ONNYX : Optimized Neural Networks Yielding eXplainable Insights from ECG Signals-Based Data Streams

Sanket Mishra[2(✉)], V. Aravindan[1], Rajkanwar Singh[1], Hasita Chowdary Meka[1], Sandipan Maiti[1], and Subhrakanta Panda[3]

[1] School of Computer Science and Engineering, VIT-AP University, Amaravati, India
sandipan.maiti@vitap.ac.in
[2] Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, India
sanket.mishra@manipal.edu
[3] Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Hyderabad Campus, Hyderabad, India
spanda@hyderabad.bits-pilani.ac.in

Abstract. Deep learning classification models are extensively utilized for the automated diagnosis of heart disease (HD) by analyzing various physiological signals, such as electrocardiogram (ECG), magnetocardiography (MCG), heart sounds (HS) and impedance cardiography (ICG) signals. In this study, we introduce the ONNYX framework (Optimal Neural Networks Yielding eXplainable insights from ECG signals-based data streams), which demonstrates a big data strategy for ECG classification. This framework incorporates several modules, including FastAPI, MinIO, mlflow, Ray, Kubernetes, and Pulsar. We have developed a high throughput and low latency system using Kubernetes' distributed architecture and Ray's distributed training to classify ECG signals. The ECG records of subjects sourced from the MIT-BIH repository are sampled and input into the classification models to distinguish between normal and abnormal heart rate patterns in patients. We introduce an innovative optimal model selection algorithm that assesses classification techniques according to training efficiency and identifies the most suitable ones for testing. Our weighted ensemble method attained an overall accuracy of 99.27% and 99.16% in binary and multiclass classification settings respectively.

Keywords: ElectroCardiogram signals · Heart Rate Variability · Distributed deep learning · Explainable AI

1 Introduction

Cardiovascular diseases (CVDs) are the leading cause of death worldwide, claiming more than 17 million lives each year[1]. These include coronary artery disease,

[1] https://www.who.int/health-topics/cardiovascular-diseases.

C. K. Leung et al. (Eds.): DaWaK 2025, LNCS 16048, pp. 317–331, 2026.
https://doi.org/10.1007/978-3-032-02215-8_26

rheumatic heart disease, and heart failure, with many premature deaths occurring in individuals under 70 years [1]. Conditions such as arrhythmias and valve disorders disrupt cardiac function, often requiring clinical intervention [2]. Manual interpretation of ECGs is time-consuming and error-prone, particularly in low-resource settings. Deep learning offers a promising alternative, with models increasingly used to detect CVDs from ECG signals [3]. These approaches enhance diagnostic speed and accuracy, facilitating personalized care. Advancements in AI have transformed healthcare, excelling in image analysis tasks such as X-rays, CT, MRI, and mammograms, often assisting medical persons in detecting conditions such as tumors and fractures. In deep learning, we have noticed that convolutional networks [4] have proven successful in chest radiographs and ECG data to identify irregular heart patterns (arrhythmias). The capacity of these approaches to identify symptoms of cardiovascular disease may be further enhanced by including transfer learning [5] and federated learning methodologies [6]. On the other hand, the concept of employing an online deep learning system could introduce a vital element for improving the real-time identification of CVDs, which might be essential for quicker diagnostics.

In this work, we propose ONNYX (Optimized Neural Networks Yielding eXplainable insights from ECG signals), a real-time CVD detection framework leveraging a distributed Kubernetes-based architecture. ONNYX concurrently deploys multiple deep learning models, using a multi-armed bandit strategy with Thompson sampling to dynamically select the best-performing model based on incoming data streams. Our key contributions are:

(a) We introduce a microservice-based AI framework for arrhythmia detection from ECG streams using scalable deep learning deployments.
(b) We implement distributed concurrent model execution, with adaptive model selection via probabilistic Thompson sampling; Apache Pulsar is used to simulate real-time data ingestion.
(c) We enhance model interpretability using SHAP and achieve a 99.27% (binary classification) and 99.16% (multiclass classification) accuracy via probabilistic ensembling of top-performing classifiers.

In the subsequent section, we examine various methodologies employed in the domain of ECG classification that have inspired the present study.

2 Related Works

Traditional ECG interpretation systems struggled with intraclass variability and overreliance on supervised datasets, often performing poorly on unseen data [7]. Deep learning methods, particularly CNNs, have since demonstrated cardiologist-level accuracy in multi-label ECG classification, outperforming commercial systems when trained on large datasets [8]. Architectures like 1D ResNet extract features directly from heartbeat sequences, improving diagnostic capabil-

ity [9]. However, challenges remain in model explainability and bias mitigation, prompting the use of techniques like LIME to highlight important ECG segments [10].

To address class imbalance and enhance generalization, hybrid methods using Bi-LSTM with GANs and SMOTE have been explored, though most studies remain retrospective, lacking scalable, real-time implementations [9]. Big data analytics have enabled ECG-derived predictions of structural abnormalities such as left ventricular mass, yet real-time inference remains underexplored [11]. Variability in ECG formats and annotation standards also impairs model interoperability [12].

