Department of Electronic and Telecommunication Engineering
University of Moratuwa

BM4152 - Biosignal Processing

Project Final Report

210730B Wimalasiri W.M.

210169L Fernando W.W.R..N.S.

December 2, 2025

1 Introduction and Background

Cardiovascular diseases (CVDs) remain a major global health burden, causing nearly one-third of all deaths worldwide. Early detection of abnormal heart rhythms (arrhythmias) is therefore essential for effective diagnosis and timely intervention.

The electrocardiogram (ECG) is the primary non-invasive tool used to record the heart's electrical activity and to identify conduction or rhythm abnormalities. Traditional automatic arrhythmia classification methods typically rely on hand-crafted features (such as wavelet transforms, entropy measures, or autoregressive modelling) followed by classifiers like SVMs or k-nearest neighbours. However, these approaches depend heavily on manual feature engineering and often fail to generalise across different patients or recording conditions.

Deep learning offers an alternative by learning features directly from data. The selected work proposes converting ECG signals into time–frequency images using the short-time Fourier transform (STFT), and classifying these spectrograms using a 2D convolutional neural network (CNN) [1]. This approach eliminates manual feature extraction and enables more robust, data-driven arrhythmia classification.

2 Overview of Selected Paper and Dataset

The selected paper, "ECG Arrhythmia Classification Using STFT-Based Spectrogram and Convolutional Neural Network" by Huang *et al.*, proposes a deep learning framework for multi-class ECG arrhythmia classification [1]. One-dimensional ECG segments are transformed into two-dimensional STFT spectrograms, which are then fed to a custom 2D CNN for classification.

The authors consider five heartbeat types:

- Normal beat (NOR),
- Left bundle branch block beat (LBB),
- Right bundle branch block beat (RBB),
- Premature ventricular contraction beat (PVC),
- Atrial premature contraction beat (APC).

Using this representation and network, the paper reports an average accuracy of about 99%, outperforming a 1D-CNN baseline and several traditional feature-extraction-plus-classification methods [1].

2.1 MIT-BIH Arrhythmia Database

Both the original study and our implementation use the MIT-BIH Arrhythmia Database, a widely adopted benchmark for ECG analysis [2]. The database contains long-term ambulatory ECG recordings sampled at 360 Hz, with beat annotations provided by expert cardiologists.

Following the protocol in [1], we:

- select a single lead (V5) from each chosen record,

- segment the continuous ECG into non-overlapping windows of 10 s,
- select 2520 segments corresponding to the five arrhythmia categories, and
- split them into balanced training and testing sets (e.g., 450 training and 90 testing samples per class, with slightly fewer samples for PVC).

This ensures a balanced multi-class problem and allows direct comparison with the results reported in the paper.

3 Methods Description

The overall ECG arrhythmia classification pipeline consists of three main stages:

1. ECG data acquisition and segmentation,
2. time–frequency transformation using STFT,
3. classification using a 2D convolutional neural network.

We closely follow the methodology of Huang *et al.* [1], and then extend it with an adaptive multi-window STFT front-end.

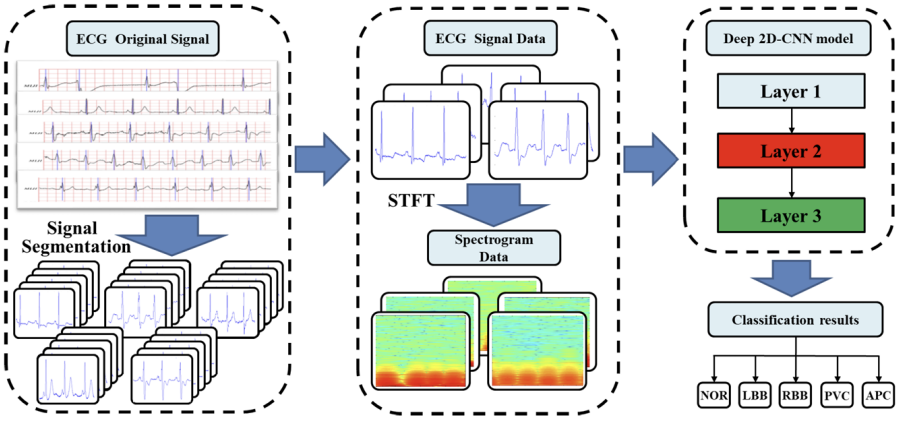

Figure 1: Overall procedures in ECG arrhythmia classification based on proposed 2D-CNN.

3.1 ECG Data Acquisition and Segmentation

ECG segments corresponding to the five heartbeat categories are obtained from the MIT-BIH Arrhythmia Database [2]. The V5 lead is extracted at 360 Hz, and each long recording is divided into 10-second segments. Segments are labelled according to the expert annotations, and a total of 2520 segments are retained and split into class-balanced training and test sets, as in [1]. This produces fixed-length inputs that preserve sufficient temporal information for rhythm analysis.

3.2 Time–Frequency Spectrogram Generation Using STFT

Because ECG signals are non-stationary, a time–frequency representation is more informative than a purely time- or frequency-domain view. The short-time Fourier transform (STFT) is used to obtain spectrograms for each segment.

