1 ECG Time Series Classification - AMLS SoSe 2025

This project implements a complete machine learning pipeline for ECG time series classification.

1.1 Group Members

0520102 - Armin Ebrahimi saba

1.2 Installation

1. Install Python 3.8+
2. Install dependencies:

```
pip install -r requirements.txt
```

1.3 Usage

1. Download the ECG dataset from TU-Cloud and place in the `Uni/AMLS` directory
2. chmod +x AMLS.py
3. Run the script to get more instruction:

```
./AMLS.py
```

1.4 Dataset Exploration

This section summarizes the dataset and describes how the data is split for training and validation.

Dataset Summary

The dataset consists of 6179 ECG recordings, each labeled as one of the four classes:

- **Normal (Class 0)**: Regular rhythm with clear R-peaks, low noise, and periodic structure indicating healthy heart activity.
- **AF (Class 1)**: Irregular, chaotic signal with high-frequency components and absence of consistent P-waves or PR intervals; indicative of atrial fibrillation.
- **Other (Class 2)**: Signals in this class show variations such as ectopic beats or conduction abnormalities, maintaining partial periodicity but differing from the Normal class.
- **Noisy (Class 3)**: Corrupted by noise making them difficult to interpret

Length Statistics

The ECG signals vary significantly in length:

- Minimum length: 2714 samples
- Maximum length: 18286 samples
- Mean length: 9760.2 samples
- Standard deviation: 3292.2 samples
- Median: 9000 samples

This high variance in sequence length suggests the need for either padding/truncation or adaptive pooling before classification. Padding/truncation or adaptive pooling is necessary before classification. It may impact temporal models (like LSTMs) that assume more uniform input lengths.

Class Distribution

Class	Description	Count	Percentage
0	Normal	3638	58.9%
1	AF	549	8.9%
2	Other	1765	28.6%
3	Noisy	227	3.7%