Efforts to integrate Complex Event Processing (CEP) with LSTM models show promise for streaming prediction but face scalability issues under fluctuating data loads [13]. While frameworks using Apache Spark and Flink support high-throughput analytics, they either rely on micro-batching or lack native deep learning support, complicating deployment in clinical settings [14]. Clinical decision support systems increasingly incorporate explainable AI, yet face similar integration and scalability challenges when applied to real-time cardiovascular monitoring.

In the next section, we elaborate on the different microservices-based modules that were considered in the ONNYX framework.

3 ONNYX Framework

Figure 1 illustrates the open source and modular containerized framework proposed for the classification of ECG data streams and the early detection of arrhythmias.

To integrate the underlying principles of the algorithms utilized in the proposed ECG classification framework, a concise description and analysis of the performance effects of each module are provided in Table 1.

3.1 Data Ingestion and Preprocessing

Training data is ingested via a FAST API[2] endpoint and stored in MinIO[3], a high-performance, S3-compatible object store used as a data lake. This schema-on-read approach supports scalable, cloud-native storage and efficient handling of large, unstructured deep learning datasets. MinIO enables parallel read/write operations and outperforms traditional databases for such workloads.

The Ray[4] job first ingests this data into Ray Data and applies filtering functions to enhance signal quality:

1. Bandpass Filter: We use a bandpass filter (0.5âĂŞ50 Hz) to remove low and high-frequency noise, like that from muscle contractions or electrical equipment, improving signal clarity.

[2] https://fastapi.tiangolo.com/.
[3] https://min.io/.
[4] https://www.ray.io/.

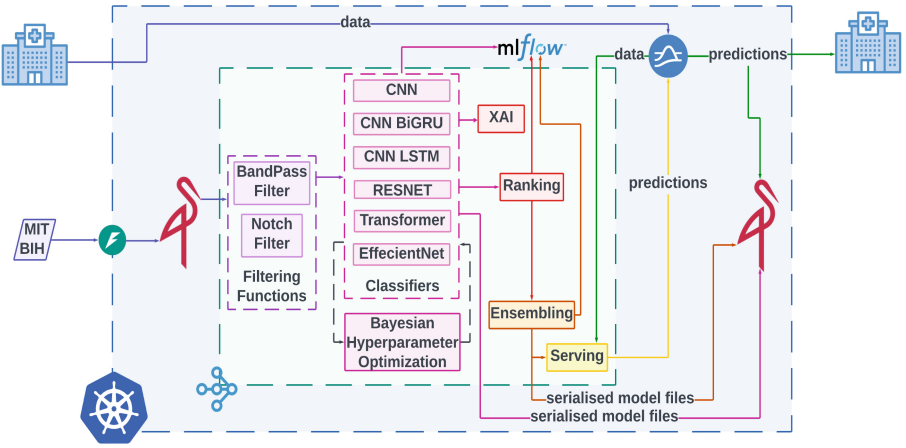


Fig. 1. ONNYX Framework

Table 1. Summary of ONNYX Architecture

Component	Technology	Details
Streaming	Apache Pulsar	**Rationale:** Utilizes a robust event streaming platform for managing real-time data flows **Performance Boost:** Delivers sub-second latency and high throughput for continuous data ingestion
Distributed Computing	Ray	**Rationale:** Leverages a distributed framework to enable concurrent task execution **Performance Boost:** Dynamically scales compute resources to reduce overall processing time
Preprocessing	Ray Data	**Rationale:** Employs distributed data handling to efficiently transform and augment datasets **Performance Boost:** Minimizes I/O and computation bottlenecks to enhance preprocessing throughput
Model Training	Ray Train, PyTorch, TensorFlow	**Rationale:** Integrates scalable deep learning libraries for training on varied hardware **Performance Boost:** Accelerates training cycles by distributing workloads across multiple nodes
HPO (Hyperparameter Optimization)	Ray Tune	**Rationale:** Uses advanced search algorithms to systematically optimize hyperparameters **Performance Boost:** Cuts down tuning iterations, conserving computational resources
Model Serving	Ray Serve	**Rationale:** Provides a modular framework for deploying production-grade ML models **Performance Boost:** Supports autoscaling and asynchronous processing for real-time inference
Explainability (XAI)	SHAP	**Rationale:** Integrates interpretability tools to elucidate model predictions and decisions **Performance Boost:** Enhances transparency and compliance by quantifying feature contributions
Orchestration	Kubernetes	**Rationale:** Automates container orchestration for streamlined service deployment **Performance Boost:** Boosts system resilience and uptime with dynamic scaling and self-healing capabilities
Model Tracking	MLflow	**Rationale:** Enables systematic logging and tracking of experiments and metadata **Performance Boost:** Facilitates reproducibility and comparative analysis of model versions
Storage	MinIO	**Rationale:** Employs a high-performance object storage system compatible with S3 APIs **Performance Boost:** Supports scalable, schema-on-read storage for efficient data access in ML workloads

2. Notch Filter: Notch filter removes narrow frequency interference, specifically powerline noise (60 Hz in the US, 50 Hz elsewhere).