For a discrete-time ECG signal $x[n]$, the STFT is

$$X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]\, w[n-m]\, e^{-j\omega n}, \tag{1}$$

where $w[\cdot]$ is a finite-length analysis window. The paper uses a Hanning window

$$w(n) = \begin{cases} 0.5\left(1 - \cos\left(\dfrac{2\pi n}{M-1}\right)\right), & 0 \le n \le M-1, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

with window length $M = 512$ samples and 50% overlap between windows [1].

For each 10-second segment, the magnitude of the STFT is mapped to a 256×256 image, normalised, and stored as a tensor of size $256 \times 256 \times 1$ for CNN input. The spectrograms show distinct patterns for different arrhythmia types [1], indicating that the STFT captures useful discriminative information.

In our extended implementation, we also experiment with an adaptive multi-window STFT front-end: spectrograms computed with window lengths of 128, 256 and 512 samples are stacked as three channels, giving a richer $256 \times 256 \times 3$ input that simultaneously encodes fast (QRS) and slower (P/T) components.

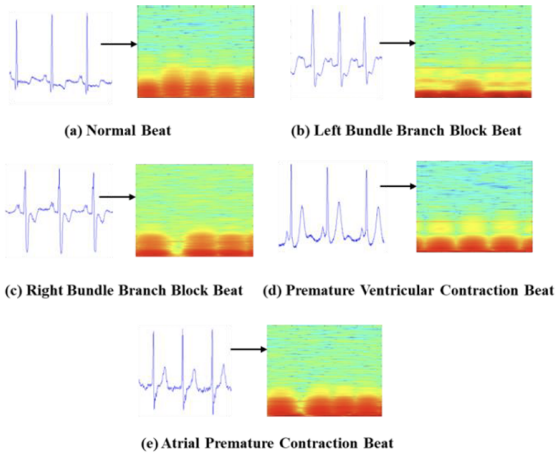

Figure 2: ECG spectrograms of a sample data instance belonging to each type of arrhythmia.

3.3 2D Convolutional Neural Network Architecture

The classifier is a 2D CNN operating on the STFT spectrograms. The architecture in [1] is summarised below:

Convolutional Feature Extractor

- **Layer 1:** Conv2D with 8 filters of size 4×4, ReLU activation, followed by 2×2 max-pooling.
- **Layer 2:** Conv2D with 13 filters of size 2×2, ReLU, and 2×2 max-pooling.
- **Layer 3:** Conv2D with 13 filters of size 2×2, ReLU, and 2×2 max-pooling.

Fully Connected Classifier

- The output feature maps are flattened.
- A dense layer with 64 units and ReLU activation is applied.
- The final dense layer has 5 units with softmax activation, corresponding to the five heartbeat classes.

The network has roughly 8.5×10^5 trainable parameters, which makes it compact yet expressive [1]. Training is performed using categorical cross-entropy loss and mini-batch optimisation. In our work, we first reproduce this baseline architecture with single-window STFT inputs, and then evaluate the impact of the proposed multi-window STFT representation while keeping the CNN backbone largely unchanged.

4 Implementation Details

This section describes the complete processing pipeline used to reproduce the STFT–CNN arrhythmia classification method. The implementation closely follows the methodology of Huang *et al.* [1], with modifications where necessary due to computational constraints.

4.1 ECG Data Acquisition and Beat-Centred Segmentation

ECG signals were obtained from the MIT-BIH Arrhythmia Database using the `wfdb` Python library. Each record was read using `wfdb.rdrecord()`, and the V5 lead was selected when available; otherwise the MLII lead was used. All signals were resampled to a sampling rate of 360 Hz to ensure consistency across records.

Beat annotations were extracted using `wfdb.rdann()`. MIT-BIH annotation symbols were mapped into the five target heartbeat classes according to:

Class	MIT-BIH Symbols
NOR	N
LBB	L
RBB	R
PVC	V
APC	A, a

For every annotated beat belonging to a class, a 10-second window (3600 samples) centred around the beat was extracted. Windows that exceeded the signal boundary were discarded. This produced a pool of fixed-length 1D segments for each class.

4.2 Train–Test Split Construction

To replicate the evaluation protocol described in [1], a fixed balanced split was constructed. For the classes NOR, LBB, RBB, and APC, 450 windows were selected for training and 90 for testing. For PVC, 300 training and 60 testing segments were used. Random selection was performed using a fixed NumPy random generator for reproducibility.

The resulting dataset shapes were:

$$\text{Train: } (2100,\ 3600), \qquad \text{Test: } (420,\ 3600).$$

Class labels were mapped to integer indices (0–4) and converted into one-hot representations for training.

4.3 STFT-Based Spectrogram Generation

Each 1D ECG segment was transformed into a 2D time–frequency spectrogram using the short-time Fourier transform (STFT). A symmetric Hann window of length 512 samples was used, with a 50% overlap:

$$\text{window length } = 512, \qquad \text{overlap } = 256.$$

The STFT was computed using:

$$X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]\, w[n-m]\, e^{-j\omega n},$$

where $w[\cdot]$ denotes the Hann window. The magnitude of the STFT was converted to decibels:

$$S_{\text{dB}} = 20 \log_{10} \left(\frac{|X|}{\max |X|} \right),$$

normalized to $[0, 1]$, and resized to a 256×256 grayscale image using bicubic interpolation. Each spectrogram was stored as a tensor of shape $(256, 256, 1)$ suitable for CNN input.

