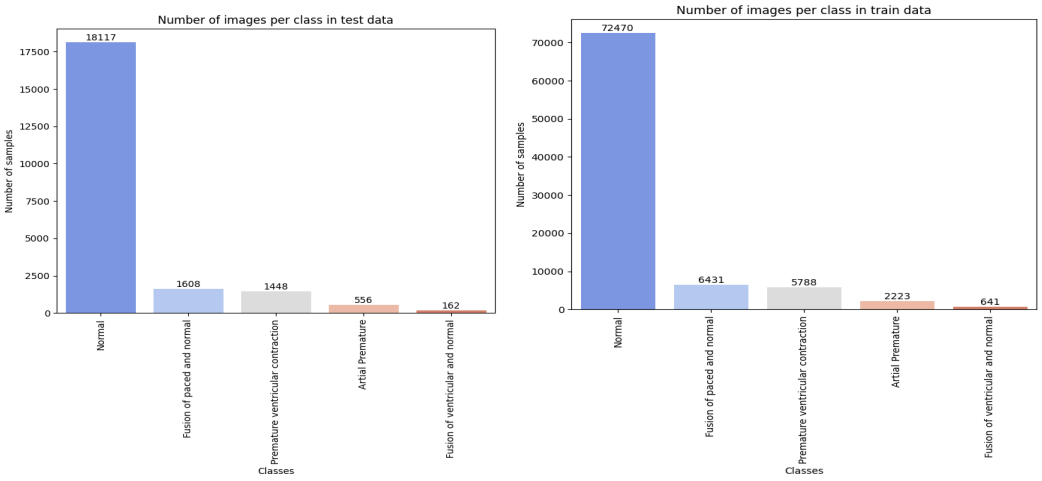

Figure 1: Number classes in Train and Test Data

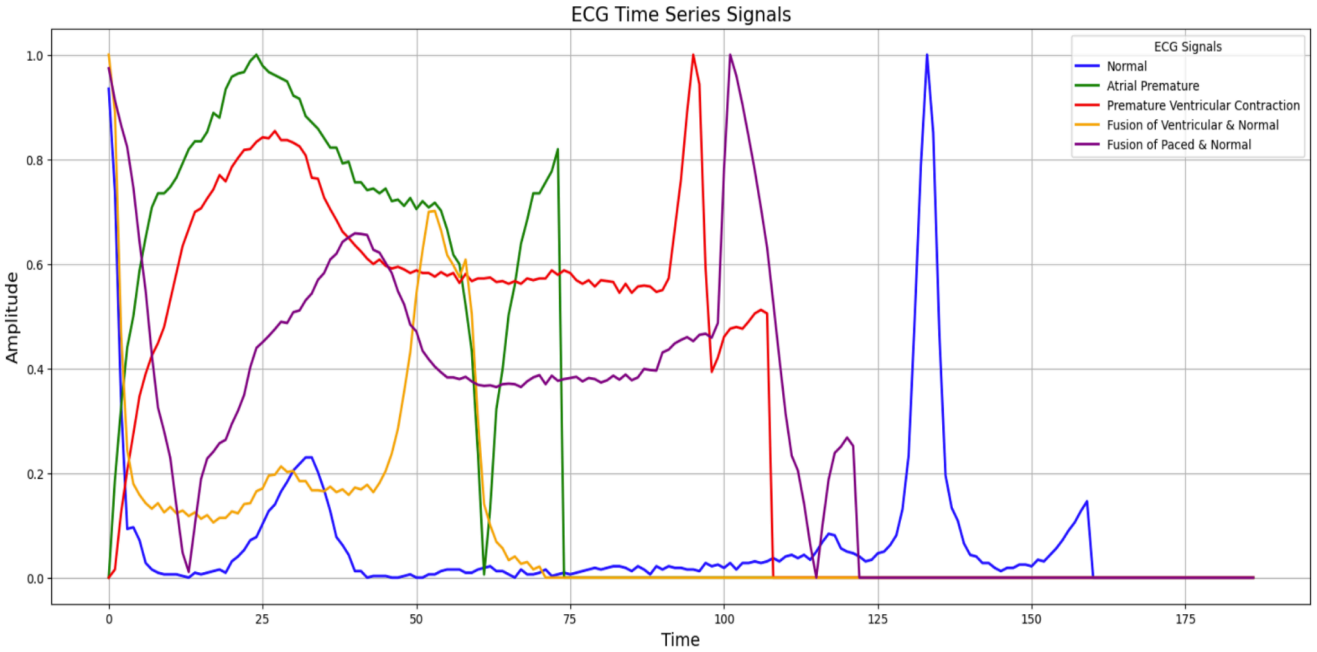

Figure 2: Heartbeat ECG Signal Example from the 5 different classes

Data Preprocessing

The training dataset is further split into training and validation sets with a ratio of 80:20. For each sample in the datasets, the first 187 columns form our input tensor and the last column forms the target tensor.

