universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Algorithms for Automatic and Robust Interval Detection in Endurance Training Sessions

verfasst von | submitted by

Katharina Böhm

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2026

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 977
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Business Analytics
Betreut von | Supervisor:	Univ.-Prof. Dr. Nikolaus Hautsch

Acknowledgements

I would like to thank Dr.-Ing. Christoph Thiem, MSc, for his guidance and constructive feedback throughout this thesis. I also thank Univ.-Prof. Dr. Nikolaus Hautsch for his supervision.

Abstract

Accurate segmentation of interval training sessions is a prerequisite for meaningful automated analysis of athletic performance data. Despite its practical relevance, this problem has received little academic attention. While power output provides a direct measure of exercise intensity, heart rate remains the most widely available data in wearable devices, motivating the question of whether heart rate data alone is sufficient for reliable interval boundary detection. This thesis investigates this question by developing and comparing six machine learning architectures – XGBoost, a feedforward neural network, a convolutional neural network, a temporal convolutional network, a bidirectional long short-term memory network, and a 1D U-Net – evaluated on a dataset of 93 manually annotated rowing and cycling sessions. The best models demonstrate that meaningful boundary detection from heart rate data is feasible, with performance depending heavily on extensive feature engineering. However, results vary considerably across sport type and session complexity, and no single architecture dominates universally, likely reflecting the limited amount of labelled training data available for this task. These results clearly show that performance is not yet sufficient to justify fully automatic deployment without human oversight, motivating the design of a User-in-the-Loop workflow in which model predictions serve as reviewable suggestions that substantially reduce manual annotation effort.

Kurzfassung

Eine präzise Segmentierung von Intervalltrainingseinheiten stellt eine zentrale Voraussetzung für die sinnvolle automatisierte Analyse von Leistungsdaten im Sport dar. Trotz der hohen praktischen Relevanz wurde dieses Thema bislang nur begrenzt wissenschaftlich untersucht. Während die erbrachte Leistung ein direktes Maß für die Trainingsintensität darstellt, sind Herzfrequenzdaten in Wearables am weitesten verbreitet, was die Frage aufwirft, ob diese allein für eine zuverlässige Erkennung von Intervallgrenzen ausreichen. Zur Untersuchung dieser Fragestellung werden sechs Modelle miteinander verglichen: XGBoost, ein Feedforward Neural Network, ein Convolutional Neural Network, ein Temporal Convolutional Network, ein bidirektionales Long Short-Term Memory Netzwerk sowie ein 1D U-Net. Die Evaluation erfolgt auf einem Datensatz von 93 manuell annotierten Ruder- und Radfahreinheiten. Die besten Modelle zeigen, dass eine sinnvolle Erkennung von Intervallgrenzen aus Herzfrequenzdaten grundsätzlich möglich ist, wobei die Leistungsfähigkeit der Modelle stark von umfangreichem Feature Engineering abhängt. Die Ergebnisse variieren deutlich je nach Sportart und Komplexität der Trainingseinheiten, ohne dass eine einzelne Architektur durchgängig überlegen ist. Dies ist vermutlich auf die begrenzte Menge an verfügbaren annotierten Trainingsdaten zurückzuführen. Insgesamt zeigt sich, dass die derzeitige Leistungsfähigkeit der Modelle noch nicht ausreicht, um einen vollautomatischen Einsatz ohne menschliche Kontrolle zu rechtfertigen. Dies motiviert das Design eines User-in-the-Loop-Workflows, in dem Modellvorhersagen als überprüfbare Vorschläge dienen und den manuellen Annotationsaufwand erheblich reduzieren.

Contents

List of Tables

List of Figures

List of Algorithms

1. Motivation

In recent years, there has been a growing interest in applying intelligent data analysis to optimize athletic performance. With rapid advancements in technology, artificial intelligence, and wearable devices, athletes and coaches increasingly seek to enhance training through data-driven decision-making rather than relying solely on intuition ([RRKF25], [RF20]). Intelligent data analysis refers to the application of sophisticated computational and statistical methods - including machine learning and pattern recognition - to uncover hidden patterns and generate valuable insights from complex datasets. In other words, the aim is to gain more information than just the obvious from data summarization ([Han03]). Within this context, Smart Sport Technologies (SST) encompass diverse training approaches that integrate wearables, sensors, and Internet of Things (IoT) systems with intelligent analytical techniques to improve training efficiency and overall outcomes ([RF20]).

1.1. AIROW Project

The AIROW (Artificial Intelligence in Rowing) project represents a collaborative initiative at the University of Vienna, bringing together the Research Network Data Science (Data Science @ Uni Vienna), the Department of Sport and Human Movement Science, and the Department of Sports Medicine, Exercise Physiology and Prevention, in close cooperation with the Austrian Rowing Federation. The project's primary objective is to optimize the training load of elite athletes in endurance sports by leveraging large-scale data collected from individual training and performance sessions. Through the systematic processing and analysis of this data, AIROW seeks to accurately assess athletes' effective training load, forecast the outcomes of specific training stimuli, and enhance overall training management. Initially designed as a pilot project in rowing, the project is tailored to the sport's specific physiological and environmental demands. In the long term, the insights and methodologies developed through AIROW are intended to be transferred to other endurance disciplines that share comparable training structures and performance requirements, such as swimming, triathlon, cycling, cross-country skiing, and middle- to long-distance running. ([AIR22]).

2. Problem Statement

Endurance sports encompass a variety of training forms, one of the most prominent being interval training. For the purpose of improving both training quality and athletic performance, the identification of training intervals plays a crucial role in performance analysis, training planning, and athlete monitoring. Manual annotation of intervals, however, is highly time-consuming and demands significant effort, while relying solely on predefined laps or training plans often fails to reflect what truly occurred during a session. Existing automatic detection methods generally depend on power data, yet this approach is not always straightforward: challenges arise in defining what exactly constitutes an interval, determining the minimum duration or number of intervals, and dealing with incomplete or noisy data. Furthermore, power data are not always available, as not all training environments or sports disciplines utilize power meters.

In contrast, heart rate data are much more commonly available, since heart rate sensors are integrated into nearly all modern wearables and can be used across a wide range of activities. Nonetheless, heart-rate-based detection poses additional challenges: heart rate reacts to workload changes with a physiological delay and exhibits a gradual upward drift even under constant intensity ([SHD+21]). These factors make it difficult to design reliable heart-rate-based algorithms capable of accurately identifying training intervals in real-world conditions.

2.1. Research Questions

Q1: Can heart rate data alone be used to automatically detect interval boundaries in interval training sessions in endurance sports?

Q2: How do session-level characteristics influence model performance?

2.2. Contribution

This research contributes to sports science and applied machine learning by providing an automated, systematic method for detecting interval boundaries in endurance training sessions using heart rate data alone. Six machine learning architectures are implemented and evaluated, offering insights into which model types are best suited for this task. The

cross-sport evaluation across rowing and cycling sessions examines the generalizability of heart-rate-based detection across different endurance sport modalities. Additionally, a User-in-the-Loop annotation interface is introduced that leverages inter-model agreement, reducing manual annotation effort. The findings offer guidance for future research and practical applications in endurance sports, particularly for tools that assist athletes and coaches in systematic training analysis.

3. Related Work

In this section, works related to automatic interval detection in endurance sports will be discussed.

3.1. Power-Based Interval Detection

No scientific publications that describe or validate algorithms for fully automatic interval detection based solely on power data could be identified in this search. The only available implementations are found in commercial tools such as Intervals.icu [Int26], Athletica.ai [And25], and Vekta [Vek25]. These systems offer automated interval detection features, but their methods are proprietary and thus provide limited transparency. Only Athletica's Interval IQ™ and Vekta superficially share the process of their algorithms. Athletica first uses signal processing to smooth out spikes, drops, and noise in the power signal, then a clustering engine using change-point detection to identify segments of relatively constant power, after which post-processing merges similar efforts or splits embedded spikes (e.g., short sprints within longer tempo blocks), and finally concludes with context tagging, in which intervals are labeled with duration, average power, training zone, and heart rate. Vekta's approach centers on power relative to the athlete's Critical Power (CP), alongside cadence and torque characteristics, to detect and classify intervals. Each detected interval is assigned an intensity label, ranging from "Aerobic" to "Neuromuscular", based on its duration and its percentage of CP, and can be further tagged with descriptors such as "High Torque", "High Cadence", or "Progressive" to capture the qualitative nature of the effort. At the session level, Vekta then determines an overall training stimulus reflecting the dominant physiological demand of the session.