4.4 2D Convolutional Neural Network Architecture

A convolutional neural network consistent with the structure described in [1] was implemented in TensorFlow/Keras. The architecture consists of three convolutional blocks:

- **Block 1:** Conv2D (8 filters, 4×4, ReLU, same padding) + MaxPooling (2×2)
- **Block 2:** Conv2D (13 filters, 2×2, ReLU) + MaxPooling (2×2)
- **Block 3:** Conv2D (13 filters, 2×2, ReLU) + MaxPooling (2×2)

The feature map was flattened and passed through a fully connected layer with 64 ReLU units, followed by a softmax layer of 5 units corresponding to the arrhythmia classes. The model contains approximately 853,000 trainable parameters.

4.5 Training Configuration

Training was performed using the Adam optimizer with a learning rate of 0.001, categorical cross-entropy loss, and 100 epochs, consistent with the paper's configuration. A `ReduceLROnPlateau` callback was used to stabilize convergence.

Due to GPU memory limitations (two NVIDIA T4 GPUs with 16 GB VRAM each), the batch size of 2500 reported in the paper could not be reproduced. The maximum feasible batch size was 1000, which was used for all experiments. Despite this reduction, training remained stable and achieved high test accuracy.

5 Implementation results (and comparison with results in the paper if suitable)

5.1 Challenges Faced

5.1.1 Insufficient Specification of CNN Architectural Parameters

Insufficient Specification of CNN Architectural Parameters One significant challenge we encountered during the replication of the study was the lack of detailed information regarding the convolutional neural network (CNN) architecture used in the paper. Although the authors claimed that a $256 \times 256 \times 1$ spectrogram was provided as input and that the output of the "first hidden layer" was $32 \times 8 \times 1024$, the description of the intermediate operations was incomplete.

The methodology section mentioned the use of an 8×4 convolution kernel followed by a 2×2 max-pooling layer, but no additional information was provided about the number of layers, kernel shapes in subsequent layers, padding settings, stride values, or pooling configurations.

This omission created ambiguity in reconstructing the model. For example, a single 4×4 convolution with 8 filters followed by 2×2 max-pooling cannot mathematically transform a $256 \times 256 \times 1$ input into a $32 \times 8 \times 1024$ output.

5.1.2 Insufficient Computer Resources

A significant practical limitation we encountered during the replication process was the mismatch between the computational resources available to us. The authors report using a batch size of 2500 during training and evaluation.

Our training environment consisted of two NVIDIA T4 GPUs, each with 16 GB of VRAM. Despite various optimisation attempts, we were unable to support a batch size of 2500 but we were able to go as far as 1500 batch size. So the results include contrast experiments up to batch size of 1500.

To evaluate the model, we tested multiple batch sizes while keeping the learning rate (0.001) and epochs (100) fixed. The goal was to identify the best-performing configuration and compare the behaviour across different parameter settings.

5.2 Fixed Learning Rate vs Batch Sizes

Table 1 shows that smaller batch sizes achieve the best overall performance. Batch size 100 produced the highest validation and test accuracy (about 98.3%), closely matching the

results reported in the paper. As the batch size increases, accuracy gradually decreases; batch sizes of 1000 and 1500 drop to the 90–93% range.

Table 1: Performance for different batch sizes

Batch Size	Avg Val Acc	Train Acc	Val Acc	Test Acc
1500	75.08%	93.67%	90.95%	90.95%
1000	76.43%	96.86%	93.81%	93.81%
750	83.44%	98.38%	96.43%	96.43%
500	82.92%	97.05%	94.29%	94.29%
250	86.47%	99.33%	97.62%	97.62%
100	**94.26%**	**100%**	**98.33%**	**98.33%**

5.2.1 Loss and Accuracy Curves

The loss and accuracy curves support the above trend:

- Smaller batches converge faster and remain stable throughout training.
- Larger batches learn more slowly and show reduced generalisation.
- Some mid-sized batches (e.g., 250–750) show occasional spikes in validation loss around epochs 70–80.

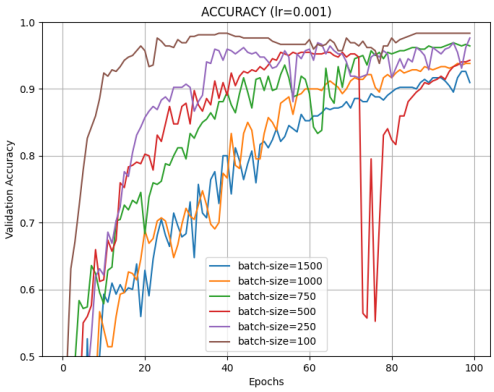

Figure 3: accuracy curves

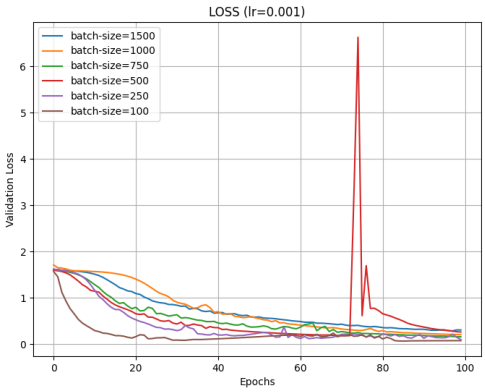

Figure 4: loss curves

5.3 Comparison With the Paper

Compared to the original study, our implementation achieves very similar performance at smaller batch sizes. For example, batch size 100 reaches 98.3% test accuracy in our experiments, closely matching the paper's reported accuracy of 98.8–99.0%. Batch sizes 250 and 500 also follow the same trend, reaching 97–94%, which is consistent with the paper's reported 98.5–99.0%.