Model Configuration

No hyperparameter tuning was performed in order to ensure

- Fair Comparison: Ensuring differences in performances stem from model architectures instead of hyperparameter selection
- Resource Constraint: Limited resources prevent us from extensive hyperparameter tuning with GridSearch

Hyperparameters that are kept constant for all models

- Batch Size: 32
- Learning rate: 1 x e^{-4}
- Optimiser: Adam optimiser
- EarlyStopper: Patience value of 3, Delta of 0.00

Model Selection

RNN

Rationale: We first trained the data using an LSTM neural network. Due to the sequential nature of the training data, the LSTM model can process the sequential information of the ECG data. Furthermore, LSTM is used over the traditional RNN model to mitigate the vanishing and exploding gradient problem during gradient backpropagation.

Setup:

- The Train and Test input data are first scaled
- Train data is split into training and validation data with a ratio of 80:20
- Training, validation and test inputs are transformed into a Numpy array (samples, timesteps, features)

Model Architecture:

```
RNN(
  (lstm): LSTM(1, 128, batch_first=True)
  (fc): Linear(in_features=128, out_features=5, bias=True)
```

Figure 3: Model architecture of RNN Model

Results:

```
Test accuracy: 82.67%, Test loss: 0.636445
```

Figure 4: Test Accuracy of RNN Model

	precision	recall	f1-score	support
N	0.83	1.00	0.91	18118
S	0.31	0.09	0.14	556
V	0.00	0.00	0.00	1448
F	0.00	0.00	0.00	162
Q	0.00	0.00	0.00	1608
accuracy			0.83	21892
macro avg	0.23	0.22	0.21	21892
weighted avg	0.70	0.83	0.75	21892

Figure 5: Classification Report of RNN Model

Insights:
We can observe that, although the test accuracy of the model is high at 82.67%, and a low test loss, the precision, recall and F1 score for the model is low. We conclude that the imbalanced nature of the train and test data has led to the overall poor performance of the model.

2-Dimensional CNN

Rationale: ECG signals are inherently 1-dimensional time series data. Data transformation via the Short-Time Fourier Transform (STFT) yields 2-dimensional time-frequency data, which can reveal temporal frequency patterns while preserving the temporal aspect of the data

Setup: Conduct STFT onto current ECG signal dataset & Create model with 2-dimensional Convolution and Pooling layers

Model Architecture:

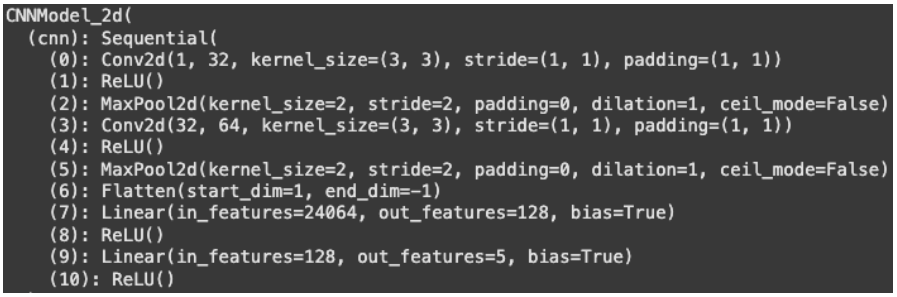

Figure 6: Model architecture of 2-Dimensional CNN Model

Results:

Figure 7: Test accuracy of 2-Dimensional CNN Model

	precision	recall	f1-score	support
N	0.96	0.98	0.97	18117
S	0.00	0.00	0.00	556
V	0.78	0.94	0.85	1448
F	0.00	0.00	0.00	162
Q	0.96	0.97	0.96	1608
accuracy			0.94	21891
macro avg	0.54	0.58	0.56	21891
weighted avg	0.92	0.94	0.93	21891

Figure 8: Classification report of 2-Dimensional CNN Model

Insights:

Strong overall model performance: (Refer to figure 7 & 8)

- Strong overall classification performance with High test accuracy & High weighted averages for precision, recall and f1-score.