3.2. Heart-Rate-Based Interval Detection

Previous research on interval detection from heart rate (HR) data has largely relied on manual annotation or semi-automatic modeling approaches, such as exponential curve fitting and classification based on heart rate variability (HRV). In exponential modeling, HR responses during exercise or recovery are approximated by exponential functions that describe how HR increases or decreases over time. Transition points are then estimated by fitting these curves to the HR data; however, the accuracy of this method depends strongly on model assumptions and the manual selection of the time interval to be fitted

([BFdST+15]). HRV-based methods, in contrast, use statistical features of beat-to-beat variability to infer exertion levels, but these parameters are highly sensitive to individual characteristics such as age, sex, and cardiovascular fitness, limiting their generalizability ([JF15], [VPP+07]).

Among the few existing methods, one approach relies exclusively on HR data. It uses a fully automatic, model-free approach that applies HR variance analysis to detect transitions between rest, exercise, and recovery phases directly from wearable sensor recordings. The algorithm was validated only against manually annotated HR transitions, not against intervals derived from external measures such as power output or speed. Thus, both the detection and evaluation processes are entirely HR-based, focusing on patterns within the heart rate signal itself rather than on the actual training events that occurred. This approach represents a shift toward simpler, signal-driven methods for continuous cardiovascular monitoring in athletic contexts ([RSS+21]).

While the HR-only validation of Romagnoli et al. ([RSS+21]) demonstrates that physiological transitions can be identified directly from the heart rate signal, it also introduces an important limitation. Because both the detection and the ground truth were based solely on visual inspection of HR dynamics, the identified transitions represent changes in cardiovascular response, not necessarily the true physical onset or offset of exercise. In other words, the algorithm may detect when the heart rate begins to rise or recover, but this does not always coincide precisely with when the athlete actually starts or stops producing power. This distinction is critical for applications where training load quantification or performance analysis depend on accurate alignment with real activity intervals. Therefore, a key opportunity for further research lies in linking HR-based interval detection with external performance data, such as power, to create more context-aware and physiologically grounded models of training behavior.

3.3. Previous work

Before this present thesis, a preliminary project was conducted with the objective of developing and evaluating algorithms for the automatic detection of training intervals in cycling and rowing sessions based on power. The number of intervals was not provided to the models as metadata, requiring the algorithms to determine the interval count from the signal alone. Within that earlier work, three approaches were explored:

- **Heuristic threshold model.** A self-developed, threshold-based heuristic that segments the signal when it crosses predefined limits derived from power zones and the short-term rate of change in power.
- **Kernel change-point detection.** A kernel change-point detection method that flags distributional shifts in the signal (changes in mean and derivatives) using kernelized discrepancy measures ([GA18]); we evaluated linear, radial basis function (RBF), and cosine kernels for the cost function.

- **One-dimensional convolutional neural network.** A machine-learning approach that learns temporal dependencies and recurring patterns directly from labeled training examples ([KAA+21]).

Among these, the self-developed model achieved the most consistent and accurate results, demonstrating superior performance in identifying interval boundaries across the training data. A summary of results is provided in Appendix A.2.

4. Algorithmic viewpoint

The task of automatic interval boundary detection can also be viewed from an algorithmic perspective as a one-dimensional time-series segmentation problem. The goal is to identify the precise location of each interval boundary, framing the problem as a supervised classification task. Boundary detection is performed on complete, recorded sessions, making this an offline task where both past and future context are available at inference time.

In this thesis, the ground truth boundaries are derived from power data; however, the boundary detection itself must rely solely on heart rate. Unlike power, which responds almost instantaneously to variations in workload, heart rate represents a delayed and smoothed physiological response. Consequently, transitions between intervals are often gradual rather than abrupt, making conventional change-point detection methods, those based on sharp statistical discontinuities, less effective. Instead, this task requires models that can learn temporal dependencies, capture smooth and context-dependent transitions, and infer underlying effort changes from the evolving structure of the heart rate signal alone.

To address this challenge, we identify six machine learning architectures representing different strategies for one-dimensional time-series classification, spanning gradient boosting and neural networks of varying architectures. A seventh candidate, the signature-based variational auto-encoder (VAE) approach, is explored but ultimately not pursued due to computational constraints. The selection of the evaluated architectures is guided by two considerations. First, we include models that classify each timestamp independently based on its engineered feature vector (extreme gradient boosting, XGBoost; feedforward neural network, FFNN), allowing us to evaluate how far explicit feature engineering can carry the task without sequential modeling. Second, we include architectures that process temporal sequences directly (convolutional neural network, CNN; temporal convolutional network, TCN; bi-directional long short-term memory, BiLSTM; 1D U-Net), each embodying a different mechanism for capturing temporal structure.

4.1. XGBoost

XGBoost (extreme gradient boosting) is a gradient boosted decision tree algorithm that has become a standard choice for supervised learning on tabular data. It builds a sequence of decision trees, with each subsequent tree designed to improve upon the previous ones. XGBoost operates on a fixed-length feature vector for each timestamp

independently, without explicit modeling of sequential dependencies between adjacent timesteps [CG16]. Its inclusion serves two purposes: as a strong baseline representing the best achievable performance from traditional machine learning on engineered features, and as a deterministic, fully reproducible reference point against which the stochastic neural network models can be compared.

4.2. Feedforward Neural Network

A feedforward neural network (FFNN) processes each timestamp's feature vector through multiple fully connected layers to produce a boundary probability [GBC16]. Like XGBoost, it operates on individual timestamps without explicit sequential modeling, making it a per-timestamp classifier that relies entirely on engineered input features. The network learns continuous, differentiable decision boundaries through gradient-based optimization, enabling it to capture complex nonlinear feature interactions. The architecture additionally incorporates a variational bottleneck inspired by VAEs, introduced as a regularization mechanism to mitigate overfitting on the small training set. Rather than passing a fixed intermediate representation to the subsequent layers, the bottleneck encodes the input into a learned distribution in latent space, from which a sample is drawn during each forward pass. This stochastic compression discourages the network from memorizing individual training examples and encourages it to learn a more generalizable representation of the feature space [KW19].

4.3. Convolutional Neural Network

A one-dimensional convolutional neural network (CNN) applies learned filters over local temporal windows of the input sequence. Each convolutional layer extracts features by sliding a small kernel across the time axis, detecting local patterns within a fixed receptive field - the span of input timesteps visible to each filter. By stacking multiple layers, the network builds hierarchical representations where deeper layers capture increasingly abstract temporal patterns [KAA+21]. Unlike the per-timestamp models, the CNN processes a temporal window of feature vectors jointly, allowing it to detect patterns in how engineered features evolve across consecutive timesteps rather than treating each timestamp in isolation. This property aligns well with the boundary detection task, where transitions manifest as gradual shifts in heart rate across a local temporal neighborhood rather than abrupt statistical changes.

4.4. Temporal Convolutional Network

A Temporal Convolutional Network (TCN) extends the CNN approach through the use of dilated convolutions, where filters are applied at increasing intervals (e.g., every 1st,

2nd, 4th, 8th timestep) rather than to consecutive samples ([BKK18]). By increasing the dilation factor across layers, the network's receptive field grows exponentially without a proportional increase in parameters or computational cost. This design enables the TCN to capture both short-range patterns (e.g., a local increase in HR) and long-range dependencies (e.g., the overall trajectory of heart rate across a multi-minute interval) within a single forward pass. Unlike the original causal formulation of Bai et al. ([BKK18]), which restricts each timestep's prediction to depend only on past inputs, the present implementation uses symmetric (non-causal) padding to incorporate both past and future context — consistent with the offline setting. The combination of multi-scale receptive fields and bidirectional context makes the TCN well-suited to the heart rate segmentation task, where gradual physiological transitions must be inferred from smoothed, delayed signals rather than from abrupt statistical changes.