The main differences appear at larger batch sizes. While the paper reports almost no degradation even at batch sizes 1500–2500, our accuracy drops to 90–93% for batch

sizes 1000 and 1500. This gap is likely due to hardware constraints: limited GPU memory required smaller effective batch processing, which affects gradient updates and overall convergence. Despite this, the general trend smaller batches performing better matches the findings of the original paper.

5.3.1 Observations

- Best performance: batch size 100 (98.3% test accuracy).
- Accuracy decreases as batch size increases.
- Small batches provide faster convergence and better generalisation.
- The reproduced results successfully validate the original paper for smaller batch configurations.

5.4 Fixed Batch Size vs Learning Rates

To evaluate the replication of the paper we have chosen a series of contrast experiments, with different parameter sets as it was done in the original paper. In this section we focus the experiments done with different learning rates when keeping the batch size unchanged. The different learning rates employed were 0.001, 0.0025, 0.005, 0.0075, 0.1, 0.15, 0.2 as they had done in the paper [1]. For all experiments number of iterations were set to 100.

For each batch size, the performance was evaluated on the basis of accuracy, loss, precision, recall, f1-score and AUC.

5.4.1 Batch Size of 100

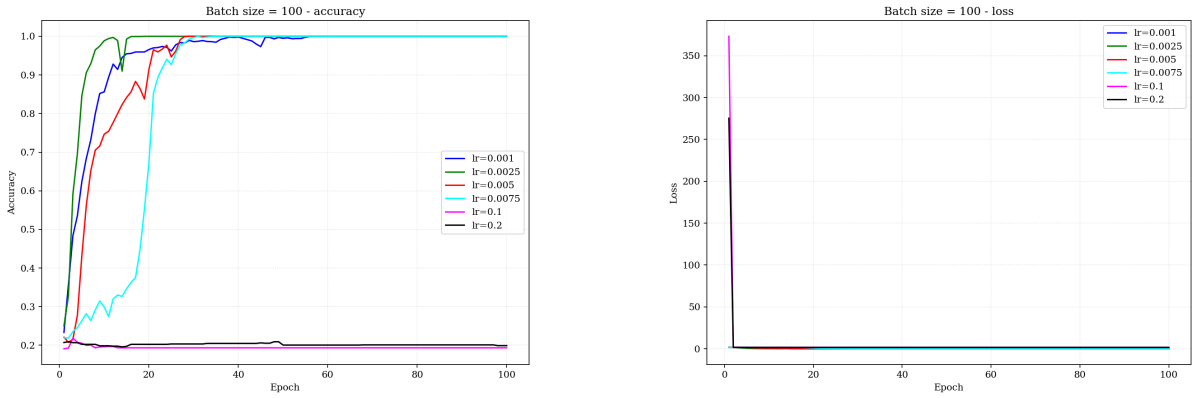

Figure 5: Accuracy and Loss curves for 7 different learning rates with the batch size of 100

In addition to the accuracy and loss curves over epochs we plot the ROC and confusion matrices for different learning rates as well.

