Deep Learning Technical Report: Hybrid Multimodal Classification

Project Goal: To build and evaluate a hybrid deep learning model capable of classifying patient status (target variable: sport_ability) using a combination of **12-lead Electrocardiogram (ECG) time-series data** and **structured clinical/anthropometric tabular data**.

1. Pipeline Overview and Data Handling

The pipeline is composed of four main stages: Data Loading, Multimodal Preprocessing, Model Training, and K-Fold Cross-Validation. A core feature of this solution is its ability to process two heterogeneous data types simultaneously.

1.1 Data Loading and ECG Processing

The script is specifically configured to load **real data** from .mat (MATLAB) files, which contain the raw 12-lead ECG signals, and tabular data from CSV files.

A. Signal Preprocessing

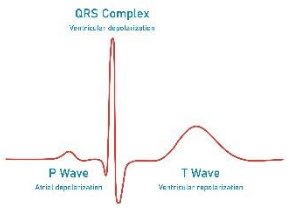
Raw ECG signals are often noisy and contain baseline wander. The following digital signal processing steps are applied to each of the 12 leads for every subject:

1. **Baseline Correction:** The mean of the signal is subtracted to center it around zero, removing DC offsets.
2. **Bandpass Filtering:** A Butterworth Bandpass Filter (1.0 Hz to 40.0 Hz) is applied to remove low-frequency baseline drift and high-frequency muscle noise.
3. **Notch Filtering:** A 50Hz Notch Filter is used to eliminate powerline interference, a common artifact in clinical ECG recordings.

B. Feature Extraction (Beat Segmentation)

Instead of feeding the entire, long ECG time series (5000 samples) into the model, a representative heartbeat is extracted:

1. **R-Peak Detection:** The **R-peak**, the most prominent point of the QRS complex, is detected, typically on Lead **II**, using a signal differentiation and integration technique to enhance peak visibility.



2. **Beat Segmentation:** Fixed-length segments (600 samples, centered around the R-peak) are extracted for every detected beat.

3. **Averaging:** All segmented beats for a subject are **averaged** to create a single, high-fidelity, representative heartbeat morphology (12 leads x 600 samples). This drastically reduces dimensionality while preserving the key diagnostic morphology (P-QRS-T complex).
4. **Z-Score Normalization:** The final averaged beat is normalized per-lead.

1.2 Tabular Data Preprocessing

The tabular data—composed of clinical and demographic variables—undergoes a structured preprocessing workflow:

1. **Feature Engineering:** New, informative features are calculated from the existing ones:
   - **Body Mass Index (BMI)**
   - **Body Surface Area (BSA)**
2. **Imputation:** An **IterativeImputer** (using a technique like Bayesian Ridge Regression) is fit *only* on the training data to estimate and fill missing values (NaNs) in the numerical features.
3. **Scaling and Encoding:**
   - **Numerical Features** (age, weight, BMI, etc.) are scaled using a **StandardScaler** to have zero mean and unit variance.
   - **Categorical Features** (sex, sport_classification) are converted to numerical format using **OneHotEncoder**.
4. **Output:** The final tabular input is a vector of 9 features.

2. Model Architecture: The Three-Branch CNN

The network is a **Three-Branch Convolutional Neural Network (CNN)** designed for multimodal fusion:

1. **ECG Frontal Branch:** A CNN processes the 6 frontal leads (I, II, III, aVR, aVL, aVF).
2. **ECG Precordial Branch:** A CNN processes the 6 precordial leads (V1-V6).
3. **Tabular Branch:** A small **Multi-Layer Perceptron (MLP)** processes the 9 tabular features.

- **Feature Extraction:** Both ECG branches use identical CNN blocks (Conv1D, ReLU, MaxPool) to independently learn spatial and temporal patterns from the two main physiological views of the heart.
- **Feature Fusion:** The learned features from the three branches are **concatenated** (joined end-to-end) into a single, high-dimensional vector of 4,640 features.
- **Classification:** A final fully connected head (with **Dropout** for regularization and **Sigmoid** activation for binary classification) takes the fused vector and outputs the final probability of the positive class.

3. Evaluation Methodology: K-Fold Cross-Validation

Why K-Fold?

The **Stratified K-Fold Cross-Validation** approach (K=5) was chosen for rigorous evaluation:

- **Robustness:** It provides a much more reliable estimate of the model's true generalization performance than a single train/test split.
- **Full Data Utilization:** Every data point is used exactly once in the validation set and K-1 times in the training set, which is crucial for smaller datasets.
- **Stratification:** Ensures that the proportion of positive and negative classes is preserved in every fold, preventing bias in the evaluation due to class imbalance in one specific split.

4. Analysis of Final Results

The model was trained for **10 epochs** across 5 folds. The final results reveal a model with decent discriminative power but a significant bias caused by **overfitting**.