Table 1: Class distribution in the dataset

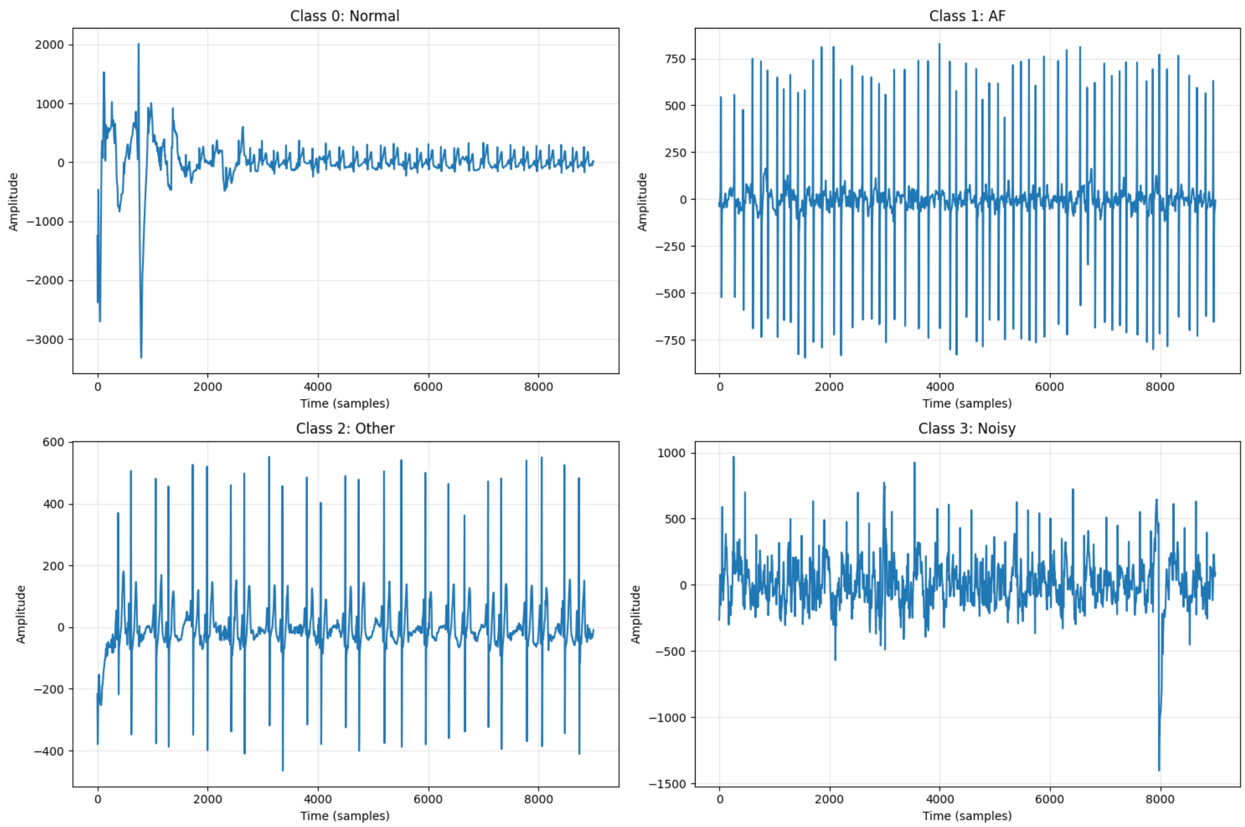

Figure 1: ECG samples for each class

Validation Split

A stratified 80/20 sampling approach is used to preserve class proportions to avoid bias in evaluation.
Training: 4943 —— Validation: 1236

Class	Description	Count	Percentage
0	Normal	2910	58.9%
1	AF	439	8.9%
2	Other	1412	28.6%
3	Noisy	182	3.7%

Table 2: Class distribution in the training set

1.5 Task 2: Modeling and Tuning (40 points)

We implement and compare two models: a simple baseline, which is recommended in the project description, and another architecture that leverages self-attention.

1.5.1 Trainer

The `ECGTrainer` class is responsible for handling the training and evaluation processes of the model.

- **Early Stopping:** If the validation F1-score doesn't improve for a number of epochs, stop training to prevent overfitting.
- **Categorical Cross-Entropy Loss**
- **Adam Optimizer:** It adapts the learning rate, making convergence faster and more stable.

1.5.2 Model 1: `SimpleSTFTModel`

Applies an STFT to extract the frequency-domain features of the ECG signal. The STFT magnitude is passed through a series of 2D convolutional layers followed by an LSTM and a fully connected classification head. It uses 2D convolutional layers to extract spatial patterns from the STFT spectrogram and an LSTM layer to capture temporal dependencies across time frames. The convolution and pooling layers reduce the dimensionality of the input, minimize overfitting, and improve robustness to noise. it has **3,918,212 params**.

Layer	Description
Input	ECG signal, shape (B, T)
STFT	Log-magnitude spectrogram, shape $(B, 1, F, T')$
Conv2D + ReLU + MaxPool	64 filters, kernel size 3×3, stride 2
Conv2D + ReLU + MaxPool	128 filters, kernel size 3×3, stride 2
Conv2D + ReLU + MaxPool	256 filters, kernel size 3×3, stride 2
Conv2D + ReLU + MaxPool	512 filters, kernel size 3×3, stride 2
Conv2D + ReLU + MaxPool	256 filters, kernel size 3×3, stride 2
AdaptiveAvgPool2D	Output size $(8, 8)$
Reshape	To sequence of vectors $(B, T'', 512)$
BiLSTM	Hidden size 256, bidirectional, output dim 512
BiLSTM	Hidden size 256, bidirectional, output dim 512
Attention	Weighted average over time (not present in SimpleSTFTModel)
Fully Connected (FC1)	512 $\rightarrow$ 1024 + BatchNorm + ReLU + Dropout
Fully Connected (FC2)	1024 $\rightarrow$ 512 + BatchNorm + ReLU + Dropout
Fully Connected (FC3)	512 $\rightarrow$ 256 + BatchNorm + ReLU + Dropout
Output Layer (FC4)	256 $\rightarrow$ Number of classes

Table 3: layers of `ImprovedSTFTModel`. Highlighted row is not present in the Simple model.