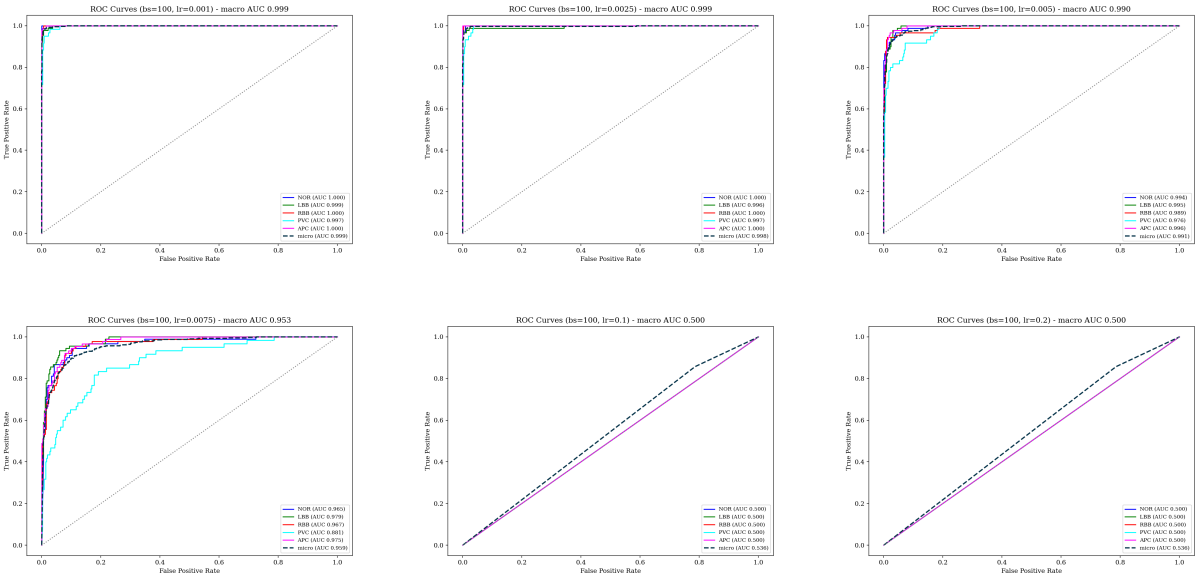

Figure 6: ROC Curves for each different learning rates

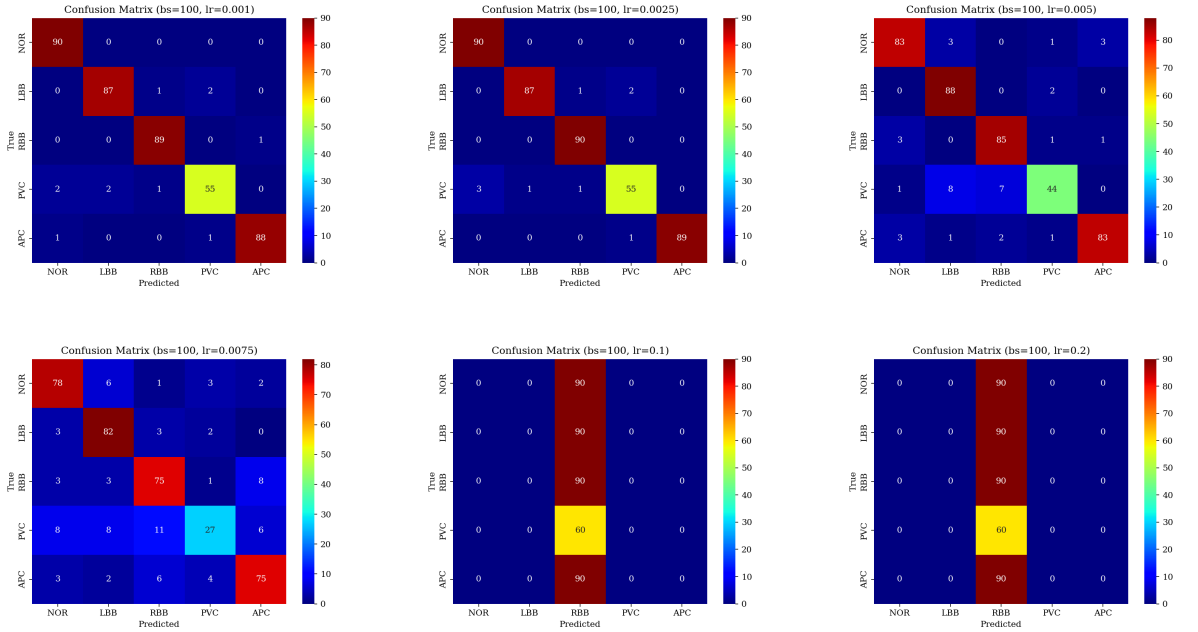

Figure 7: Confusion Matrices for each different learning rates

Learning Rate	Accuracy	Precision	Recall	F1-Score	AUC
0.001000	0.9738	0.9721	0.97	0.9709	0.9992
0.002500	0.9786	0.9766	0.9744	0.9754	0.9985
0.005000	0.9119	0.9117	0.9	0.9035	0.9901
0.007500	0.8024	0.7936	0.7789	0.7788	0.9534
0.100000	0.2143	0.0429	0.2	0.0706	0.5
0.200000	0.2143	0.0429	0.2	0.0706	0.5

Table 2: Overall Validation Accuracy, Precision, Recall, F1-Score and AUC scores for different learning rates with the batch size of 100

Next we analyze the average accuracy of the replication of the paper and the results that author got for the batch size of 2500.

Batch Size	*Learning Rate*	*Avg. Accuracy*
2500	0.001	0.9900
2500	0.0025	0.9893
2500	0.005	0.9878
2500	0.0075	0.9885
2500	0.01	0.9882
2500	0.02	0.9885

Table 3: Results obtained by the authors for a batch size of 2500

Batch Size	*Learning Rate*	*Avg. Accuracy*
100	0.001	0.9738
100	0.0025	0.9786
100	0.005	0.9119
100	0.0075	0.8024
100	0.01	0.2143
100	0.02	0.2143

Table 4: Results of our replication for a batch size of 100

5.4.2 Batch Size of 250

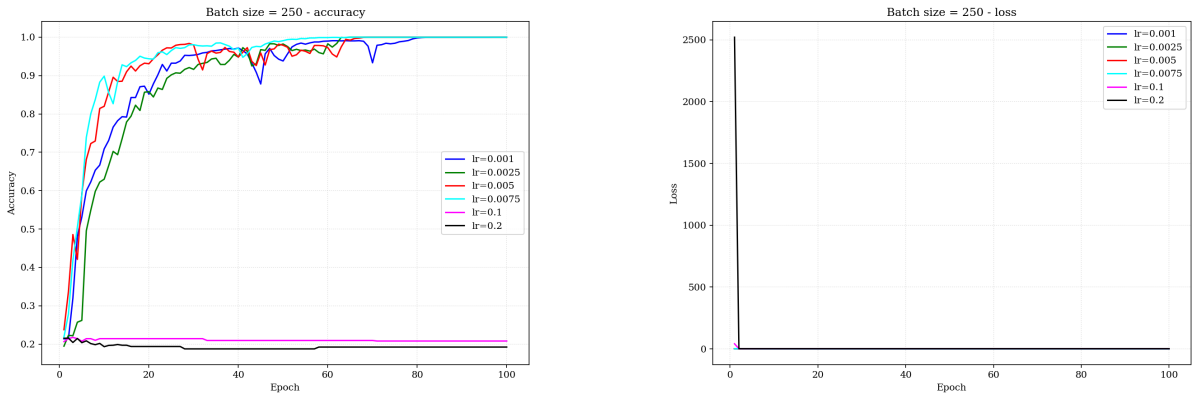

Figure 8: Accuracy and Loss curves for 7 different learning rates with the batch size of 250

In addition to the accuracy and loss curves over epochs we plot the ROC and confusion matrices for different learning rates as well.