Classification performance:

- Strong classification for class N & Q
- Good classification for class V, Slightly lower precision score ≈ higher false positive rate
- Bad classification for class S & F. Precision & recall & f1-score are 0.00, indicating that the model is unable to identify these classes.

Hybrid CNN-LSTM

Rationale: We decided to try an CNN-LSTM model as it effectively combines the strengths of RNNs (specifically LSTMs in this case) and CNNs. LSTMs are ideal for capturing temporal dependencies over the ECG data, while CNNs are able to extract important local features, such as certain patterns in short segments of the data, which may be important to characterising the patterns. This combination allows the model to leverage on both the temporal structure and local patterns of the data, allowing accurate classification of the ECG signals.

Model Architecture:

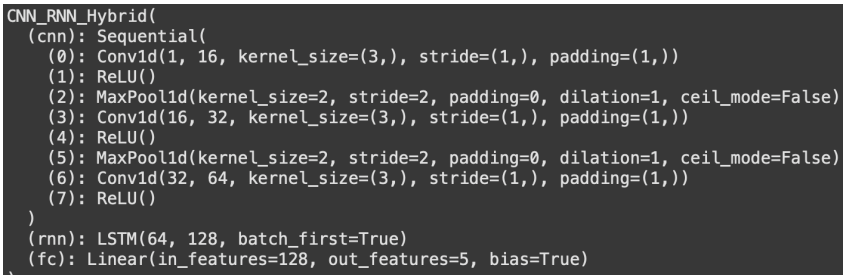

Figure 9: Model architecture of Hybrid CNN-LSTM

Results:

```
Test accuracy: 97.71%, Test loss: 0.088998
```

Figure 10: Test accuracy of Hybrid CNN-LSTM Model

	precision	recall	f1-score	support
N	0.98	0.99	0.99	18117
S	0.89	0.65	0.75	556
V	0.94	0.93	0.93	1448
F	0.72	0.75	0.73	162
Q	0.99	0.97	0.98	1608
accuracy			0.98	21891
macro avg	0.91	0.86	0.88	21891
weighted avg	0.98	0.98	0.98	21891

Figure 11: Classification report of Hybrid CNN-LSTM

Insights:
We can observe that the model has a good accuracy on the test set and has good recall and f1-score on all the different classes as well.

<u>Alternative Approach - CNN-BiLSTM</u>
Hypothesis: ECG data patterns may depend on both prior and future signals, meaning that all the data points are crucial and can be used together for more accurate classification.

Experiment: Replace LSTM with Bidirectional LSTM (BiLSTM)

LSTMs can be bidirectional, allowing the model to get context from both past and future timesteps. By considering the sequence from both directions, the feature representation generated by the BiLSTM should be more detailed, which should improve the classification accuracy.

Results:

	precision	recall	f1-score	support
N	0.98	1.00	0.99	18117
S	0.89	0.69	0.78	556
V	0.97	0.91	0.94	1448
F	0.78	0.74	0.76	162
Q	0.99	0.97	0.98	1608
accuracy			0.98	21891
macro avg	0.92	0.86	0.89	21891
weighted avg	0.98	0.98	0.98	21891

Figure 12: Classification report of Hybrid CNN-BiLSTM

Insights:
We can observe that the recall and f1-score using the CNN-BiLSTM model is slightly better compared to the CNN-LSTM model.

<u>Transformer</u>

Rationale: Similar to RNN models, Transformers are designed to handle sequential/temporal data with self-attention mechanisms. But unlike RNN models, transformers are more effective in capturing long-range dependencies with positional encoding and better gradient flow over long sequences.
Setup: Create an encoder-only transformer model, with a positional encoding linear layer to vectorise the ECG signals

Model Architecture:

```
TransformerClassifier(
  (input_embedding): Linear(in_features=1, out_features=256, bias=True)
  (positional_encoding): LearnedPositionalEncoding(
    (dropout): Dropout(p=0.05, inplace=False)
    (pos_embedding): Embedding(187, 256)
  )
  (transformer_encoder): TransformerEncoder(
    (layers): ModuleList(
      (0-3): 4 x TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=256, out_features=256,
bias=True)
        )
        (linear1): Linear(in_features=256, out_features=2048, bias=True)
        (dropout): Dropout(p=0.05, inplace=False)
        (linear2): Linear(in_features=2048, out_features=256, bias=True)
        (norm1): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0.05, inplace=False)
        (dropout2): Dropout(p=0.05, inplace=False)
      )
    )
  )
  (classifier): Linear(in_features=256, out_features=5, bias=True)
  (dropout): Dropout(p=0.05, inplace=False)
  (relu): ReLU()
```