4.1 Aggregate Metrics Summary

Metric	**Mean (μ)**	**Standard Deviation (σ)**	**Interpretation**
Mean AUC	**0.7093**	0.0821	The model has a fair ability to distinguish between the classes. The high σ shows high variability between folds, indicating the model's sensitivity to the training data.
Mean Accuracy	**0.7243**	0.0639	The model correctly predicts the class for about 72.4% of subjects on average.
Mean Sensitivity	**0.8694**	0.0692	**Excellent** at detecting **True Positives** (i.e., when the subject *should* be classified as Positive, the model is correct 87% of the time).
Mean Specificity	**0.4102**	0.1016	**Poor** at detecting **True Negatives** (i.e., when the subject *should* be classified as Negative, the model is correct only 41% of the time).

4.2 Overfitting and Bias Analysis

The primary issue revealed by the loss curves and the metrics is **overfitting** coupled with a strong **classification bias**:

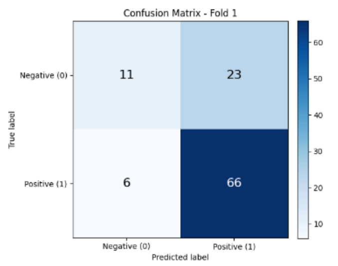
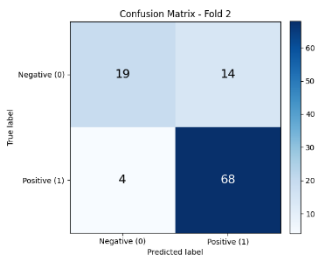
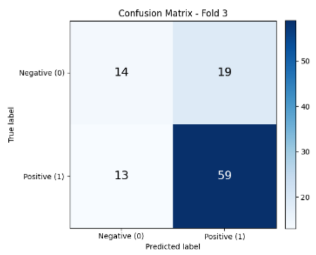
1. **Overfitting Confirmation:** In Folds 3, 4, and 5, the **Training Loss** drops consistently (down to $approx\ 0.3$), while the **Validation Loss** stops decreasing around Epoch 4-5 and begins to rise significantly (up to $approx\ 0.84$). This

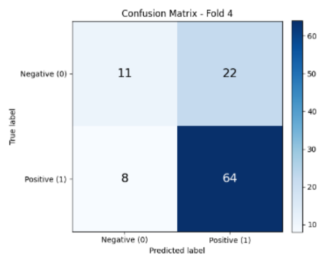
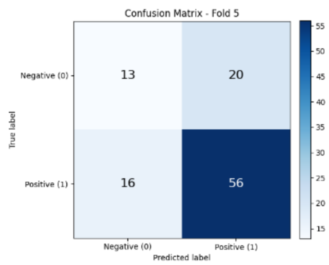
divergence is the hallmark of overfitting. The model has learned the training data noise but fails to generalize.

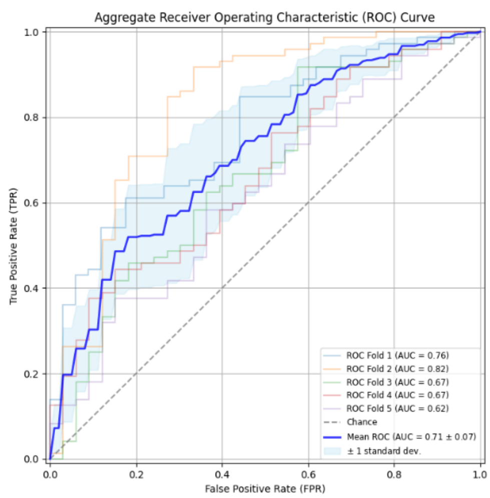
2. **Classification Bias:** The massive gap between **Sensitivity (approx 87%)** and **Specificity (approx 41%)** indicates the model has learned to predominantly predict the **Positive class**. It minimizes false negatives at the expense of creating many **False Positives**. This behavior is typical when the target variable is imbalanced, and the model minimizes the loss by favoring the more frequent class.

4.3 Visual Results

The model results are visualized below, confirming the aggregate metrics and cross-fold variability.

Fold 1 Confusion Matrix	Fold 2 Confusion Matrix	Fold 3 Confusion Matrix
		

Fold 4 Confusion Matrix	Fold 5 Confusion Matrix
	

Aggregate ROC Curve


4.4 Recommendations for Next Steps

To improve generalization and reduce the classification bias, the following steps are recommended:

1. **Early Stopping:** Implement a rule to stop training when the **Validation Loss** fails to decrease for several epochs. This is the most effective way to combat the observed overfitting.
2. **Addressing Imbalance:** Employ a **weighted Binary Cross-Entropy (BCE) loss function** to place a higher penalty on misclassifying the minority class, forcing the model to improve its Specificity.
3. **Regularization Tuning:** Increase the Dropout rate or introduce **L2 regularization (weight decay)** to the optimizer to further penalize complex models.