Ray Data facilitates efficient I/O and preprocessing operations that scale across multiple nodes, overcoming data bottlenecks that typically arise when using in-memory single-machine compute via Pandas[5].

3.2 Model Training

Our system uses Ray Train to parallel-train multiple deep learning models (built with PyTorch[6]) across distributed CPUs and a dedicated GPU. This accelerates training and optimizes resource use by dynamically scaling tasks across clusters. We elaborate on the various classifiers, their respective architecture and the rationale behind using them in Table 2.

Table 2. Architectural Summary of Deep Learning Models for ECG Classification

Model	Architecture	Rationale
Simple1DCNN [15]	Stacked 1D Convolutional layers → Max Pooling → Fully Connected layer	Automatically extracts spatial hierarchies from raw signals, excelling in the detection of ECG arrhythmias *Generalizes well across signal qualities, robust to noise*
CNN-BiGRU	Convolutional layers → Max Pooling → Bidirectional GRU layer → Fully Connected layer	Combines spatial and temporal modeling. CNN detects morphology, BiGRU captures bidirectional dependencies *Improves sequence awareness in arrhythmia classification*
CNN-LSTM [16]	Convolutional layers → Max Pooling → Bidirectional LSTM layer → Fully Connected layer	Extracts spatial features via CNN and models temporal dependencies via LSTM *Effective for long ECG windows where rhythm and morphology matter*
EfficientNet1D [17]	Scaled 1D Convolutional blocks → Batch Normalization and Activation → Adaptive Average Pooling → Fully Connected layer	Balances depth, width, and resolution for optimal performance with fewer parameters *Efficient and accurate for real-time ECG applications*
ResNet1D [18]	Initial Convolution → Residual Blocks with skip connections → Deep stacking → Fully Connected layer	Residual connections prevent vanishing gradients *Captures fine-grained ECG variations while training deeper networks*
Transformer1D [19]	Initial Convolution → Transformer Encoder blocks → Multi-head Attention + Feed-forward layers → Fully Connected layer	Uses self-attention to capture global dependencies *Ideal for modeling long ECG sequences without recurrence*

[5] https://pandas.pydata.org/.
[6] https://pytorch.org/.

Hyperparameter Optimization. Ray Tune[7] uses Bayesian Hyperparameter Optimization for parallel execution to quickly find optimal classifier hyperparameters. It is more efficient than grid or random search since it uses a probabilistic model to explore promising options and prunes underperforming models early, saving computational resources.

Explainability (XAI). Integrating XAI is crucial for us when working with models trained on the MIT-BIH arrhythmia dataset, given the critical nature of medical AI. SHapley Additive exPlanations (SHAP) significantly enhances our model's interpretability, which is vital for achieving clinical acceptance and building trust. It helps us identify if our models are learning incorrect patterns (for example, from irrelevant ECG features) and also aids in amending any biases. SHAP also provides explanations for each patient, offering clinicians valuable insight into the rationale behind our model's specific ECG classifications.

3.3 Model Ranking

Instead of static weighting, we rank classifiers using a multi-armed bandit approach with Thompson Sampling. This allows us to dynamically balance exploring new ranking possibilities with exploiting previously successful ones, ensuring we always select the optimal classifier for predictions.

Multi-bandits Algorithm for Model Ranking. The algorithm begins with **Initialization** (line 2), setting success and failure counters $s_i = 0$ and $f_i = 0$ for each classifier i.

In the **Main Loop** (lines 3–14), repeated for $t = 1$ to T, it first performs **Sampling** (line 5) by drawing $\theta_i \sim \text{Beta}(s_i + 1, f_i + 1)$, modeling uncertainty using the Beta-binomial framework.

Then, it executes **Selection** (line 7), choosing $j = \arg\max_i \theta_i$, balancing exploration and exploitation.

The chosen classifier yields a binary **Reward** $r \sim \text{Binomial}(1, p_j)$ (line 8), and the algorithm **Updates Counters** (lines 9–12): incrementing s_j if $r = 1$, otherwise f_j.

Finally, after T rounds, the estimated success probability for each classifier is computed as

$$\hat{p}_i = \frac{s_i}{s_i + f_i + \epsilon}$$

with ϵ preventing division by zero. Classifiers are then ranked by $\hat{p}_i$ (line 15).

Each step systematically integrates new information, ensuring an optimal selection process through a balance of exploration and exploitation.

The rationale behind ranking models using Thompson sampling than just using metrics like F1 Score or Accuracy is that they are a point estimate and inform us how well a model did on a fixed test set, ignoring variance and lacking dynamic adjustment to account for possible data distributions drift.