Figure 13: Model architecture of Transformer Model

Results:

```
Test accuracy: 92.14%, Test loss: 0.234010
```

Figure 14: Test accuracy of Transformer Model

	precision	recall	f1-score	support
N	0.99	0.92	0.95	18117
S	0.35	0.81	0.49	556
V	0.90	0.89	0.90	1448
F	0.24	0.90	0.38	162
Q	0.93	0.98	0.96	1608
accuracy			0.92	21891
macro avg	0.68	0.90	0.74	21891
weighted avg	0.96	0.92	0.93	21891

Figure 15: Classification report of Transformer Model

Insights:

Strong overall model performance: (Refer to figure 14 & 15)

- Strong overall classification performance with High test accuracy (~92%) & High weighted averages for precision, recall and f1-score.

Classification performance:

- Strong classification performance for class N, Q, V
- Transformer model has trouble classifying class S, F where precision scores are lower than the others, indicating that the model has classified many False Positives for class S & F

Autoencoder

Rationale: We also decided to test out an Autoencoder followed by a Linear Layer for classification of the ECG data. Autoencoders are able to learn compact, meaningful representations of the input data by encoding the most relevant features of different patterns which can be used for classification.

Model Architecture:

```
Autoencoder(
  (encoder): Sequential(
    (0): Linear(in_features=187, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=128, bias=True)
    (3): ReLU()
  )
  (decoder): Sequential(
    (0): Linear(in_features=128, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=187, bias=True)
    (3): Sigmoid()
```

Figure 16: Model Architecture of Autoencoder model

Results:

```
Test accuracy: 93.25%, Test loss: 0.252659
```

Figure 17: Test accuracy of Autoencoder Model

	precision	recall	f1-score	support
N	0.94	0.99	0.96	18117
S	0.91	0.34	0.50	556
V	0.77	0.62	0.69	1448
F	0.57	0.30	0.39	162
Q	0.97	0.87	0.92	1608
accuracy			0.93	21891
macro avg	0.83	0.62	0.69	21891
weighted avg	0.93	0.93	0.93	21891

Figure 18: Classification report of Autoencoder Model

Insights:
We can observe that the recall and f1-score of the S, V and F classes are generally quite low.

<u>Alternative Approach - MLP for Classifier</u>
Hypothesis: Instead of using a single linear layer for classification of the outputs of the encoder, an MLP would perform better

Experiment: Replace linear layer with an MLP for classification

A single linear layer for classification may have better accuracy if the encoder output of the autoencoder already contains well-separated features that can be linearly classified. On the other hand, if the output is complex and has more complicated decision boundaries, an MLP will likely perform better as it can capture non-linear relationships.

Model Architecture:

```
ClassifierMLP(
  (classifier): Sequential(
    (0): Linear(in_features=128, out_features=128, bias=True)
    (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Dropout(p=0.3, inplace=False)
    (4): Linear(in_features=128, out_features=64, bias=True)
    (5): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): Dropout(p=0.3, inplace=False)
    (8): Linear(in_features=64, out_features=5, bias=True)
  )
)
```

Figure 19: Model Architecture of the Classifier using MLP

Results:

```
Test accuracy: 97.34%, Test loss: 0.092593
```

Figure 20: Test accuracy with of the Classifier using MLP

	precision	recall	f1-score	support
N	0.98	0.99	0.99	18117
S	0.89	0.62	0.73	556
V	0.93	0.91	0.92	1448
F	0.88	0.49	0.63	162
Q	0.99	0.97	0.98	1608
accuracy			0.97	21891
macro avg	0.93	0.80	0.85	21891
weighted avg	0.97	0.97	0.97	21891

Figure 21: Classification report with of the Classifier using MLP

Insights:
Using an MLP instead of a single linear layer for classification improved the classification results.

<u>Alternative Approach - Separate Autoencoders & Novel Classification Method</u>
Hypothesis: Training separate autoencoders for each class would reduce the effect of the data imbalance.

Experiment: Create Autoencoders for each class, trained on the data of the respective classes and use the reconstruction error for classification [8].

We tried to use the data of the classes to train corresponding autoencoder models. Then during testing, the test sample will be reconstructed by each of the class autoencoder models and the class autoencoder with the minimum reconstruction loss will be used as the predicted label.