4.5. Bidirectional Long Short-Term Memory Network

A bi-directional long short-term memory (BiLSTM) network is a recurrent architecture that processes the input sequence in both forward and backward directions simultaneously [GS05]. At each timestep, the model has access to both past and future context through the concatenation of forward and backward hidden states. Given the offline setting, this bidirectional processing is a natural fit for boundary detection, since the heart rate context on both sides of a potential boundary is informative: a rising heart rate preceding a work phase onset and a declining heart rate following it, for instance. The recurrent structure also allows the BiLSTM to maintain a running representation of the full sequence history, without the architectural constraints on receptive field size inherent to convolutional approaches.

4.6. 1D U-Net

The U-Net architecture was originally developed for biomedical image segmentation and has since been adapted for one-dimensional temporal sequences [SED18]. A 1D U-Net uses an encoder that gradually reduces the temporal resolution while extracting higher-level features, and a decoder that restores the original resolution through upsampling. Skip connections between corresponding encoder and decoder layers combine coarse, high-level features with fine-grained temporal detail, enabling the network to process information across multiple time scales simultaneously. This multi-resolution structure makes the U-Net a promising architecture for heart rate-based segmentation, where both the global session structure (captured at coarse resolution) and precise boundary locations (requiring fine resolution) are relevant for accurate detection.

4.7. Variational Autoencoders with Path Signatures

Variational autoencoders (VAEs) with path signatures offer a generative approach to timeseries modeling that combines a compact encoder-decoder framework with signature-based feature extraction from rough path theory [BHL$^+$20]. The signature transform encodes a time series as a sequence of iterated integrals that capture its essential sequential properties, including drift, volatility, and higher-order path characteristics. The VAE learns to produce log-signatures whose distribution matches that of the observed data, operating effectively even in small data environments. The combination of signature-based encoding with a VAE's latent space is hypothesized to be applicable to heart rate time series, where similar sequential structure—trends, variability, and transition patterns—characterizes interval boundaries.

Although this approach was ultimately not pursued due to the excessive computational cost of signature computation at the scale of this dataset, it provided the conceptual basis for the variational bottleneck incorporated into the FFNN (Section 4.2).

5. Data

5.1. Data Origin

Training data was collected from two sports: rowing and cycling.

Rowing. Data was collected using Concept2 rowing ergometers. Athletes connect a heart rate monitor to the ergometer, complete their session, and the resulting FIT file is automatically uploaded to a central database. Heart rate is recorded on a stroke-by-stroke basis rather than at fixed time intervals.

Cycling. Data was collected from cycling sessions via two setups:

- *Power meter:* Athletes use a power meter attached to the pedals or crank arm of their bike. Power and heart rate data are recorded by a bike computer and exported as a FIT file, which is then uploaded to the cloud.
- *Virtual training software:* Athletes use platforms such as Zwift or MyWhoosh, where the smart trainer measures and transmits power output to the application.

In both cycling setups, power is sampled at 1 Hz and heart rate is recorded using a heart rate monitor, such as a smartwatch or chest strap.

5.2. Definition of an Interval

Interval training can be defined as repeated high-intensity work bouts alternating with periods of lower-intensity recovery or rest. In performance contexts, exercise intensity is typically divided into three main domains: moderate, heavy, and severe. For athletes and high-level sports, a finer classification is often used, extending the model to five, six, or seven distinct zones to allow for more precise exercise description ([CJL+23]).

However, in this thesis, we do not necessarily classify an interval solely based on transitions between zones. Instead, we adhere to the general definition introduced above of an interval as a period of sustained higher effort followed by recovery. Consequently, a single interval may encompass several higher zones depending on the intensity dynamics observed in the power data.

5.3. Ground Truth Data

To obtain reliable ground truth data for model development and evaluation, a custom annotation tool was developed as part of a previous project by this author. This web-based application allowed users to import recorded training sessions and manually mark the start and end points of individual intervals within each session, based on power data. Using this tool, a total of 93 training sessions from indoor rowing and outdoor rowing were manually annotated. The annotations provided the temporal boundaries required for subsequent algorithm validation and served as the reference dataset in this study. The tool is available online.[1]

Figure 5.1.: Example Session with Manually Annotated Boundaries

It is important to distinguish between the athlete's perspective and the annotation scheme used in this study. An athlete reporting "six intervals" typically refers to six high-intensity work bouts. However, in our framework, every sustained period of effort or recovery constitutes an interval, including the initial warm-up and each recovery phase between work bouts. Consequently, a session contains more intervals than an athlete would typically report. An interval boundary marks the transition point between two consecutive intervals. The session shown in Figure 5.1 contains 12 annotated boundaries, which divide the session into 12 intervals: a warm-up, five work bouts, five recovery phases, and a cool-down. Session structures vary across athletes and disciplines, and not all sessions follow this pattern.

Figure 5.2 shows another example of an annotated training session.

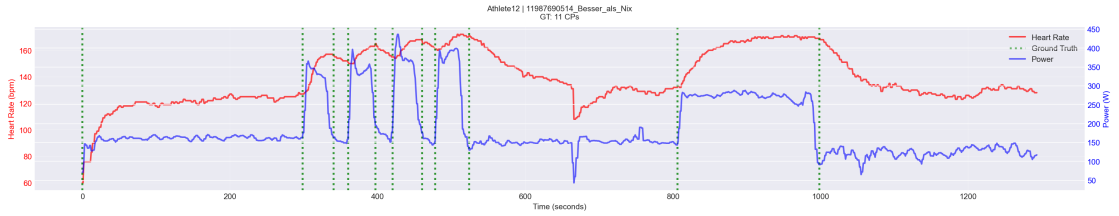

Figure 5.2.: Example Session 2 with Manually Annotated Boundaries

[1] https://airow-smm5.onrender.com/

6. Implementation

This chapter presents the machine learning approaches investigated for automatic boundary detection, the shared components underlying all models, and the evaluation framework used to assess and compare their performance. Six model architectures were evaluated, spanning gradient boosting and neural networks of varying architectures, with four of the neural network models additionally tested under a reduced feature set, yielding ten model configurations in total. Complete implementation details and code are available in the accompanying open-source repository. Figure 6.1 gives an overview of the machine learning pipeline that all model configurations follow, from raw heart rate input through feature engineering and model inference to the final boundary predictions.

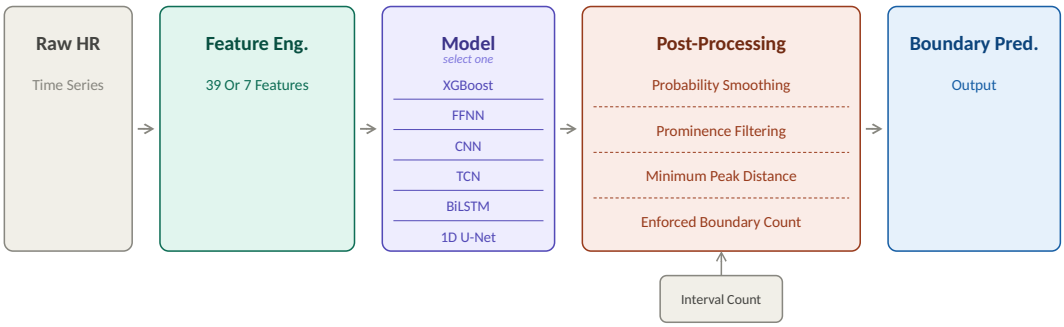

Figure 6.1.: Overview of the machine learning pipeline for boundary detection

6.1. Shared Pipeline Components

All model configurations were trained on identical training data and share common components for feature engineering and post-processing. These shared elements ensure a fair comparison and allow us to isolate the impact of each modeling technique.