[7] https://docs.ray.io/en/latest/tune/index.html.

Algorithm 1. Thompson Sampling for Classifier Selection

Input: Number of classifiers n, number of rounds T, true reward probabilities $\{p_1, p_2, \ldots, p_n\}$, classifier names.
Initialize: For each classifier $i = 1, \ldots, n$, set

$$s_i \leftarrow 0 \quad \text{(successes)} \quad \text{and} \quad f_i \leftarrow 0 \quad \text{(failures)}.$$

for $t = 1$ **to** T **do**
  for $i = 1$ **to** n **do**
    Sample $\theta_i \sim \text{Beta}(s_i + 1, f_i + 1)$.
  end for
  Select classifier $j = \arg\max_i \theta_i$.
  Obtain reward $r \sim \text{Binomial}(1, p_j)$.
  if $r = 1$ **then**
    $s_j \leftarrow s_j + 1$.
  else
    $f_j \leftarrow f_j + 1$.
  end if
end for
Output: For each classifier i, compute the posterior mean:

$$\hat{p}_i = \frac{s_i}{s_i + f_i + \epsilon},$$

with a small $\epsilon > 0$ to avoid division by zero. Sort classifiers based on $\hat{p}_i$ in descending order.

3.4 Model Ensembling

In our approach, we selected the top $n = 3$ models from the Thompson Sampling Multi-Armed Bandit ranking, as supported by the ablation study in Table 5. To ensure numerical stability during weight computation, we introduced a small constant $\epsilon = 1 \times 10^{-6}$. The weight for each model was calculated using:

$$w = \frac{1}{(1 - \text{Accuracy}) + \epsilon}$$

This formulation assigns greater importance to more accurate models. We obtained class probabilities from each selected model on the test set X_{test}, and combined them using a weighted average. Final binary predictions were derived by applying a 0.5 threshold to the aggregated probabilities.

The PWPAE framework inspired our method [20], which emphasises dynamic model selection and performance-based weighting. By prioritising models with lower error rates, we enhanced the robustness and generalizability of the ensemble. We evaluated its performance using accuracy, precision, recall, and F1 score.

The rationale to use this ensembling technique over top-ranked classifier or a non weighted ensemble is that it gives higher weight to better models while still retaining contributions from others, thereby mitigating overconfidence and reducing the impact of model-specific bias and overfitting tendencies.

3.5 Model Management and Experiment Tracking

MLflow[8] handles model logging, tracking performance metrics, and managing serialized model files. It also allows data scientists to perform comparisons with various visualization tools.

3.6 Model Serving

ONNYX serves models via Ray Serve over an HTTP endpoint to enable low-latency, autoscalable inference as a Prediction-as-a-Service framework. To manage burst traffic and cold-start delays, Apache Pulsar[9] is used as a message queue between the client and the model endpoint. Predictions are published back to the client and forwarded to the data lake via Pulsar.

Unlike Apache Kafka, which uses polling and disk-based storage that can increase latency and overwhelm downstream services, Pulsar offers native queueing, in-memory caching, and multitopic routing—enabling controlled, efficient, and parallel data flow to clients and storage.

3.7 Kubernetes

Kubernetes[10] orchestrates ONNYX's distributed model services, offering scalability, reliability, and cloud-agnostic deployment across local or cloud infrastructures (AWS, GCP, Azure). It ensures dynamic load balancing, autoscaling, and fault tolerance. For local development, we used Minikube[11] with the Docker[12] driver, ensuring consistent testing across environments.

4 Experimental Results and Discussion

In this section, we examine the various experimental results obtained during the ECG classification process.

4.1 MIT-BIH Arrhythmia Dataset

The MIT-BIH Arrhythmia Database [21], a widely recognized resource for ECG analysis, contains 48 half-hour, two-channel ambulatory ECG recordings collected from 47 subjects between 1975 and 1979. The researchers digitized these recordings at 360 samples per second with 11-bit resolution, selecting a mix of random excerpts and clinically significant cases of arrhythmia.

In our study, we adopted a 70:30 intra-patient train-test split strategy, where ECG beats from the same patient may appear in both training and testing sets. This is consistent with several prior works that benchmark ECG arrhythmia classification performance on the MIT-BIH dataset using similar splitting strategies [3,4].

[8] https://mlflow.org/.
[9] https://pulsar.apache.org/.
[10] https://kubernetes.io/.
[11] https://minikube.sigs.k8s.io/docs/.
[12] https://www.docker.com/.