Model Architecture:

```
Autoencoder(
  (encoder): Sequential(
    (0): Linear(in_features=187, out_features=256, bias=True)
    (1): ReLU()
    (2): Linear(in_features=256, out_features=128, bias=True)
    (3): ReLU()
    (4): Linear(in_features=128, out_features=64, bias=True)
    (5): ReLU()
  )
  (decoder): Sequential(
    (0): Linear(in_features=64, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=256, bias=True)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=187, bias=True)
  )
)
```

Figure 22: Model Architecture of the Autoencoder models

Results:

```
Test Accuracy: 93.44%
```

Figure 23: Test accuracy

	precision	recall	f1-score	support
N	0.93	1.00	0.96	18117
S	0.96	0.46	0.62	556
V	0.98	0.48	0.64	1448
F	0.77	0.06	0.11	162
Q	0.99	0.88	0.93	1608
accuracy			0.93	21891
macro avg	0.93	0.57	0.65	21891
weighted avg	0.94	0.93	0.92	21891

Figure 24: Classification report

Insights:
The recall and f1-scores were quite poor for the S, V and especially the F class. Unfortunately, even though this approach was aimed to reduce the effects of the imbalanced dataset, it did not work that well.

Data Augmentation Techniques

SMOTE

Due to the unbalanced nature of the dataset, we decided to use a data technique called Synthetic Minority Over-Sampling Technique (SMOTE). The goal of SMOTE is to balance the dataset by creating new “synthetic” data in the feature space of the minority class [9]. SMOTE algorithm is a widely used oversampling strategy, and we believe that it can be applied to our ECG dataset.

0.0	72470
1.0	72470
2.0	72470
3.0	72470
4.0	72470

Figure 25: Count of Dataset after applying SMOTE

We can observe that after applying the SMOTE algorithm, all the classes in the dataset are well represented with the same number of samples in each class. The models are then retrained with the new Training dataset.

Results

	precision	recall	f1-score	support
N	1.00	0.97	0.98	18117
S	0.61	0.88	0.72	556
V	0.93	0.96	0.94	1448
F	0.59	0.89	0.71	162
Q	0.96	0.99	0.97	1608
accuracy			0.97	21891
macro avg	0.82	0.94	0.87	21891
weighted avg	0.97	0.97	0.97	21891

Figure 26: Results of RNN model with new Train Data

Performance metrics

Precision score

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Figure 27: Precision score formula

Precision score accounts for the False Positive rate within the model predictions, and provides a more objective view of model accuracy. A high precision score should be prioritised if False Positive instances are costly or dangerous.

Recall score

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

Figure 28: Recall score formula

High recall score indicates the model has a low false negative rate. Given our use case of classifying arrhythmia, a False Negative instance is more critical than False Positive instance, as failure to identify a patient with arrhythmia can lead to delayed medical care

F1 score

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figure 29: F1 score formula

F1 score combines Precision score and Recall score into 1 single metric allowing easier evaluation. Balancing both score also allows F1 score to account for both False positives and False negatives

Comparative analysis

Model Architecture	Macro Precision	Macro Recall	Macro F1-Score	Weighted Precision	Weighted Recall	Weighted F1-Score
RNN	0.62	0.56	0.56	0.92	0.93	0.92
CNN 2-D	0.54	0.58	0.56	0.92	0.94	0.93
Autoencoder (Novel)	0.93	0.57	0.65	0.94	0.93	0.92
Autoencoder	0.83	0.62	0.69	0.93	0.93	0.93
Transformer	0.68	**0.90**	0.74	0.96	0.92	0.93
Autoencoder (MLP)	0.93	0.80	0.85	0.97	0.97	0.97

CNN LSTM	0.91	0.86	0.88	**0.98**	**0.98**	**0.98**
CNN BiLSTM	**0.92**	0.86	**0.89**	**0.98**	**0.98**	**0.98**

Figure 30: Results visualisation

Based on our results, Hybrid CNN-LSTM models performed the best out of all our approaches.