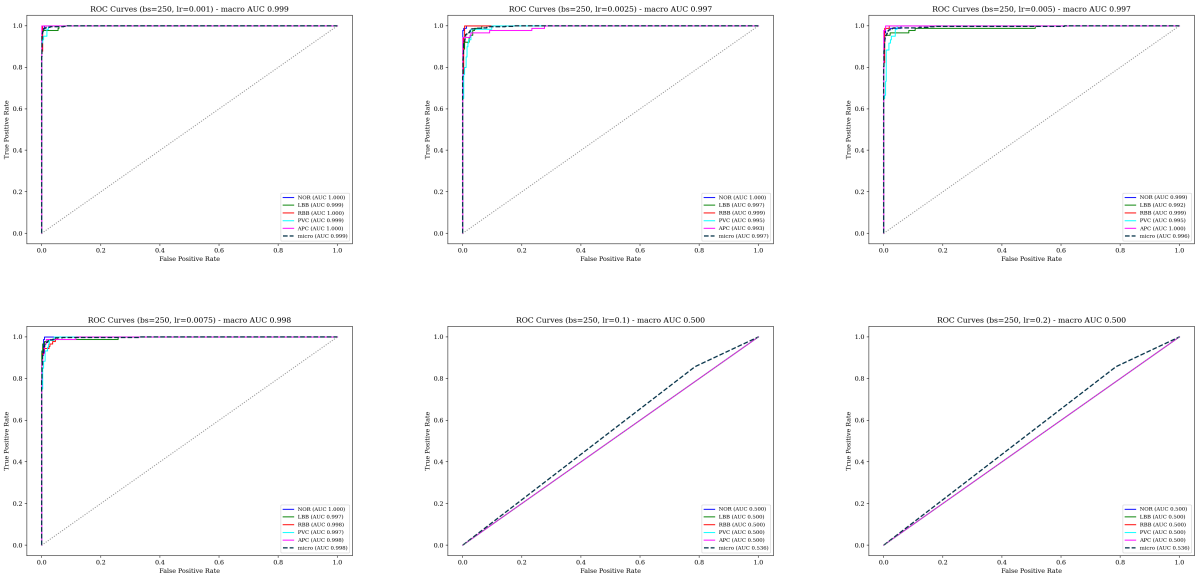

Figure 9: ROC Curves for each different learning rates

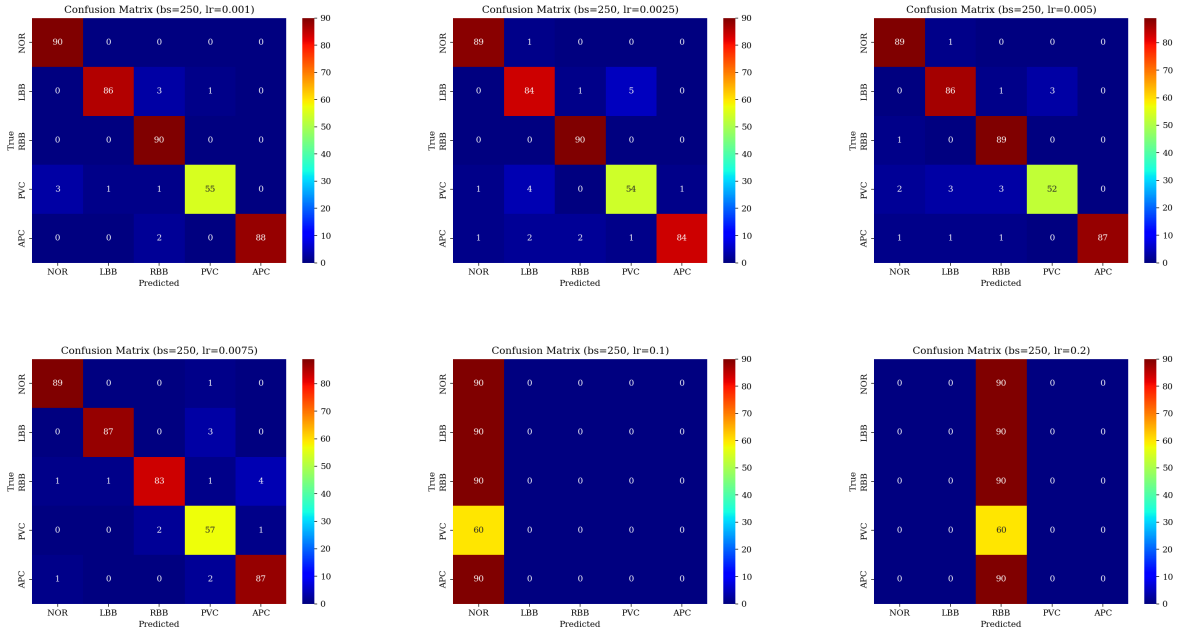

Figure 10: Confusion Matrices for each different learning rates

Learning Rate	Accuracy	Precision	Recall	F1-Score	AUC
0.001000	0.9738	0.9752	0.97	0.972	0.9994
0.002500	0.9548	0.9514	0.9511	0.951	0.9966
0.005000	0.9595	0.9589	0.9533	0.9555	0.997
0.007500	0.9595	0.9559	0.9589	0.957	0.9978
0.100000	0.2143	0.0429	0.2	0.0706	0.5
0.200000	0.2143	0.0429	0.2	0.0706	0.5