Training Results

- **Best Training Accuracy:** 82.18% (at epoch 90)
- **Best Validation Accuracy:** 77.91%
- **Best Validation F1-Score:** 0.6682
- **Loss Gap (Train - Val):** Final gap = -0.2570
- **Total Epochs Trained:** 80

Epoch	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Val F1
0	1.0836	53.81	1.2497	47.33	0.2493
10	0.7763	68.66	0.8484	64.24	0.3434
20	0.6735	72.85	0.8022	69.26	0.4135
30	0.5947	75.14	0.7228	72.09	0.5424
40	0.5433	77.12	0.7573	74.19	0.6016
50	0.4903	80.54	0.7600	75.65	0.6356
60	0.4913	81.00	0.7059	77.91	0.6733
70	0.4840	80.88	0.7136	77.75	0.6682
79	0.4715	82.18	0.7285	76.86	0.6672

Table 4: Training and Validation Metrics Across Epochs

Justification

Time-frequency representations are useful for ECG classification due to:

- AF and noisy signals having distinct spectral properties
- Time-domain variability being difficult to capture via raw CNNs alone

Using an LSTM captures temporal dependencies in frequency-domain features, helping generalize across varying heart rhythms. Dropout reduces overfitting, especially in ECG with imbalanced class distributions.

Training History Visualization

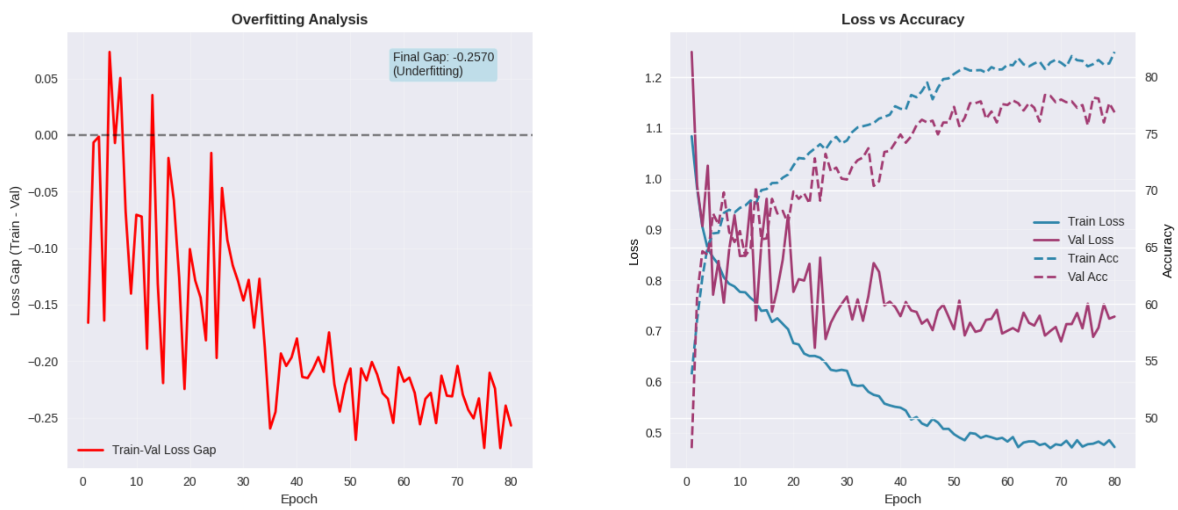

Figure 2: Training history of SimpleSTFTModel across 80 epochs.

1.5.3 Model 2: `ImprovedSTFTModel`

Justification for Using Focal Loss and Attention Mechanism

Focal Loss for Class Imbalance. As shown in Table 1, the dataset suffers from significant class imbalance:

- **Class 0 (Normal)** accounts for nearly 59% of the data.
- **Minority classes** such as **AF (8.9%)** and **Noisy (3.7%)** are underrepresented.