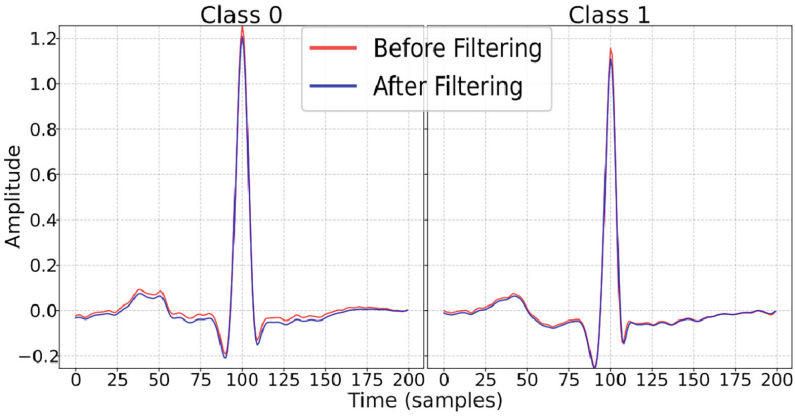


Fig. 2. Signal Before and After Filtering

Figure 2 demonstrates the impact of signal processing using a bandpass and notch filter on each class within the dataset. This filtering procedure enhances the signal by mitigating baseline drift, attenuating high-frequency noise, and diminishing power line interference, while maintaining critical electrocardiogram characteristics.

Table 3. Thompson Ranked Classifier Metrics on Binary Classification

Rank	Name	Estimated Reward	Accuracy	Precision	Recall	F1 Score	Hyperparameters
1	ResNet	0.9927	0.9920	0.9689	0.9558	0.9623	lr = 0.0016
2	CNN-LSTM	0.9821	0.9810	0.9515	0.8673	0.9075	conv = 8.4945, lstm = 25.5679, lr = 0.0095
3	EfficientNet	0.9779	0.9816	0.9623	0.8618	0.9093	lr = 0.0038
4	Transformer	0.9730	0.9766	0.9315	0.8440	0.8856	lr = 0.0007
5	CNN-BiGRU	0.7778	0.9823	0.9427	0.8888	0.9150	conv = 8.4945, gru = 30.8171, lr = 0.0073
6	1D-CNN	0.5000	0.9725	0.9522	0.7832	0.8595	conv = 9.1833, lr = 0.0030

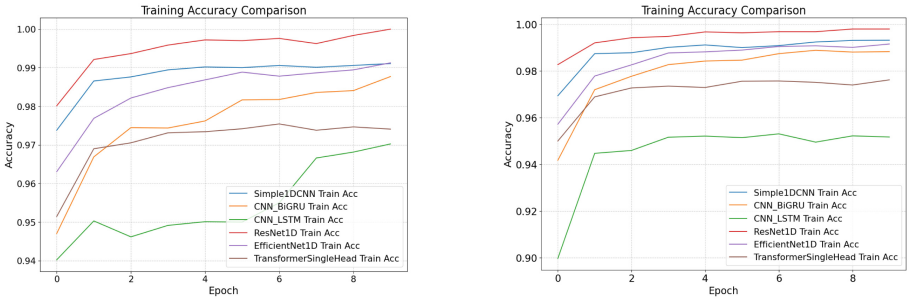
Table 3 illustrates the results of various models along with their respective rankings. Our analysis covers six deep learning methodologies, executed concurrently within the distributed ONNYX framework. We observed that ResNet requires a substantially longer duration for training in comparison to other models, whereas the 1D-CNN model demonstrates the shortest training time. Additionally, we recognize that sophisticated models like Transformers also require considerable training time. The implemented ranking method positions ResNet, CNN-LSTM, and EfficientNet as the top three models. This ranking strategy facilitates the identification of models that may be integrated with probabilities to advance in the ensemble process.

Table 4. Thompson Ranked Classifier Metrics on Multiclass Classification

Rank	Name	Estimated Reward	Accuracy	Precision	Recall	Hyperparameters
1	ResNet	0.9914	0.9911	0.9911	0.9911	lr = 0.001644585
2	Transformer	0.9869	0.9868	0.9864	0.9868	lr = 0.000675028
3	1D-CNN	0.9863	0.9804	0.9795	0.9804	conv = 9.1833, lr = 0.002983168
4	EfficientNet	0.9757	0.9838	0.9831	0.9838	lr = 0.003807947
5	CNN-LSTM	0.9388	0.9326	0.8986	0.9326	conv = 8.4945, lstm = 25.5679, lr = 0.009512072
6	CNN-BiGRU	0.9369	0.9115	0.8768	0.9115	conv = 8.4945, gru = 30.8171, lr = = 0.00734674

To demonstrate the framework's capability, we also perform multiclass classification, and the respective metrics are depicted in Table 4. While we do see a shift in the rankings of the classifiers, given the dynamic nature of the pipeline, the best-performing ensemble will ultimately be put into production. We have presented upcoming results on binary and multiclass classification to depict the capability of ONNYX in detecting arrhythmia on ECG data streams.

Figure 3 illustrates the accuracy of deep learning models, demonstrating marked improvements in classification performance in various epochs for both Binary and Multi Class classifiers (Table 6).



