Supplementary Material 2

Backbone Probability Interfaces: Hyperparameters and Training Settings

Chen Xu
School of Information Science and Engineering, Lanzhou University
Corresponding author: xchen2024@lzu.edu.cn
ORCID: 0009-0009-8592-4336

This Online Resource accompanies the manuscript "Clinical Evidence Morphology–Rhythm for ECG (CEMR-ECG): a traceable evidence framework for imbalanced heartbeat classification" submitted to Physical and Engineering Sciences in Medicine. It documents the complete hyperparameter settings of the 20 evaluated raw backbones, the training procedure, and the software stack used in all framework experiments.

1 Machine-learning baselines

Eleven machine-learning models were evaluated on flattened two-lead heartbeat windows without CEMR-ECG evidence features. Hyperparameters were fixed before the framework comparison and were not tuned on test data. Table 1 lists the settings.

Table 1: Machine-learning baseline hyperparameter settings. All models used the same flattened two-lead heartbeat window. No CEMR-ECG evidence features, class weights, or sample weights were supplied to these baselines.

Backbone	Fixed setting
Logistic regression (LogReg)	Standardised features; $C = 1.0$; default solver.
Linear SVM	Standardised features; $C = 1.0$; `LinearSVC` with Platt scaling.
RBF SVM	Standardised features; $C = 1.0$; RBF kernel with default γ.
K-Nearest Neighbors (KNN)	$k = 7$; Euclidean distance.
Multilayer perceptron (MLP)	256/128 hidden units; ReLU activation; early stopping on validation loss.
Random forest (RF)	400 trees; $\sqrt{p}$ feature sampling.
Extremely randomized trees (ExtraTrees)	500 trees; $\sqrt{p}$ feature sampling.
Histogram gradient boosting (HGB)	260 iterations; default tree depth.
XGBoost	350 boosting iterations; learning rate 0.05.
LightGBM	350 boosting iterations; learning rate 0.05.
CatBoost	350 boosting iterations; learning rate 0.05.

These fixed settings were chosen before the framework comparison as common baseline-scale choices that keep model fitting tractable across the 300 paired runs of the primary analysis. They were not selected by test-set optimization or by a separate hyperparameter search.

2 ECG-specific deep baselines

Four ECG-specific deep models were evaluated. They were trained from random initialisation on raw beat waveforms, without external pretrained weights. Table 2 summarizes the architectures and architecture-specific imbalance handling used inside each raw baseline definition.

Table 2: ECG-specific deep baselines. Architecture-specific imbalance mechanisms were kept within each raw-baseline definition; none used CEMR-ECG evidence features.

Backbone	Reference and imbalance mechanism
CAT-Net	Convolution–attention–transformer hybrid for single-lead ECG; inverse-frequency sampling with SMOTE–Tomek waveform resampling.
ECGTransForm	Bidirectional transformer for ECG; context-aware loss for class imbalance.
MSFT	Multi-scale feature-based transformer; focal-style loss for minority recovery.
MCTnet	Multi-scale information-bottleneck network for 12-lead ECG, applied to the two-lead beat window.

3 General time-series baselines

Five general time-series backbones were evaluated through the same beat-window interface. These models kept their fixed waveform-training losses and augmentations and did not use CEMR-ECG evidence features.

Table 3: General time-series backbones used through the AAMI-compatible beat-window interface.

Backbone	Description
Medformer	Multi-granularity patching transformer for medical time series.
PatchTST	Patch-based transformer for long-horizon time-series forecasting, adapted to classification.
ModernTCN	Modern temporal convolutional network with depthwise large-kernel design.
TimeMixer	Decomposable multiscale mixing model.
TimesNet	Temporal 2D-variation modelling network.

4 Training procedure for neural baselines

All neural baselines (ECG-specific and general time-series) followed the same fixed training procedure:

- Optimiser: Adam or AdamW, depending on the original architecture default.
- Batch size: 128 by default; automatically reduced to 64 or 32 after GPU out-of-memory retries.
- Early stopping criterion: validation M-F1(4); patience equal to one fifth of the maximum training budget.
- Gradient-norm clipping at 5.0.
- Initialisation: random; no pretrained weights from external corpora.
- Mixed precision: enabled when supported by the architecture.

Each raw model supplied its own validation and test backbone probability interface $\mathbf{p}_i^B$ to CEMR-ECG; CEMR-ECG used these probabilities together with the fixed evidence vector $\mathbf{z}_i$ defined in the main text.

5 Reusability conditions on the probability interface

For a candidate backbone to be paired with CEMR-ECG under the protocol used in this work, the backbone must:

1. Supply an AAMI-compatible probability vector over N/S/V/F/Q for validation and test beats.
2. Use beat windows aligned with the evidence extractor (234 samples around the annotated R peak, resampled when needed; see main text Section 2.2).
3. Be paired with the available RR rhythm, morphology, inter-lead waveform differences, shape-derivative summaries, and prototype evidence.

Prototype templates and evidence-encoder fitting use the training split, adapter and decoder selection use only the internal validation labels, and final test labels are reserved for reporting. CEMR-ECG is thus reusable as a shared evidence layer under these interface conditions; different label spaces or missing evidence streams would require separate specification and validation.

6 Software stack and reproducibility

All experiments were implemented in Python 3.11 using the following libraries:

- Numerical and machine-learning: NumPy 1.26, SciPy 1.11, scikit-learn 1.4.
- Deep learning: PyTorch 2.2.
- Gradient boosting: XGBoost 2.0, LightGBM 4.3, CatBoost 1.2.
- Plotting: Matplotlib 3.10.
- Bootstrap confidence intervals: a custom percentile bootstrap (10,000 resamples, fixed seed 20260525, NumPy `default_rng`) applied to the dataset–backbone mean M-F1(4) gains, with dataset-stratified resampling for the overall interval. Paired significance was summarized with a paired-t statistic and its two-sided p value, together with an exact two-sided sign test; effect size was reported as Cohen's d_z. These routines were implemented in-house and do not depend on the SciPy `bootstrap` interface.
- Classification metrics: scikit-learn `classification_report` utility cross-checked against an in-house implementation.

Paired significance testing (paired-t, sign test, Cohen's d_z) and the bootstrap confidence intervals were computed at the dataset–backbone mean level ($n = 60$, or $n = 20$ within each database) after averaging the five seeds. Seed-level direction-count summaries (60/60 means, 297/300 seed-level runs) are reported separately as descriptive consistency evidence because the 300 seed-level rows share datasets and backbones and are therefore correlated; no pooled independent-sample hypothesis test was applied to those 300 rows. Five fixed seeds were used throughout: 303, 1303, 2303, 3303, 4303. The same seeds were applied to Python random generators, NumPy, PyTorch, and estimator-level random states where available.