Using standard cross-entropy loss in such imbalanced settings causes the model to be biased toward the majority class, resulting in low recall for minority classes. This is evident from the poor F1 score in early epochs (e.g. 0.4135 at Epoch 20), despite the relatively high validation accuracy (69.26%).

Focal loss addresses this issue by down-weighting easy (i.e., correctly classified) examples and focusing the learning on harder, misclassified samples. This encourages the model to learn more discriminative features for underrepresented classes.

Attention Mechanism for Temporal Localization. The ECG signals exhibit high variability in length (ranging from 2714 to 18286 samples) and include noisy patterns. Diagnostically relevant features (e.g., irregular P waves in AF or noise spikes in Class 3) may occur at arbitrary time points and are not uniformly distributed across the sequence.

Recurrent models like LSTMs compress the temporal information into fixed-size hidden states, which may oversimplify the representation and important temporal features. Self-attention helps solve this issue. The improvement in the F1 validation score from 0.7652 to 0.8176 supports the effectiveness of this choice.

Conclusion. Focal loss and attention mechanisms address two fundamental challenges in this ECG classification task: class imbalance and temporal feature localization. Table 5 shows a significant improvement in the maximum accuracy of the validation (7.2%) and F1 (0.0981). Therefore, I chose the second model.

- **Best Training Accuracy:** 88.57% (at epoch 60)
- **Best Validation Accuracy:** 85.11% (at epoch 60)
- **Best Validation F1-Score:** 0.7714 (at epoch 40)
- **Total Epochs Trained:** 80

Epoch	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Val F1
0	1.0543	56.48	0.9700	62.46	0.2467
10	0.5644	78.05	0.6765	76.62	0.5064
20	0.4801	81.91	0.5106	83.66	0.7328
30	0.4187	84.42	0.5779	83.33	0.7429
40	0.3729	86.14	0.5227	84.87	0.7714
50	0.3404	87.76	0.5630	84.39	0.7545
60	0.3309	88.57	0.5305	85.11	0.7634
70	0.3137	88.43	0.5405	84.95	0.7638
80	0.3215	88.14	0.5256	85.03	0.7652