6.1.1. Feature Engineering

Two feature sets were constructed and evaluated: a full set of 39 handcrafted features capturing the temporal and statistical properties of the heart rate signal, and a minimal reduced set of 7 features consisting of the heart rate signal and session-level descriptors, without any temporal feature engineering. This dual approach allows us to assess whether sequential neural network architectures can learn relevant temporal patterns directly from minimally processed input, or whether explicit feature engineering is necessary. Since XGBoost and FFNN operate on individual timestamps without sequential modeling, the reduced set was only applied to the four sequential architectures: CNN, TCN, BiLSTM, and 1D U-Net.

Full Feature Set (39 Features)

All features are computed relative to session-level statistics (mean, standard deviation, interquartile range, percentiles) to normalize for differences in heart rate ranges across athletes and sports. For each timestamp, we computed features across seven categories:

1. **Workout structure features** (3 features): boundary density (boundaries per minute), expected interval length, and a binary flag for high-density sessions (≥ 20 boundaries). These features encode prior knowledge about the session's overall structure.
2. **Smoothed and rolling statistics** (7 features): smoothed heart rate over medium and long windows, rolling maximum, minimum, range, and standard deviation over a medium window, and a short-window rolling maximum. These features reduce noise and expose sustained trends in the signal.
3. **Session-normalized intensity features** (9 features): z-score of the current heart rate, smoothed heart rate normalized by session statistics, differences from session mean, 25th and 75th percentile, and long-term baseline, and ratio features expressing heart rate as a fraction of session range, maximum, and mean. These situate each timestamp within the session context, helping the model account for inter-athlete differences in heart rate range.
4. **Temporal derivatives** (6 features): finite differences at 5, 10, and 30-second intervals, normalized derivatives at 10 and 30 seconds, and percentage change over 30 seconds. These capture the velocity of heart rate change across multiple time scales.
5. **Look-ahead and look-behind features** (8 features): future heart rate at 30 and 60 seconds, past heart rate at 30 seconds (raw and normalized), changes from lagged values at 30 and 60 seconds, and a bidirectional change feature capturing the total heart rate difference between 60 seconds in the future and 60 seconds in the past, in raw and normalized form. These provide explicit access to future and past

context, which is feasible given the offline setting and useful for detecting gradual transitions.

6. **Zone and trend indicators** (3 features): a binary trend flag indicating whether heart rate is currently rising, and binary zone indicators for high-intensity (above $\mu + \sigma$) and low-intensity (below the 25th percentile) regions. These complement the continuous intensity features by providing discrete indicators of effort level, which are particularly informative at transitions between intensity zones.
7. **Session-level constants** (3 features): session mean, standard deviation, and interquartile range of heart rate, broadcast to every timestamp as contextual features. They provide the model with explicit access to absolute session-level distributional information, alongside the per-timestamp normalized values.

All feature values were standardized using z-score normalization based on training set statistics prior to model training.

Reduced Feature Set (7 Features)

To assess how much the models rely on explicit feature engineering, we also evaluated models using a minimal set of only 7 features. This reduced set contains the heart rate signal, three session-level heart rate statistics (mean, standard deviation, and interquartile range), and three workout structure features (boundaries per minute, expected interval length, and a binary high-density flag). The reduced set omits all temporally engineered features, derivatives, rolling statistics, look-ahead/look-behind values, and normalized intensity measures, retaining only the heart rate signal and session-level context. This design allows for a direct comparison between minimally processed and fully engineered input representations across the sequential architectures.

6.1.2. Post-Processing Pipeline

All models output continuous probability scores $P(\text{boundary})_t \in [0, 1]$ for each timestamp. Converting these into discrete boundary predictions requires a shared post-processing pipeline.

Probability Smoothing. The raw probability outputs can contain brief, isolated spikes that do not correspond to true boundaries. We applied a Gaussian smoothing filter $\sigma = 3.0$, which averages each timestamp's probability with its temporal neighbors. This reduces isolated spikes, while regions where the model consistently predicts high boundary probability remain largely unaffected.

Peak Detection via Prominence Filtering. From the smoothed probability signal, boundaries were identified as local maxima (peaks). To distinguish meaningful peaks from residual noise, we applied a minimum prominence threshold of 0.15. Prominence measures how much a peak rises above its surrounding landscape and is calculated as

follows: from the peak, the algorithm traverses the signal in both directions (left and right) until it encounters a higher peak or reaches the edge of the signal. Along each direction, it records the lowest probability value crossed. The higher of these two lowest values defines the peak's base, and the prominence is the difference between the peak height and this base. A prominence of 0.15 therefore requires that a peak rises at least 0.15 above the higher of the two valleys that separate it from its neighboring peaks. This filters out small fluctuations that survived Gaussian smoothing but do not represent sufficient changes in boundary probability to indicate a true interval transition.

Minimum Peak distance. A minimum distance constraint between consecutive boundaries was enforced to prevent detecting multiple boundaries within the same transition, adapted to the expected boundary density of each session:

Expected Boundaries	Minimum Peak Distance
<10 boundaries	30 seconds
10–14 boundaries	20 seconds
15–19 boundaries	15 seconds
20–24 boundaries	10 seconds
25–34 boundaries	8 seconds
35–49 boundaries	5 seconds
≥50 boundaries	3 seconds

When multiple candidate peaks fell within the minimum distance window, the constraint was resolved by retaining the peak with the highest predicted probability and suppressing the lower-scoring neighbor. This means that if two true boundaries were separated by less than the enforced minimum distance, the one with the lower model confidence would be discarded.

Enforced Boundary Count. Since the number of boundaries per session is known a priori (from session metadata), peak detection was constrained to return exactly n boundaries. When more than n peaks were detected, the n peaks with the highest probability scores were retained. When fewer than n peaks were found, the prominence constraint was relaxed and peak detection was retried; if still insufficient, the n timestamps with the highest probability values were selected subject to a minimum distance constraint. The first timestamp ($t = 0$) was always included as the initial boundary.

Algorithm 1: EnforceBoundaryCount: Match Peak Set to Expected Count

Input: Peak set $\mathcal{P}$, probability sequence $\tilde{\mathbf{p}}$, required count n_{rem}, minimum distance δ

Output: Set of exactly n_{rem} boundary indices

if $|\mathcal{P}| = n_{rem}$ **then**
  return $\mathcal{P}$;
else if $|\mathcal{P}| > n_{rem}$ **then**
  return *top* n_{rem} *elements of* $\mathcal{P}$ *ranked by* $\tilde{\mathbf{p}}$;
else
  `// Too few: retry without prominence constraint`
  $\mathcal{P}' \leftarrow$ FindPeaks($\tilde{\mathbf{p}}$, distance $\geq \delta$);
  if $|\mathcal{P}'| \geq n_{rem}$ **then**
    return *top* n_{rem} *elements of* $\mathcal{P}'$ *ranked by* $\tilde{\mathbf{p}}$;
  `// Last resort: greedy distance-constrained selection`
  $\mathcal{C} \leftarrow$ top $3 \cdot n_{\text{rem}}$ indices of $\tilde{\mathbf{p}}$ ranked by probability;
  $\mathcal{S} \leftarrow \emptyset$ `// selected boundary set`
  foreach $c \in \mathcal{C}$ *in descending probability order* **do**
    if $|c - s| \geq \delta \ \ \forall\, s \in \mathcal{S}$ **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$;
      if $|\mathcal{S}| = n_{rem}$ **then return** $\mathcal{S}$;
  return $\mathcal{S}$;

The following algorithm summarizes the complete post-processing pipeline:

Algorithm 2: Post-Processing: Adaptive Peak Detection with Boundary Count Constraint