(a) Binaryclass Classifiers (b) Multiclass Classifiers

Fig. 3. Change in Accuracy Epoch in Binary and Multiclass Classifiers

Table 5. Thompson Ranked Weighted Ensemble Binary Classifier Metrics

Ensemble Size	Models	Model Weights	Accuracy	Precision	Recall	F1-Score
2	ResNet,EfficientNet	0.7096,0.2904	0.9925	0.9761	**0.9533**	0.9646
3	**ResNet,EfficientNet,CNN-LSTM**	0.5713,0.2338,0.1950	**0.9927**	**0.9804**	0.9509	**0.9654**
4	ResNet,EfficientNet,CNN-LSTM,1DCNN	0.4868,0.1992,0.1661,0.1479	0.9923	0.9785	0.9490	0.9635

Table 6. Thompson Ranked Weighted Ensemble Multiclass Classifier Metrics

Ensemble Size	Models	Model Weights (Normalized)	Accuracy	Precision	Recall	F1-Score
2	**ResNet, Transformer**	0.5983, 0.4017	**0.9916**	**0.9915**	**0.9916**	**0.9915**
3	ResNet, Transformer, 1D-CNN	0.4707, 0.3161, 0.2132	0.9909	0.9907	0.9909	0.9907
4	ResNet, Transformer, 1D-CNN, EfficientNet	0.3741, 0.2512, 0.1694, 0.2053	0.9912	0.9910	0.9912	0.9910

Table 5 shows the results of the different ensembles created from the top-ranked models identified in Table 3 for binary classification. We conducted an ablation analysis using two, three and four models to determine the optimal ensemble combination that may improve classification performance. In addition, custom weights were formulated and integrated into each model to facilitate improved convergence. We observed that the ensemble comprising three methodologies, namely ResNet, EfficientNet, and CNN-LSTM, demonstrated superior performance across all metrics, achieving an exceptional accuracy of 99.27%. Performance variations among the various ensembles fluctuated within a margin of 0.01% to 0.03%, indicating enhanced performance through the majority voting ensembles. In addition, Table 6 presents the same analysis for a multiclass scenario. In this table, we observe that the ensemble size of 2 outperforms all other ensembles and thus is put to production.

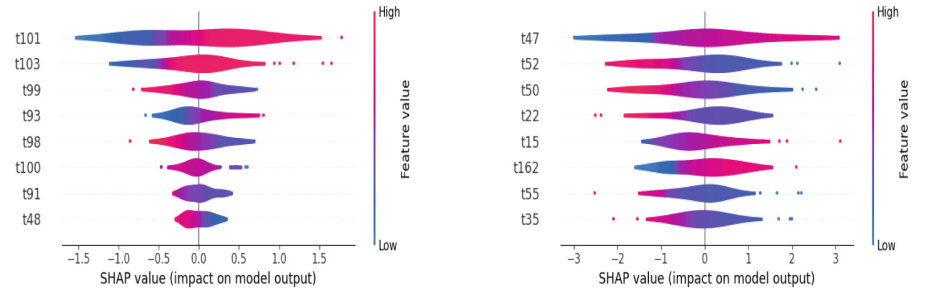
Table 7. Inference Metrics for Top 3 Binary Ensemble Classifier

Signals	Total Time (s)	Throughput (SPS)	Avg Latency (ms)	Min Latency (ms)	Max Latency (ms)	95% Latency (ms)
50	1.5083	33.1490	30.1626	14.3369	60.9237	58.9972
100	3.9701	25.1880	39.6973	13.7745	65.6122	65.3230
200	7.7110	25.9371	38.5503	13.8197	73.2426	65.5490
500	19.8225	25.2239	39.6403	13.9686	72.7405	65.2281
1000	40.3844	24.7621	40.3795	14.1674	67.4932	65.5553

Table 7 presents the inference metrics for the classifier of the highest performing binary ensemble, while being served on Ray, evaluated using increasing request sizes from 50 to 1000. We observe that the average latency ranges from 30.16 ms (50 requests) to 40.38 ms (1000 requests). At the 95th percentile latency also remains consistent at 65 ms. The maximum latency increases slightly with load, peaking at 73 ms, but does not indicate exponential degradation. Throughput (signals per second, SPS) falls slightly as the number of requests increases from 33.15 SPS, at 50 requests to 24.76 SPS, at 1000 requests. Similarly Table 8 presents the top-performing multiclass ensemble classifier. We observe that the classifier maintains stable performance under increasing load, with average latency remaining around 32–35 ms and 95th percentile latency consistently below 57 ms, supporting real-time inferencing. Compared to the binary classifier (Table 7), the multiclass classifier (Table 8) exhibits slightly lower 95th percentile latency and more stable throughput under load.