Table 5: Accuracy and Loss curves for 7 different learning rates with the batch size of 250

Batch Size	*Learning Rate*	*Avg. Accuracy*
2500	0.001	0.9900
2500	0.0025	0.9893
2500	0.005	0.9878
2500	0.0075	0.9885
2500	0.01	0.9882
2500	0.02	0.9885

Table 6: Results obtained by the authors for a batch size of 2500

Batch Size	*Learning Rate*	*Avg. Accuracy*
250	0.001	0.9738
250	0.0025	0.9548
250	0.005	0.9595
250	0.0075	0.9595
250	0.01	0.2143
250	0.02	0.2143

Table 7: Results of our replication for a batch size of 250

6 Discussion / conclusion and possible improvements

6.1 Discussion

The following table gives metrics of the models which attained best performance according to each metrics.

Batch Size	Learning Rate	Accuracy	Precision	Recall	F1-Score	AUC
100	0.001	0.9738	0.9752	0.97	0.972	0.9994
100	0.0025	0.9786	0.9766	0.9744	0.9754	0.9985
250	0.001	0.9738	0.9721	0.97	0.9709	0.9992

Table 8: Best Performing models

In the paper the best performance was obtained for the batch size of 2500 and a learning rate of 0.001.

Batch Size	Learning Rate	Accuracy
2500	0.001	0.9900

Table 9: Best Performance Parameter Set given in the Paper

With the challenges mentioned above, our best-performing model achieved an accuracy loss of 0.0162. However, the original paper did not report key performance metrics

such as Precision, Recall, F1-Score, or AUC. We included these additional metrics in our evaluation to enhance the comparability and interpretability of the results, particularly for biological processes like arrhythmia classification.

6.2 Proposed Improvement and Implementation

The original paper employs a fixed single Short-Time Fourier Transform (STFT) window of 256 samples to generate the time frequency representation of ECG signals. However, using only one fixed window size introduces a fundamental limitation,

- The STFT provides the *same* time and frequency resolution at all time locations, regardless of the underlying signal dynamics. This is problematic for ECG signals, where events have inherently different scales.
  - High-frequency, fast events such as the QRS complex require a small window to achieve high temporal resolution.
  - Low-frequency, slow events such as the P and T waves require a arge window to accurately capture their spectral characteristics.
- A single fixed-window STFT cannot accurately localise both time and frequency information for events with diverse frequency content.

These limitations directly arise from the **Fourier Uncertainty Principle**;

$$\Delta t \, \Delta f \geq \text{constant}, \tag{3}$$

meaning that improving time resolution necessarily worsens frequency resolution, and vice versa. As a result, a single STFT window size cannot capture the complete morphological variability of ECG events without losing information, causing some wave features to appear blurred or poorly localized.

6.2.1 Adaptive Multi-Window STFT Front-End

To address this issue, we propose an *adaptive multi-window STFT* front-end. Specifically, we compute three separate spectrograms using window sizes of **128**, **256**, and **512** samples. These spectrograms are then stacked to form a three-channel (RGB-like) input to the convolutional neural network.

6.2.2 Why Multi-Window STFT?

Each window size emphasizes different characteristics of the ECG morphology:

- 128-sample windows capture rapid, high-frequency components such as the sharp QRS complex.
- 256-sample windows provides a balanced time–frequency trade-off.
- 512-sample window captures slow, low-frequency components such as the P and T waves.

6.2.3 Results

Batch Size	Avg Val Acc %	Final Train Acc %	Final Val Acc %
100	96.567	100	99.048
250	95.950	98.805	98.571
500	86.361	99.476	98.810
750	87.490	100	98.810
1000	83.912	100	96.190
1500	89.017	100	98.571

Table 10: Performance metrics of 6 different batch sizes with the learning rate of 0.001

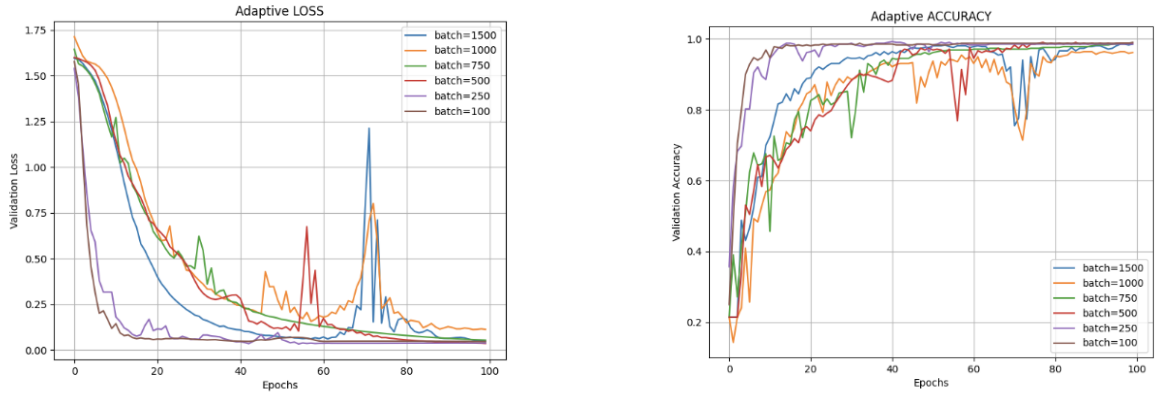

Figure 11: Accuracy and Loss curves for 6 different batch sizes with the learning rate of 0.001

6.2.4 Results Comparison

In contrast to the constant-window STFT used in the original replication, our adaptive multi-window STFT method achieves higher accuracy.