Input: Probability sequence $\mathbf{p} \in [0,1]^T$, expected boundary count n
Output: Sorted set of n boundary indices $\hat{B}$

```
// Force first boundary at session start
```
$\hat{B}_0 \leftarrow \{0\}$;
$n_{\text{rem}} \leftarrow n - 1$;
```
// Smooth probabilities to suppress isolated spikes
```
$\tilde{\mathbf{p}} \leftarrow \textsc{GaussianSmooth}(\mathbf{p},\ \sigma = 3.0)$;
```
// Set minimum peak distance based on expected boundary count
```
$\delta \leftarrow \textsc{MinPeakDistance}(n)$;
```
// Find peaks with prominence and distance constraints
```
$\mathcal{P} \leftarrow \textsc{FindPeaks}(\tilde{\mathbf{p}}[\delta:],\ \text{prominence} \geq 0.15,\ \text{distance} \geq \delta)$;
```
// Enforce boundary count
```
$\hat{B} \leftarrow \textsc{Sort}(\hat{B}_0 \cup \textsc{EnforceBoundaryCount}(\mathcal{P},\ \tilde{\mathbf{p}},\ n_{\text{rem}},\ \delta))$;
return $\hat{B}$;

6.2. Model Architectures

We evaluated six distinct model architectures, spanning gradient boosting and several neural networks. The conceptual motivation for each architecture is presented in Chapter 4; the following subsections describe each architecture's specific configuration and training procedure, with comparative results presented in Section 7.2. In addition, a signature-based VAE approach (Section 4.7) was initially explored but abandoned due to the excessive computational cost of signature computation at the scale of this dataset; its encoder-decoder structure with a stochastic latent bottleneck was nevertheless retained and adapted into the FFNN architecture described in Section 6.2.2.

Four of the six architectures - CNN, TCN, BiLSTM, and U-Net - operate on fixed-length overlapping segments rather than individual timestamps. This sliding window inference procedure is identical across all four models and is described once in Algorithm 5; XGBoost and the FFNN process each timestamp independently and do not use it.

6.2.1. XGBoost

XGBoost passes each timestamp's feature vector through an ensemble of independent decision trees, each asking a sequence of yes/no questions about individual features, and combines their outputs into a single boundary probability. The ensemble of many simple trees collectively captures complex patterns that no single tree could reliably detect.

Configuration. The classifier employed 400 decision trees with a maximum depth of 6, learning rate of 0.03, and regularization parameters including minimum child weight of 5, gamma of 0.2, subsample ratio of 0.8, and column sampling ratio of 0.8. To address the severe class imbalance - boundary timestamps constitute less than 1% of all timesteps - a positive class weight of 30.0 was applied. XGBoost operates on the 39-dimensional feature vector for each individual timestamp independently, though the look-ahead and look-behind features in the input partially compensate for the absence of explicit sequential modeling.

Algorithm 3: XGBoost: Boundary Probability Inference

Input: Feature vector $\mathbf{x}_t \in R^{39}$ for timestamp t;
ensemble of $T = 400$ trained decision trees $\{f_1, \ldots, f_T\}$;
each tree f_k contains internal nodes with a split feature index j and threshold θ, and leaf nodes with a learned weight w (log-odds contribution, scaled by positive class weight $w^+ = 30.0$)

Output: Boundary probability $\hat{p}_t \in [0, 1]$

```
// Each tree votes independently; sum their outputs
```
$s \leftarrow 0$;
for $k = 1$ **to** T **do**
```
    // Traverse tree f_k from root to leaf:
    // at each internal node, compare one feature to a threshold
```
    node $\leftarrow$ root of f_k;
    while *node is not a leaf* **do**
        if $x_t^{(j_{node})} \leq \theta_{node}$ **then**
            node $\leftarrow$ left child of node;
        else
            node $\leftarrow$ right child of node;
    $s \leftarrow s + w_{\text{node}}$; `// add this tree's vote to the running total`

```
// s is a raw score (log-odds); convert to a probability in [0,1]
```
$\hat{p}_t \leftarrow \sigma(s) = \dfrac{1}{1 + e^{-s}}$;
return $\hat{p}_t$;

6.2.2. Feedforward Neural Network

The FFNN processes each timestamp by compressing the 39-dimensional feature vector down to a 12-dimensional representation, injecting a small amount of controlled randomness, and then expanding back up to produce a single boundary probability, the bottleneck forces the network to distill only the most essential patterns while the stochasticity acts as a regularizer against overfitting.

Architecture. The network processes the 39-dimensional feature vector through an

encoder-decoder structure with a variational bottleneck, consisting of six fully-connected layers ($39 \to 128 \to 32 \to 12 \to 32 \to 128 \to 1$, ReLU (Rectified Linear Unit) activation, sigmoid output). Optuna hyperparameter optimization over 50 trials drove the Kullback-Leibler (KL) divergence weight to $\lambda_{\text{KL}} = 0.000149$, indicating that the variational objective contributes minimally to the loss. Nevertheless, ablation experiments removing the reparameterization bottleneck entirely resulted in performance below $F_\beta = 0.7$, suggesting that the stochastic sampling itself provides a meaningful regularization benefit independent of the KL term. The architecture therefore retains the variational bottleneck while operating in a near-deterministic regime.

Training. The loss function combines weighted binary cross-entropy with a positive class weight of 60.0 to address the severe class imbalance and a KL divergence term weighted by $\lambda_{\text{KL}} = 0.000149$. Training used the Adam optimizer with a learning rate of 0.000636, batch size of 1024, and 50 epochs.

Algorithm 4: FFNN: Boundary Probability Inference

Input: Feature vector $\mathbf{x}_t \in R^{39}$ for timestamp t;
trained encoder weights W_{enc} mapping $39 \to 128 \to 32$;
trained bottleneck weights producing $\boldsymbol{\mu}, \log \boldsymbol{\sigma}^2 \in R^{12}$;
trained decoder weights W_{dec} mapping $12 \to 32 \to 128 \to 1$

Output: Boundary probability $\hat{p}_t \in [0, 1]$

`// Encode input to latent space`
$\mathbf{h} \leftarrow \text{ReLU}(W_{\text{enc}} \cdot \mathbf{x}_t)$; `// two fully-connected layers:` $39 \to 128 \to 32$
`// Variational bottleneck: estimate latent distribution parameters`
$\boldsymbol{\mu}, \log \boldsymbol{\sigma}^2 \leftarrow \text{Linear}(\mathbf{h})$; `// two separate linear projections from 32-dim` $\mathbf{h}$ `to` R^{12}
`// Reparameterization trick: sample from latent distribution`
$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
$\mathbf{z} \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$; `//` $\boldsymbol{\sigma} = \exp(\frac{1}{2} \log \boldsymbol{\sigma}^2)$; $\mathbf{z} \in R^{12}$; `stochastic regularization`
`// Decode latent sample to boundary probability`
$\mathbf{h}' \leftarrow \text{ReLU}(W_{\text{dec}} \cdot \mathbf{z})$; `// two fully-connected layers:` $12 \to 32 \to 128$
$\hat{p}_t \leftarrow \sigma(\mathbf{w}_{\text{out}} \cdot \mathbf{h}')$; `// final linear layer` $128 \to 1$ `with sigmoid activation`
return $\hat{p}_t$;

6.2.3. Segment-Based Inference

The four neural network architectures that operate on fixed-length input segments (CNN, TCN, BiLSTM, U-Net) share an identical sliding window inference procedure. Rather than processing the full session at once, the session is divided into overlapping segments of 300 timestamps with a 50% overlap (stride of 150). Each timestamp is therefore covered by two consecutive windows, and predictions from overlapping regions are averaged, this improves stability at segment boundaries compared to a single prediction per timestamp. A special case handles sessions whose length is not aligned with the stride, ensuring full

coverage of the tail of the session.