Table 8. Inference Metrics for Top 2 Multiclass Ensemble Classifier

Signals	Total Time (s)	Throughput (SPS)	Avg Latency (ms)	Min Latency (ms)	Max Latency (ms)	95% Latency (ms)
50	1.7333	28.8462	34.6614	10.3165	57.7038	57.4033
100	3.2563	30.7096	32.5583	10.2194	58.3754	57.2018
200	6.7294	29.7203	33.6428	10.5564	57.7880	56.3047
500	15.9278	31.3917	31.8512	9.8152	58.0600	56.3062
1000	34.3374	29.1228	34.3327	9.6526	61.5697	56.5906
2000	69.6609	28.7105	34.8254	9.5381	59.2717	56.9662



(a) Multiclass Classifier (b) Binaryclass Classifier

Fig. 4. XAI Plot for RESNET

Figure 4a and Fig. 4b are violin summary plots for RESNET which offers compact representation of the distribution and variability of SHAP values for each feature. The plots above illustrate the top 8 features. Shap values on the X-axis correspond to the importance of the element in the prediction of the model. From Fig. 4a, we learn that on the multi-class model, features t101, t103, and t99 dominate the influence of the model. While from Fig. 4b we observe that on the binary class model, features t47, t52, t50 dominate the influence of the model. The binary model shows a wider shap value spread, indicating stronger feature influence on predictions, while the multi class model showcases a narrow spread, indicating lower variability in feature influence across different predictions. The contrasting SHAP distributions suggest that the binary model relies more heavily on specific features for decision-making, while the multi-class model distributes importance more uniformly across features.

We compare ONNYX to existing benchmarks in the field of ECG classification in Table 9.

Table 9. Performance Comparison of ONNYX with Benchmark Models

Study/Model	Accuracy	Type	Remarks
ONNYX	99.27%	Binary & Multiclass	**Approach:** Ensemble of DL models with adaptive selection (Thompson sampling) **Key Features:** Real-time Kubernetes deployment, SHAP explainability, Apache Pulsar **Limitations:** High computational complexity
EBM [22]	96.84%	Binary	**Approach:** Explainable Boosting Machine **Key Features:** Uses a lightweight, interpretable EBM on resource-constrained edge devices **Limitations:** Lower accuracy, relying on handcrafted feature extraction
CNN-LSTM-SE [23]	98.5%	Multiclass	**Approach:** CNN + LSTM + SE attention **Key Features:** Uses ensemble empirical mode decomposition (EEMD) preprocessing; SE attention over channels **Limitations:** Lower accuracy, slower LSTM, sensitive to imbalance
1D ResNet [24]	98.63%	Multiclass	**Approach:** Residual 1D CNN with SMOTE **Key Features:** 6 conv layers + 3 pooling; high specificity (99.06%) **Limitations:** Lower sensitivity (92.4%), fixed architecture
MB-MHA-TCN [25]	98.75%	Multiclass	**Approach:** Multi-branch Temporal Convolutional Networks + Multi-head Attention **Key Features:** Parallel dilated branches, focal loss, Bayesian tuning **Limitations:** High memory/computation

5 Limitations and Future Work

This study introduces a modular and reproducible framework intended for the binary classification of ECG signals. We recognize the potential for integrating this framework within a real-time system, thereby simulating its applicability in real-world environments. However, several limitations or challenges encountered during deployment are elaborated on in the following points:

- ONNYX has been deployed on Kubernetes using the Docker driver for local development for distributed setup. We noted that the platform extensively uses RAM for storing the intermediate results which claims much of memory resources. This does not affect the predictions as the training and testing times are maintained well.
- Adapting the system to handle heterogeneous data sources will need writing of custom pre-processing jobs.

Our future initiatives include the integration of tools for scheduling and event-based job orchestration, as well as the enhancement of observability tools to improve system administration. Moreover, we intend to incorporate components that define our framework into an Infrastructure as Code (IaC) tool, thereby rendering deployment processes more adaptable to future advancements.

While ONNYX is designed to process diverse ECG datasets, as noted in the limitations, modifications to preprocessing jobs might be necessary depending on the structure of incoming data. By modifying ONNYX to avoid storing intermediate results extensively in RAM, we can accommodate even larger volumes of data in memory-constrained setups, leading to a more cost-effective solution. For end-to-end integration with a hospital in a real-world setup, since the model is offered via an HTTP API endpoint, a hospital would just need to provide a way for the system to access their data and get results. The modularity offered by ONNYX also facilitates the setup of a custom frontend.

6 Conclusion

We presented ONNYX, a big data analytics framework for the classification of ECG data streams with an open source modular architecture. Our framework used deep learning to detect heart arrhythmias and allows components to be interchanged for various ECG use cases. We integrate Thompson Sampling with probabilistic weights to find optimal methods. Using a ranking strategy, we achieved 99.27% accuracy in binary and 99.16% accuracy in multiclass with a majority voting ensemble of top-ranked approaches selected by the multibandit approach. We presented architectural details, hyperparameters, and comparisons with state-of-the-art models, including multiclass performance. ONNYX supports adaptive ensembling, enabling model combinations to evolve dynamically during real-world deployment, with the architecture accommodating these changes. XAI with Shapley plots provides insight into the model's predictions.