Batch Size	*Avg Train Acc %*	*Val Acc %*
100	96.567	99.048
250	95.950	98.571
500	86.361	98.810
750	87.490	98.810
1000	83.912	96.190
1500	89.017	98.571

Table 11: Results obtained from Multi-Window STFT

Batch Size	*Avg Train Acc %*	*Val Acc %*
100	95.257	98.333
250	86.469	97.619
500	82.934	94.286
750	83.443	96.429
1000	76.429	93.810
1500	75.076	90.952

Table 12: Results obtained from Fixed-Window STFT

6.3 Possible Improvement

Since the STFT is bound to the Fourier Uncertainty Principle, we explored the possibility of other transform to get the spectrogram.

6.3.1 Hilbert Huang Transform

The Hilbert-Huang Transform (HHT) is a signal processing technique which is used for analyzing non-linear and non-stationary signals such as ECG data. Unlike traditional methods, HHT provides an adaptive framework for extracting characteristic components of underlying biological processes through a two-stage approach that combines Empirical Mode Decomposition(EMD) with the Hilbert Transform.

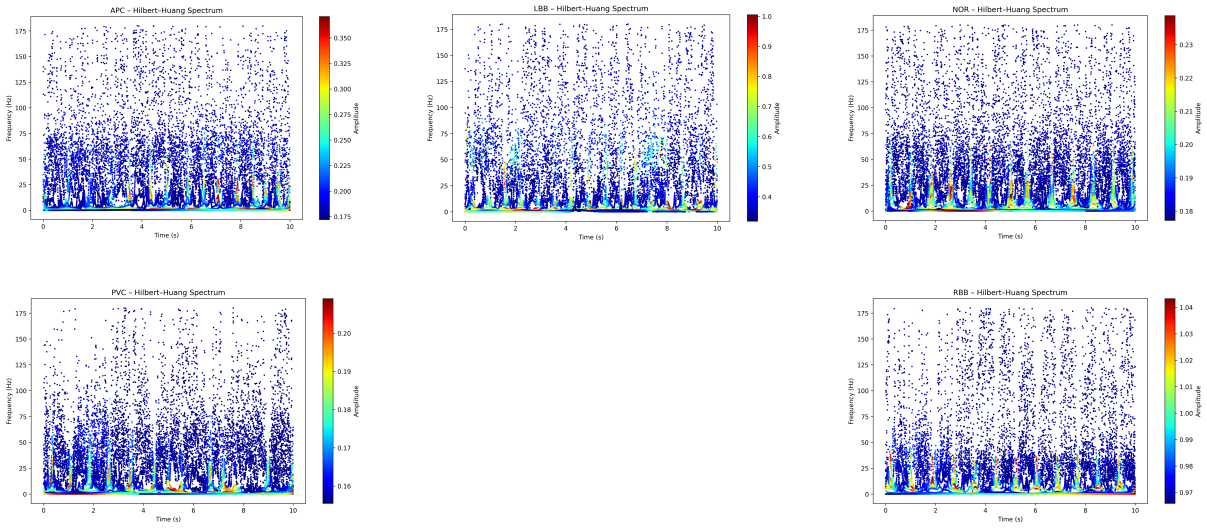

Figure 12: Spectrograms for each class using HHT

Although the Hilbert–Huang Transform provides a highly adaptive time–frequency representation, one significant limitation encountered during our experimentation was its computational inefficiency. During the model training phase, it became evident that generating HHT-based representations required substantially more time compared to the Short-Time Fourier Transform (STFT). When we rolled back our workflow to analyse the preprocessing stage, we observed that each ECG class required a noticeably longer time to compute a single HHT transform

Class	*HHT*	*STFT*
NOR	1.999 s	0.0044 s
LBB	0.214 s	0.0020 s
RBB	0.414 s	0.0016 s
PVC	0.867 s	0.0016 s
APC	0.616 s	0.0017 s

Such high computational cost makes large-scale training with HHT features impractical, especially when dealing with thousands of samples. This limitation would be because of computational burden of envelope estimation, where over-sampling and cubic spline interpolation introduce additional overhead during the EMD sifting process. Therefore, while the HHT-based representation may provide more accurate and physiologically meaningful predictions, the overall model would require significant training and inference time, limiting its suitability for real-time or resource-constrained applications.

References

[1] J. Huang, B. Chen, B. Yao, and W. He, "Ecg arrhythmia classification using stft-based spectrogram and convolutional neural network," *IEEE Access*, vol. 7, pp. 92 871–92 880, 2019.

[2] G. B. Moody and R. G. Mark, "The Impact of the MIT-BIH Arrhythmia Database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.