Algorithm 5: Segment-Based Inference: Sliding Window and Stitching

Input: Feature matrix $\mathbf{X} \in R^{T \times F}$ for a full session of T timestamps and F features;
segment length $L = 300$, stride $\Delta = 150$;
model $\mathcal{M}$ (any of CNN, TCN, BiLSTM, U-Net)

Output: Boundary probability sequence $\hat{\mathbf{p}} \in [0, 1]^T$

`// Initialise output accumulator and per-timestamp prediction count`
$\hat{\mathbf{p}} \leftarrow \mathbf{0}^T$, $\mathbf{c} \leftarrow \mathbf{0}^T$;
`// Slide window across session with stride` Δ
for *each segment start index* $s \in \{0, \Delta, 2\Delta, \ldots\}$ *while* $s < T$ **do**
  $\mathbf{X}_s \leftarrow \mathbf{X}[s : s + L]$; `// extract segment of shape` $L \times F$
  $\hat{\mathbf{p}}_s \leftarrow \mathcal{M}(\mathbf{X}_s)$; `// forward pass; returns` L `probabilities`
  $\hat{\mathbf{p}}[s : s + L] \leftarrow \hat{\mathbf{p}}[s : s + L] + \hat{\mathbf{p}}_s$;
  $\mathbf{c}[s : s + L] \leftarrow \mathbf{c}[s : s + L] + 1$;
`// Handle tail: if final timestamps were not covered, process last segment`
if $\mathbf{c}[T - 1] = 0$ **then**
  $\mathbf{X}_s \leftarrow \mathbf{X}[T - L : T]$;
  $\hat{\mathbf{p}}_s \leftarrow \mathcal{M}(\mathbf{X}_s)$;
  $\hat{\mathbf{p}}[T - L : T] \leftarrow \hat{\mathbf{p}}[T - L : T] + \hat{\mathbf{p}}_s$;
  $\mathbf{c}[T - L : T] \leftarrow \mathbf{c}[T - L : T] + 1$;
`// Average predictions in overlapping regions`
$\hat{\mathbf{p}} \leftarrow \hat{\mathbf{p}} \odot \mathbf{c}^{-1}$;
return $\hat{\mathbf{p}}$;

6.2.4. Convolutional Neural Network

The CNN applies convolutional filters at multiple scales to capture both local and gradual heart rate transitions within each segment, with sliding window inference following Algorithm 5. Within each segment, an initial projection re-weights the input features, residual blocks scan across time at increasing scales to detect patterns ranging from sharp second-level transitions to broader minute-level trends, and a two-layer output head maps the learned representations to per-timestamp boundary probabilities.

Architecture. An initial 1×1 convolutional projection with batch normalization and ReLU maps the input to 64 filters, followed by four residual blocks with multi-scale kernel sizes (3, 7, 15, and 31). Filter counts increase from 64 to 128 in the deeper blocks. Each residual block contains two convolutional layers with batch normalization, ReLU activation, and dropout (0.2–0.3). A two-layer output head consisting of a 1×1 convolution with 64 filters and ReLU activation followed by dropout (0.3), and a final 1×1

convolution with sigmoid activation, produces per-timestamp boundary probabilities.

Training. The model was trained using weighted binary cross-entropy with a positive class weight computed as (total samples − positive samples)/positive samples to address the severe class imbalance, the Adam optimizer with a learning rate of 0.001, and a batch size of 32. The validation set was constructed at session level, holding out approximately 15% of training sessions. Training used early stopping with a patience of 15 epochs, and a ReduceLROnPlateau scheduler that halves the learning rate when validation loss does not improve for 5 consecutive epochs, with a minimum learning rate of 10^{-6}.

Algorithm 6: CNN: Forward Pass for a Single Segment

Input: Segment $\mathbf{X}_s \in R^{300\times F}$ where $F \in \{7, 39\}$;
trained 1×1 projection weights;
trained residual block weights with kernel sizes $(3, 7, 15, 31)$ and filters $(64, 64, 128, 128)$;
trained output head weights (64-filter projection, then 1-filter output)

Output: Boundary probability sequence $\hat{\mathbf{p}}_s \in [0, 1]^{300}$

`// Project input features to 64 filters`

$\mathbf{H}_0 \leftarrow \mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv}_{1\times1}(\mathbf{X}_s)))$; `// shape:` 300×64

`// Pass through four residual blocks with increasing receptive fields`

for $k = 1$ **to** 4 **do**

  $\mathbf{H}_k \leftarrow \textsc{ResBlock}_k(\mathbf{H}_{k-1})$; `// kernel sizes 3,7,15,31; filters 64,64,128,128`

`// Output head: intermediate projection then final output`

$\mathbf{H}_5 \leftarrow \mathrm{ReLU}(\mathrm{Conv}_{1\times1,64}(\mathbf{H}_4))$; `// intermediate 64-filter projection with dropout`

$\hat{\mathbf{p}}_s \leftarrow \sigma(\mathrm{Conv}_{1\times1,1}(\mathbf{H}_5))$; `// shape:` 300×1

return $\hat{\mathbf{p}}_s$;

6.2.5. Temporal Convolutional Network

The TCN replaces the standard convolutions of the CNN with *dilated* convolutions, filters that skip over inputs at exponentially increasing gaps, allowing the network to capture patterns across a much wider time range without requiring more parameters. Seven residual blocks with doubling dilation rates give the TCN a receptive field of 509 timestamps, exceeding the full 300-timestamp segment, meaning every boundary probability prediction can draw on context from the entire input window. Sliding window inference follows Algorithm 5.

Architecture. An initial 1×1 convolutional projection with batch normalization and ReLU maps the input to 64 filters, followed by seven residual blocks with exponentially increasing dilation rates $(1, 2, 4, 8, 16, 32, 64)$ and a kernel size of 3, yielding a receptive field of 509 timestamps. Unlike the original causal formulation of Bai et al. [BKK18],

symmetric (non-causal) padding is used, allowing the network to incorporate both past and future context, appropriate for offline segmentation where complete sessions are available at inference time. Each residual block contains two dilated convolutions with batch normalization, ReLU activation, dropout (0.35), and L2 regularization (10^{-4}). A two-layer output head consisting of a 1×1 convolution with 32 filters and ReLU activation followed by dropout (0.35), and a final 1×1 convolution with sigmoid activation, produces per-timestamp boundary probabilities.

Training. The model was trained using weighted binary cross-entropy with a positive class weight computed as (total samples − positive samples)/positive samples to address the severe class imbalance, the Adam optimizer with a learning rate of 0.001, and a batch size of 32. The validation set was constructed at session level, holding out approximately 15% of training sessions. Training used early stopping with a patience of 15 epochs, and a ReduceLROnPlateau scheduler that halves the learning rate when validation loss does not improve for 5 consecutive epochs, with a minimum learning rate of 10^{-6}.

Algorithm 7: TCN: Forward Pass for a Single Segment

Input: Segment $\mathbf{X}_s \in R^{300 \times F}$ where $F \in \{7, 39\}$;
trained 1×1 projection weights;
trained residual block weights with dilation rates $(1, 2, 4, 8, 16, 32, 64)$ and kernel size 3;
trained output head weights (32-filter projection, then 1-filter output)

Output: Boundary probability sequence $\hat{\mathbf{p}}_s \in [0, 1]^{300}$

`// Project input features to 64 filters`

$\mathbf{H}_0 \leftarrow \mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv}_{1\times1}(\mathbf{X}_s)))$; `// shape:` 300×64

`// Pass through seven residual blocks with exponentially increasing dilation rates`

for $k = 1$ **to** 7 **do**

  $\mathbf{H}_k \leftarrow \textsc{DilatedResBlock}_k(\mathbf{H}_{k-1})$; `// dilation` 2^{k-1}`; each block skips over inputs at increasing gaps`

`// Output head: intermediate projection then final output`

$\mathbf{H}_8 \leftarrow \mathrm{ReLU}(\mathrm{Conv}_{1\times1,32}(\mathbf{H}_7))$; `// intermediate 32-filter projection with dropout`

$\hat{\mathbf{p}}_s \leftarrow \sigma(\mathrm{Conv}_{1\times1,1}(\mathbf{H}_8))$; `// shape:` 300×1

return $\hat{\mathbf{p}}_s$;

6.2.6. Bidirectional Long Short-Term Memory Network

The BiLSTM processes each segment by reading the heart rate sequence in both forward and backward directions simultaneously, allowing every timestamp's representation to incorporate context from both past and future within the segment. Sliding window inference follows Algorithm 5.