Table 5: Training and Validation Metrics Across Epochs for ImprovedSTFTModel

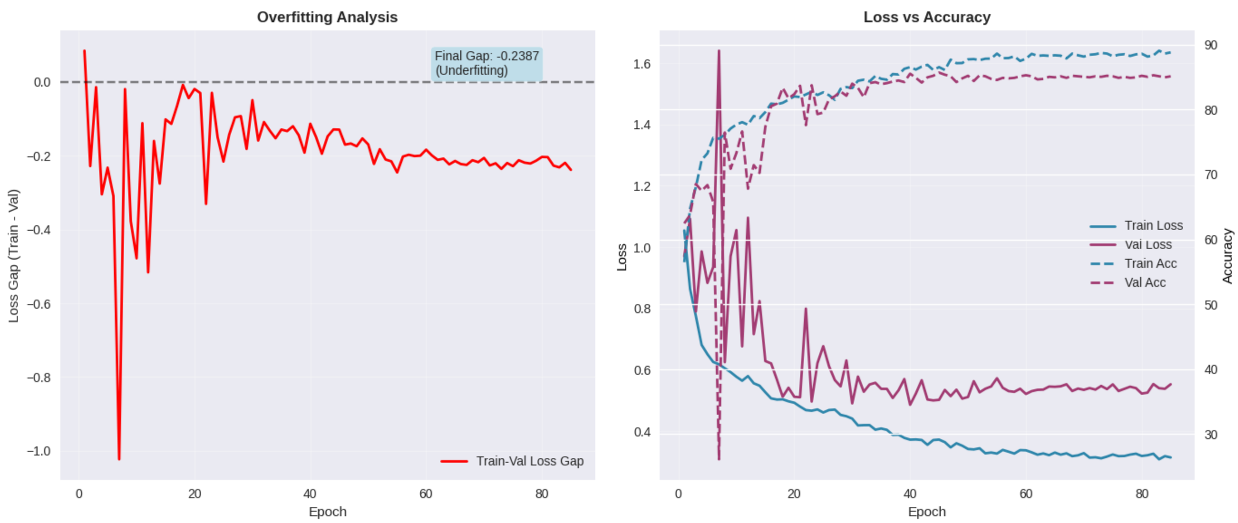

Figure 3: Training history and performance summary of ImprovedSTFTModel

1.6 Task 3: Data Augmentation (30 points)

Justification of ECG Data Augmentation Techniques

To improve model generalization, we apply domain-specific data augmentations. The selected methods simulate realistic variations found in ECG recordings.

Selected Augmentation Techniques

- **Gaussian Noise (add_noise)**: Adds low-amplitude Gaussian noise to simulate sensor artifacts such as baseline wander, electrode interference, or muscle activity. This enhances the robustness of the model to noisy real-world ECG signals without distorting the signal morphology.
- **Time Stretching/Compression (time_stretch)**: Simulates variations in heart rate (for example, bradycardia or tachycardia) by slightly speeding up or slowing down the waveform. This helps the model learn temporal invariance with respect to cardiac rhythm.
- **Amplitude Scaling (amplitude_scale)**: Adjusts the signal amplitude to mimic physiological variability between individuals, changes in electrode contact, or gain settings during acquisition. This improves the ability of the model to generalize between patients.
- **Time Shifting (time_shift)**: Applies random circular shifts to the signal to simulate alignment variations. This reduces sensitivity to phase and positional shifts during signal acquisition or cropping.

Placement in the Machine Learning Pipeline

- Increases the diversity of training data without modifying the validation set.
- Class distribution is not changed
- Avoids computational overhead during model training.
- Ensures deterministic evaluation by keeping validation data untouched.

The `ECGAugmentor` applies a suite of realistic and physiologically-inspired transformations to ECG signals. These augmentations introduce variability that improves generalization. Also, I guess with further training better results could be achieved as validation accuracy was not yet fluctuating. In addition to that, Table 6 shows a relatively low-quality augmented-model result, and Table 5 shows the best result I achieved when not using augmentation:

Table 6: Training and Validation Metrics Across Epochs

Epoch	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Val F1
0	1.1155	50.78	4.2564	4.85	0.0600
10	0.5739	78.27	0.6626	73.46	0.5904
20	0.4698	82.15	0.4910	83.58	0.7723
30	0.3906	86.24	0.5237	83.09	0.7681
40	0.3364	88.11	0.4932	83.74	0.7789
50	0.2889	89.79	0.4784	84.95	0.8043
60	0.2718	90.35	0.4952	84.95	0.8067
70	0.2607	90.63	0.4916	84.87	0.8074
79	0.2547	91.23	0.4908	85.11	0.8094

Results: F1 improved which is due to the improved class balance which was expected.

Table 7: Comparison of model performance with and without data augmentation

Metric	Without Augmentation	With Augmentation	Improvement
Training Accuracy (%)	88.57	91.23	+2.66
Validation Accuracy (%)	85.11	85.68	+0.57
Best F1-Score	0.7434	0.8094	+0.066