References

1. Li, Z., et al.: Comparative analysis of atherosclerotic cardiovascular disease burden between ages 20–54 and over 55 years: insights from the global burden of disease study 2019. BMC Med. **22**(1), 303 (2024)
2. Rabadia, J.P., Thite, V.S., Desai, B.K., Bera, R.G., Patel, S.: Cardiovascular system, its functions and disorders. In: Pullaiah, T., Ojha, S. (eds.) Cardioprotective Plants, pp. 1–34. Springer, Singapore (2024). https://doi.org/10.1007/978-981-97-4627-9_1
3. Shoughi, A., Dowlatshahi, M.B., Amiri, A., Kuchaki Rafsanjani, M., Batth, R.S.: Automatic ECG classification using discrete wavelet transform and one-dimensional convolutional neural network. Computing **106**(4), 1227–1248 (2024)
4. Prem Narayan Singh and Rajendra Prasad Mahapatra: A novel deep learning approach for arrhythmia prediction on ECG classification using recurrent CNN with GWO. Int. J. Inf. Technol. **16**(1), 577–585 (2024)
5. Peimankar, A., Ebrahimi, A., Wiil, U.K.: xECG-beats: an explainable deep transfer learning approach for ECG-based heartbeat classification. Netw. Model. Anal. Health Inform. Bioinform. **13**(1), 45 (2024)
6. Ying, Z., Zhang, G., Pan, Z., Chu, C., Liu, X.: FedECG: a federated semi-supervised learning framework for electrocardiogram abnormalities prediction. J. King Saud Univ. Comput. Inf. Sci. **35**(6), 101568 (2023)
7. Zhang, X., et al.: Automated detection of cardiovascular disease by electrocardiogram signal analysis: a deep learning system. Cardiovas. Diagn. Ther. **10**(2), 22735–22235 (2020)
8. Weston Hughes, J., et al.: Performance of a convolutional neural network and explainability technique for 12-lead electrocardiogram interpretation. JAMA Cardiol. **6**(11), 1285–1295 (2021)
9. Khan, F., Yu, X., Yuan, Z., Rehman, A.: ECG classification using 1-d convolutional deep residual neural network. PLOS ONE **18**(4), e0284791 (2023)
10. Chung, C.T., et al.: Clinical significance, challenges and limitations in using artificial intelligence for electrocardiography-based diagnosis. Int. J. Arrhythmia **23**(1), 24 (2022)

11. Tison, G.H., Zhang, J., Delling, F.N., Deo, R.C.: Automated and interpretable patient ECG profiles for disease detection, tracking, and discovery. Circ. Cardiovas. Qual. Outcomes, **12**(9) (2019)
12. Somani, S., et al.: Deep learning and the electrocardiogram: review of the current state-of-the-art. EP Europace **23**(8), 1179–1191 (2021)
13. Mishra, S., Jain, M., Siva Naga Sasank, B., Hota, C.: An ingestion based analytics framework for complex event processing engine in internet of things. In: Mondal, A., Gupta, H., Srivastava, J., Reddy, P.K., Somayajulu, D.V.L.N. (eds.) BDA 2018. LNCS, vol. 11297, pp. 266–281. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04780-1_18
14. Mishra, S., Hota, C.; A rest framework on IoT streams using apache spark for smart cities. In: 2019 IEEE 16th India Council International Conference (INDICON), pp. 1–4 (2019)
15. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
16. Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.C.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting. Adv. Neural Inf. Proces. Syst. **28** (2015)
17. Tan, M., Le, Q.: EfficientNet: rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R., (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97 pp. 6105–6114. PMLR (2019)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
19. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., (2017)
20. Yang, L., Manias, D.M., Shami, A.: PWPAE: an ensemble framework for concept drift adaptation in IoT data streams. In: 2021 IEEE Global Communications Conference (globecom), pp. 01–06. IEEE (2021)
21. Goldberger, A.L., et al.: The MIT-BIH long term database (1992)
22. Xiaolin, L., Qingyuan, W., Panicker, R.C., Cardiff, B., John, D.: Binary ECG classification using explainable boosting machines for IoT edge devices. In: 2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp. 1–4 (2022)
23. Sun, A., Hong, W., Li, J., Mao, J.: An arrhythmia classification model based on a CNN-LSTM-se algorithm. Sensors **24**(19), 6306 (2024)
24. Cretu, I., Tindale, A., Abbod, M., Khir, A., Balachandran, W., Meng, H.: Reliable multimodal heartbeat classification using deep neural networks (2023)
25. Bi, S., Rongjian, L., Qiang, X., Zhang, P.: Accurate arrhythmia classification with multi-branch, multi-head attention temporal convolutional networks. Sensors **24**(24), 8124 (2024)