Architecture. Two stacked BiLSTM layers with 128 and 64 units respectively produce concatenated forward and backward hidden states ($2 \times 128 = 256$ and $2 \times 64 = 128$ dimensions) for every timestep. Each recurrent layer is followed by batch normalization and uses both input dropout (0.3) and recurrent dropout (0.2). A time-distributed dense head maps each timestep through a 32-neuron ReLU layer with dropout (0.3) to a single sigmoid output.

Training. The model was trained using weighted binary cross-entropy with a positive class weight computed as (total samples − positive samples)/positive samples to address the severe class imbalance, the Adam optimizer with a learning rate of 0.001, and a batch size of 32. The validation set was constructed at session level, holding out approximately 15% of training sessions. Training used early stopping with a patience of 15 epochs, and a ReduceLROnPlateau scheduler that halves the learning rate when validation loss does not improve for 5 consecutive epochs, with a minimum learning rate of 10^{-6}.

Algorithm 8: BiLSTM: Forward Pass for a Single Segment

Input: Segment $\mathbf{X}_s \in R^{300\times F}$ where $F \in \{7, 39\}$;
trained BiLSTM layer 1 weights (128 units, forward and backward, input dropout 0.3, recurrent dropout 0.2);
trained BiLSTM layer 2 weights (64 units, forward and backward, input dropout 0.3, recurrent dropout 0.2);
trained time-distributed dense head weights (32-neuron ReLU, dropout 0.3, then sigmoid)

Output: Boundary probability sequence $\hat{\mathbf{p}}_s \in [0, 1]^{300}$

`// First BiLSTM layer: read segment in both directions`

$\overrightarrow{\mathbf{H}}_1 \leftarrow \mathrm{LSTM}_{128}(\mathbf{X}_s, \text{forward},\ p_{\text{in}}{=}0.3,\ p_{\text{rec}}{=}0.2)$; `// hidden states shape:` 300×128

$\overleftarrow{\mathbf{H}}_1 \leftarrow \mathrm{LSTM}_{128}(\mathbf{X}_s, \text{backward},\ p_{\text{in}}{=}0.3,\ p_{\text{rec}}{=}0.2)$; `// hidden states shape:` 300×128

$\mathbf{H}_1 \leftarrow [\overrightarrow{\mathbf{H}}_1 \,\|\, \overleftarrow{\mathbf{H}}_1]$; `// concatenate; shape:` 300×256

$\mathbf{H}_1 \leftarrow \textsc{BatchNorm}(\mathbf{H}_1)$;

`// Second BiLSTM layer: refine representations`

$\overrightarrow{\mathbf{H}}_2 \leftarrow \mathrm{LSTM}_{64}(\mathbf{H}_1, \text{forward},\ p_{\text{in}}{=}0.3,\ p_{\text{rec}}{=}0.2)$; `// hidden states shape:` 300×64

$\overleftarrow{\mathbf{H}}_2 \leftarrow \mathrm{LSTM}_{64}(\mathbf{H}_1, \text{backward},\ p_{\text{in}}{=}0.3,\ p_{\text{rec}}{=}0.2)$; `// hidden states shape:` 300×64

$\mathbf{H}_2 \leftarrow [\overrightarrow{\mathbf{H}}_2 \,\|\, \overleftarrow{\mathbf{H}}_2]$; `// concatenate; shape:` 300×128

$\mathbf{H}_2 \leftarrow \textsc{BatchNorm}(\mathbf{H}_2)$;

`// Time-distributed dense head: applied independently to each timestep`

$\mathbf{H}_3 \leftarrow \textsc{Dropout}(\mathrm{ReLU}(\mathbf{W}_{\text{dense}} \cdot \mathbf{H}_2), 0.3)$; `// shape:` 300×32

$\hat{\mathbf{p}}_s \leftarrow \sigma(\mathbf{w}_{\text{out}} \cdot \mathbf{H}_3)$; `// shape:` 300×1

return $\hat{\mathbf{p}}_s$;

6.2.7. 1D U-Net

The U-Net processes each segment through a contracting encoder and a symmetric expanding decoder. The encoder progressively compresses the temporal resolution to force the network to learn abstract, high-level patterns. The decoder then restores the original resolution step by step, and at each level it receives a skip connection from the corresponding encoder level, reintroducing the fine-grained timing detail that was lost during compression. This allows the network to combine abstract pattern recognition from the bottleneck with precise localization from the encoder, which is particularly valuable for detecting the exact position of boundaries. Sliding window inference follows Algorithm 5.

Architecture. The encoder consists of three levels of paired convolutional blocks with batch normalization, ReLU activation, dropout (0.25), and L2 regularization (10^{-4}), followed by max-pooling for downsampling (factors of 2, 2, and 3, yielding temporal resolutions of 150, 75, and 25). Filter counts increase from 32 to 64 to 128, with a fixed kernel size of 3 at all levels. A 256-filter bottleneck connects to a symmetric three-level decoder that upsamples via nearest-neighbor interpolation and concatenates skip connections from the corresponding encoder level before applying paired convolutions mirroring the encoder's configuration. A final 1×1 convolution with sigmoid activation produces per-timestamp boundary probabilities.

Training. The model was trained using weighted binary cross-entropy with a positive class weight computed as (total samples − positive samples)/positive samples to address the severe class imbalance, the Adam optimizer with a learning rate of 0.001, and a batch size of 32. The validation set was constructed at session level, holding out approximately 15% of training sessions. Training used early stopping with a patience of 15 epochs, and a ReduceLROnPlateau scheduler that halves the learning rate when validation loss does not improve for 5 consecutive epochs, with a minimum learning rate of 10^{-6}.

Algorithm 9: 1D U-Net: Forward Pass for a Single Segment

Input: $\mathbf{X}_s \in R^{300\times F}$, input segment ($F \in \{7, 39\}$)
$\boldsymbol{\theta}_{\text{enc},k}$: trained weights of encoder ConvBlock$_k$, $k = 1, 2, 3$
$\boldsymbol{\theta}_{\text{bn}}$: trained weights of bottleneck ConvBlock$_b$
$\boldsymbol{\theta}_{\text{dec},k}$: trained weights of decoder ConvBlock$_k$, $k = 3, 2, 1$
$\boldsymbol{\theta}_{\text{out}}$: trained weights of output Conv(1, k=1)
Output: Boundary probability sequence $\hat{\mathbf{p}}_s \in [0, 1]^{300}$

// -- Encoder --
$\mathbf{H}_1 \leftarrow \text{ConvBlock}_1(\mathbf{X}_s; \boldsymbol{\theta}_{\text{enc},1})$; // shape: 300×32; two Conv1D(32, k=3)-BN-ReLU pairs, Dropout(0.25)
$\mathbf{P}_1 \leftarrow \text{MaxPool}(\mathbf{H}_1, \text{size} = 2)$; // shape: 150×32
$\mathbf{H}_2 \leftarrow \text{ConvBlock}_2(\mathbf{P}_1; \boldsymbol{\theta}_{\text{enc},2})$; // shape: 150×64; two Conv1D(64, k=3)-BN-ReLU pairs, Dropout(0.25)
$\mathbf{P}_2 \leftarrow \text{MaxPool}(\mathbf{H}_2, \text{size} = 2)$; // shape: 75×64
$\mathbf{H}_3 \leftarrow \text{ConvBlock}_3(\mathbf{P}_2; \boldsymbol{\theta}_{\text{enc},3})$; // shape: 75×128; two Conv1D(128, k=3)-BN-ReLU pairs, Dropout(0.25)
$\mathbf{P}_3 \leftarrow \text{MaxPool}(\mathbf{H}_3, \text{size} = 3)$; // shape: 25×128