- CNN-LSTM & CNN-BiLSTM maintain high macro and weighted averages, indicating balanced performance across all classes
- Traditional RNN and CNN 2-Dimensional models have lower macro averages, indicating inconsistent performance across the classes and problem classifying those with fewer samples (‘S’ & ‘F’)
- Transformer has a highest macro recall scores but lower precision score, which indicates the Transformer model is effective in classifying True Positives & False Negatives but struggles in classifying False Positives

The comparative analysis between the models shows that CNN-LSTM variants are recommended for their overall and class-wise performance. If the priority shifts to minimise False Negatives, Transformer can be recommended for their high recall scores. Traditional RNN and CNN 2-Dimensional models should be avoided unless strategies to address class imbalance is implemented

Model Architecture	Macro Precision	Macro Recall	Macro F1-Score	Weighted Precision	Weighted Recall	Weighted F1-Score
RNN	0.62	0.56	0.56	0.92	0.93	0.92
RNN (SMOTE)	0.83	0.93	0.87	0.98	0.98	0.98
Transformer	0.68	0.90	0.74	0.96	0.92	0.93
Transformer (SMOTE)	0.86	0.90	0.87	0.98	0.97	0.97
CNN-BiLSTM	0.92	0.86	0.89	0.98	0.98	0.98
CNN-BiLSTM (SMOTE)	0.85	0.92	0.88	0.98	0.98	0.98

Figure 31: SMOTE Results Visualisation

Based on our results, SMOTE did result in an overall increase in classification accuracy, especially the macro scores. This indicates that SMOTE’s application led to strong & consistent classification performance over all classes and does help address the issue of imbalance classes. The only model that SMOTE did not significantly improve the performance of was the CNN-BiLSTM model, possibly because the model was already performing very well. However, the overall recall of CNN-BiLSTM did increase with SMOTE which is more important in this scenario of detecting arrhythmia as mentioned previously.

Conclusion

In conclusion, we find that the CNN-BiLSTM model outperforms other architectures in the classification of arrhythmia in the MIT-BIH dataset. The combination of CNN and BiLSTM effectively captures both local features and temporal dependencies in the ECG data. This shows that both are crucial to accurately detect arrhythmia.

Additionally, applying SMOTE data augmentation proved to be beneficial, particularly in addressing the class imbalance that was present in the dataset. Using SMOTE improved the CNN-BiLSTM’s recall and F1 score, as the model was better able to identify the minority classes, which is critical in detecting arrhythmia events that are less frequent, but important for patients’ health.

Overall, these models can contribute to the diagnosis of arrhythmia, improving the accuracy and reliability of automated ECG analysis using deep learning. Also, making use of these models will allow early and accurate detection of arrhythmia that can help healthcare professionals diagnose and treat cardiac conditions more effectively, potentially saving lives and improving patient outcomes.

References

[1]
J. Kingma, C. Simard, and B. Drolet, “Overview of Cardiac Arrhythmias and Treatment Strategies,” *Pharmaceuticals*, vol. 16, no. 6, p. 844, Jun. 2023, doi: https://doi.org/10.3390/ph16060844.

[2]
National Heart, Lung, and Blood Institute, “Arrhythmias - Types | NHLBI, NIH,” *www.nhlbi.nih.gov*, Mar. 24, 2022. https://www.nhlbi.nih.gov/health/arrhythmias/types

[3]
A. A. Ahmed, W. Ali, T. A. A. Abdullah, and S. J. Malebary, “Classifying Cardiac Arrhythmia from ECG Signal Using 1D CNN Deep Learning Model,” *Mathematics*, vol. 11, no. 3, p. 562, Jan. 2023, doi: https://doi.org/10.3390/math11030562.

[4]
S. Singh, S. K. Pandey, U. Pawar, and R. R. Janghel, “Classification of ECG Arrhythmia using Recurrent Neural Networks,” *Procedia Computer Science*, vol. 132, pp. 1290–1297, 2018, doi: https://doi.org/10.1016/j.procs.2018.05.045.

[5]
B. Murugesan et al., "ECGNet: Deep Network for Arrhythmia Classification," 2018 IEEE International Symposium on Medical Measurements and Applications (MeMeA), Rome, Italy, 2018, pp. 1-6, doi: 10.1109/MeMeA.2018.8438739.

[6]
M. Kachuee, S. Fazeli, and M. Sarrafzadeh, “ECG Heartbeat Classification: A Deep Transferable Representation,” Jul. 2018. Available: https://arxiv.org/pdf/1805.00794

[7]
G. Moody and R. Mark, “MIT-BIH Arrhythmia Database v1.0.0,” *physionet.org*, Feb. 24, 2005. https://physionet.org/content/mitdb/1.0.0/

[8]
V.-I. Tomescu, G. Czibula, and Ş. Niţică, “A study on using deep autoencoders for imbalanced binary classification,” *Procedia Computer Science*, vol. 192, pp. 119–128, 2021, doi: https://doi.org/10.1016/j.procs.2021.08.013.

[9]
N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, no. 16, pp. 321–357, Jun. 2002, doi: https://doi.org/10.1613/jair.953.