# **Design Document: 5-Class Cardiac Arrhythmia Classifier Using Linear SVM Ensemble**

## **1. Overview**

This project implements a classifier for detecting five types of cardiac rhythms using ECG data. The approach is based on a **enamble of** **one-vs-all linear SVMs**, trained on **per-beat features** using MATLAB. The models are optimized for fast inference and hardware simplicity, running on an RTX 3060 GPU during training and designed for low-latency deployment. Classification targets the following classes:

* **AFIB** – Atrial Fibrillation
* **BII** – Second-Degree Atrioventricular Block
* **VT** – Ventricular Tachycardia
* **VFL** – Ventricular Flutter
* **Normal** – Normal Sinus Rhythm

**2. System Specifications**

* **Hardware**:
  + GPU: NVIDIA RTX 3060 (12 GB VRAM)
  + CPU: Intel i9-12900k
  + RAM: 64 GB DDR4
* **Software**:
  + MATLAB R2024a
  + WFDB Toolbox for MATLAB
  + Signal Processing Toolbox
  + Statistics and Machine Learning Toolbox

## **3. Dataset and Preprocessing**

* **Format**: WFDB files (.dat, .hea)
* **Preprocessing Steps**:
  + Baseline correction using high-pass filtering
  + Amplitude normalization
  + Peak detection for R, Q, S, T
  + Feature extraction per beat
* **Sampling Frequency**: 360Hz

## **4. Feature Extraction**

Each heartbeat is characterized by six features

**Namely:** R peak, Q peak, S peak, T peak, RR interval and QS interval.

**5. Model Architecture**

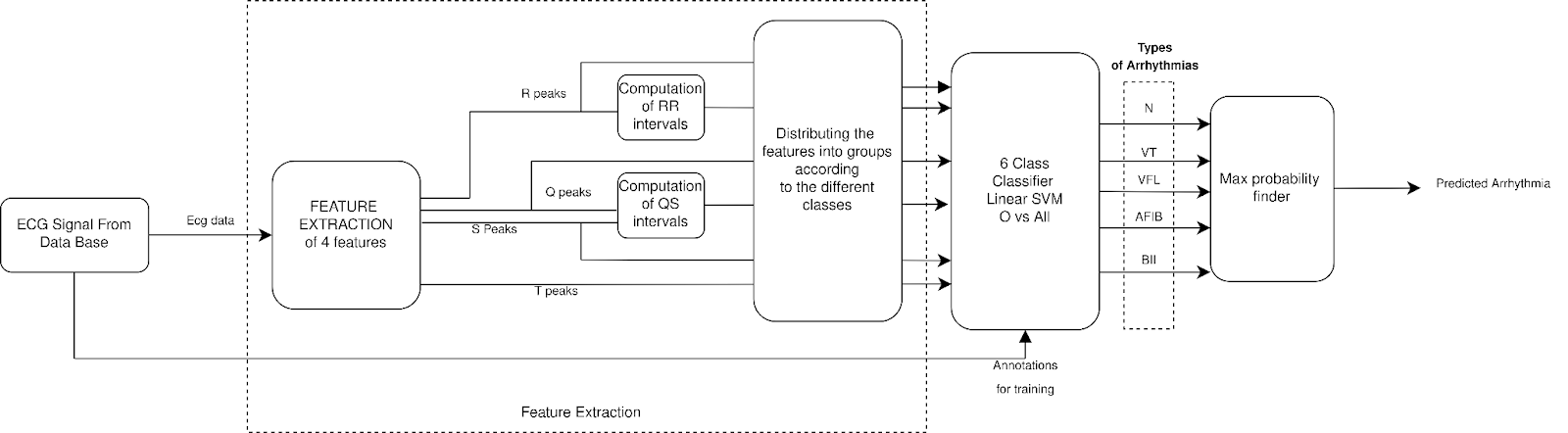
### **5.1 One-vs-All Linear SVMs**

* **Model**: Linear SVM via fitcsvm
* **Design**: One classifier per class:
  + LSVM-AFIB vs Al
  + LSVM-BII vs All
  + LSVM-VT vs All
  + LSVM-VFL vs All
  + LSVM-Normal vs All
* **Rationale**: Linear SVMs are chosen for their simplicity, fast inference, and suitability for hardware implementation.

### **5.2 Ensemble Logic**

* Each LSVM outputs a decision score.
* Final class label is determined by selecting the classifier with the highest score (argmax).
* In case of ambiguous scores, a fallback based on score margin is used.

## **6. Workflow Diagram**



## **7. Training Strategy**

* **Per-Model Dataset**: For each class, training uses only the relevant (positive + negative) samples.
* **Testing Strategy**: All models are tested against the full universal dataset.
* **Training Tool**: fitcsvm in MATLAB with:
  + Linear kernel
  + Standardized features
  + BoxConstraint = 1
* **Evaluation**:
  + 5-fold cross-validation
  + Confusion matrix
  + Accuracy, Precision, Recall, F1-score
  + Visualized using confusion matrix and console prints

## **8. Performance & Optimization**

* **Inference Speed**: Prioritized via:
  + Use of linear SVM (no kernel computations)
  + Compact model representation
* **Storage**: Models saved using saveCompactModel for lightweight deployment.
* **Hardware Compatibility**: Designed for integration into embedded or real-time systems

## **9. Deployment Plan**

* Export trained SVMs using saveCompactModel.
* Integrate ensemble into real-time diagnostic systems via:
  + MATLAB Coder for C/C++ deployment
  + Embedded hardware implementation with minimal overhead
* Consider future integration into FPGA or MCU environments.

## **10. Next Steps**

* Train and validate models for BII, VFL, and Normal classes.
* Complete ensemble integration.