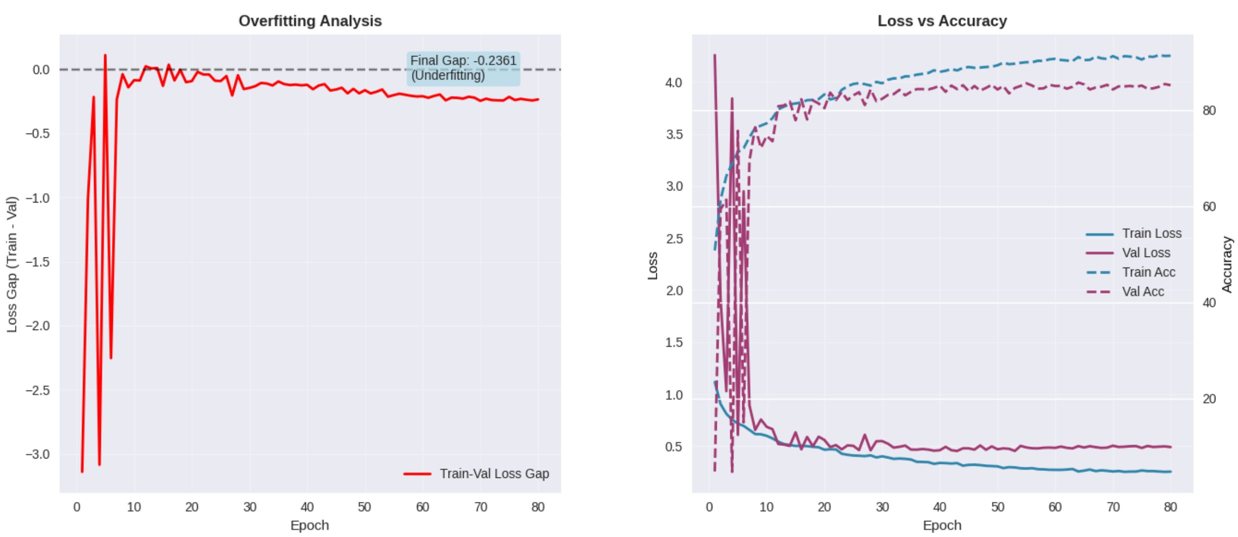

Figure 4: Training curves and summary with augmentation. Augmentations helped reduce overfitting and increase final validation performance.

1.7 Task 4: Data Reduction (15 points)

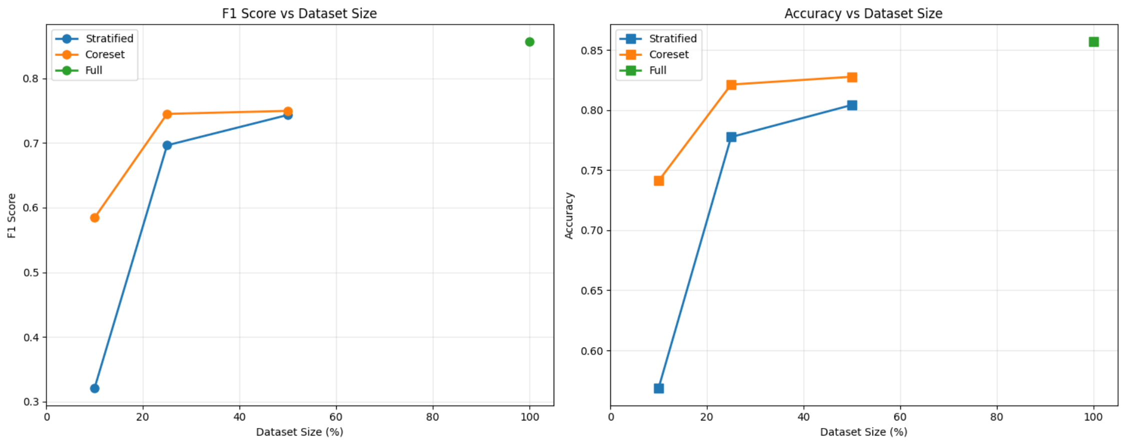

Figure 5: F1 Score and Accuracy vs Dataset Size for Coreset, Stratified, and Full datasets

Table 8: F1 Scores across dataset sizes

Dataset Size (%)	**Coreset**	**Stratified**	**Full**
10%	0.5844	0.3209	–
25%	0.7451	0.6965	–
50%	0.7498	0.7435	–
100%	–	–	0.8568

Table 9: Accuracy across dataset sizes

Dataset Size (%)	**Coreset**	**Stratified**	**Full**
10%	0.7411	0.5688	–
25%	0.8212	0.7775	–
50%	0.8277	0.8042	–
100%	–	–	0.8568