// -- Bottleneck --
$\mathbf{H}_b \leftarrow \text{ConvBlock}_b(\mathbf{P}_3; \boldsymbol{\theta}_{\text{bn}})$; // shape: 25×256; two Conv1D(256, k=3)-BN-ReLU pairs, Dropout(0.25)

// -- Decoder --
$\mathbf{D}_3 \leftarrow \text{ConvBlock}_3([\,\text{Upsample}(\mathbf{H}_b, \text{size} = 3) \mid \mathbf{H}_3\,]; \boldsymbol{\theta}_{\text{dec},3})$; // upsample: 75×256; concat: 75×384; out: 75×128
$\mathbf{D}_2 \leftarrow \text{ConvBlock}_2([\,\text{Upsample}(\mathbf{D}_3, \text{size} = 2) \mid \mathbf{H}_2\,]; \boldsymbol{\theta}_{\text{dec},2})$; // upsample: 150×128; concat: 150×192; out: 150×64
$\mathbf{D}_1 \leftarrow \text{ConvBlock}_1([\,\text{Upsample}(\mathbf{D}_2, \text{size} = 2) \mid \mathbf{H}_1\,]; \boldsymbol{\theta}_{\text{dec},1})$; // upsample: 300×64; concat: 300×96; out: 300×32

// -- Output --
$\hat{\mathbf{p}}_s \leftarrow \sigma(\text{Conv}(\mathbf{D}_1, \text{filters} = 1, k{=}1; \boldsymbol{\theta}_{\text{out}}))$; // shape: $300 \times 1 \rightarrow$ squeezed to 300

return $\hat{\mathbf{p}}_s$

6.3. Evaluation Framework

All six models were evaluated under identical conditions using the following framework.

6.3.1. Dataset

The dataset comprises 93 athletic training sessions collected from multiple athletes across two sports: rowing and cycling. Each session contains heart rate measurements, ground truth boundary annotations derived from power output data, and session metadata, including sport type. Heart rate data was resampled to one measurement per second. Missing values introduced by resampling were imputed via forward fill followed by backward fill, replacing each gap with the nearest valid measurement in time.

The dataset was partitioned into training and test sets. The training set consists of 75 sessions (51 rowing and 24 cycling; approximately 279,000 timesteps and 1,400 boundaries), while the test set contains 18 sessions (12 rowing, 6 cycling) held out for final evaluation. The test set was stratified to ensure representation of both sports. For analysis purposes, sessions are additionally categorized by complexity: low ($\leq$10 boundaries), medium (11–20 boundaries), and high (>20 boundaries).

6.3.2. Performance Metrics

Model performance was evaluated using four complementary metrics.

F_β Score. We employ F_β scoring with $\beta = \sqrt{2}$, which weights recall twice as heavily as precision. This choice reflects the practical consideration that in training analysis, missing a true boundary is more costly than predicting a false one. Athletes and coaches can quickly dismiss an incorrect prediction, but a missed boundary is harder to recover. The F_β score is computed as:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \tag{6.1}$$

Precision and Recall. Standard precision (fraction of predicted boundaries that match ground truth) and recall (fraction of ground truth boundaries successfully detected) provide complementary perspectives on model behavior.

Mean Absolute Error (MAE). For each predicted boundary, we compute the temporal distance to the nearest ground truth boundary, regardless of whether it falls within the matching tolerance. This provides an unbounded measure of localization quality that additionally penalizes spurious predictions far from any true boundary, complementing the binary match/no-match evaluation of F_β.

6.3.3. Tolerance-Based Matching

A critical challenge in boundary detection evaluation is determining when a predicted boundary "matches" a ground truth boundary. Given the inherent physiological delay between changes in exercise intensity (reflected in power output) and the corresponding

heart rate response, exact temporal alignment between predicted and true boundaries cannot be expected. We therefore employ a tolerance window of ±10 seconds.

The evaluation uses a greedy matching strategy that prioritizes closest pairs first: all valid prediction–ground truth pairs within the tolerance window are ranked by temporal distance, and matches are assigned in order of proximity. Each prediction can match at most one ground truth boundary and vice versa, ensuring a strict one-to-one correspondence, illustrated by Figure 6.2. Without this constraint, a cluster of predictions near one boundary could yield multiple true positives for a single correct detection.

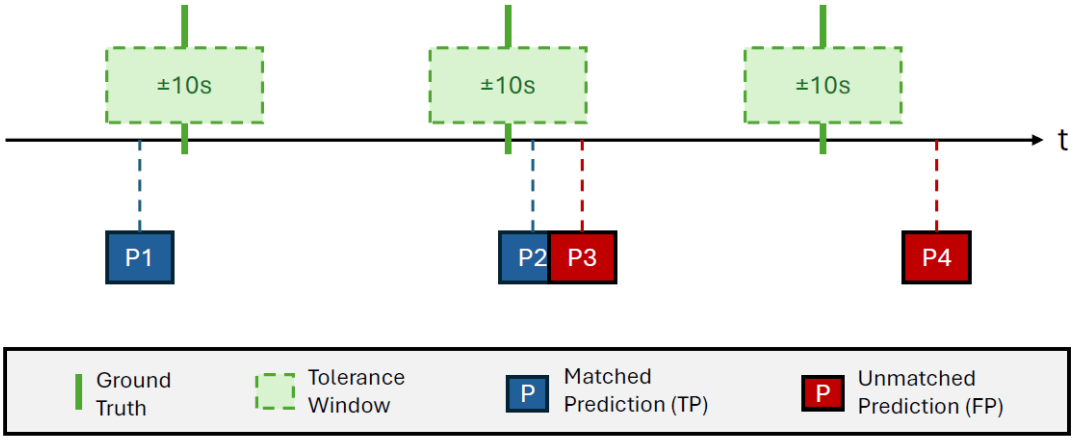

Figure 6.2.: Illustration of tolerance-based boundary matching. Ground truth boundaries (green) each define a ±10 s tolerance window. Predictions within a window are matched greedily by proximity: P1 and P2 are matched to their respective ground truth boundaries (TP), while P3 is rejected despite falling within GT2's window because P2 was closer (FP), and P4 lies outside all windows (FP).

6.3.4. Metric Equality Patterns

An important property of our evaluation setup is that precision, recall, and F_β are often mathematically identical for individual sessions. This arises from two design choices: our post-processing pipeline (Section 6.1.2) enforces exactly n predicted boundaries for sessions containing n true boundaries, and our matching strategy enforces one-to-one correspondence between predictions and ground truth boundaries. Together, every unmatched prediction necessarily implies an unmatched ground truth boundary, yielding:

$$\mathrm{FP} = n - \mathrm{TP} = \mathrm{FN}$$

which yields Precision = Recall = TP/n, and consequently F_β equals this same value.

7. Evaluation

7.1. Feature Set Ablation

To quantify the contribution of the full feature engineering effort, four of the five neural network architectures - CNN, TCN, BiLSTM, and U-Net - were evaluated in two configurations: a reduced 7-feature set and the full 39-feature set. The reduced set reflects the original motivation for these architectures, which are designed to learn temporal patterns directly from minimal input representations. The full feature set was subsequently applied to test whether explicit feature engineering could further improve their performance. XGBoost and the FFNN were evaluated only with the full feature set, as both models depend fundamentally on hand-crafted features: XGBoost by the nature of tree-based learning, and the FFNN by design, having been conceived as a feature-engineering-driven approach with its architecture optimized for the 39-dimensional input space.

This ablation serves two purposes. First, it quantifies how much of the models' performance is attributable to explicit feature engineering versus the heart rate and session-level context alone. Second, it reveals whether sequence architectures can compensate for reduced input representations through their own learned temporal processing.

Architecture	**Full F_β**	**Reduced F_β**	**Difference**	**Full / Reduced / Tie**
BiLSTM	0.72	0.53	−0.19	13 / 0 / 5
U-Net	0.67	0.52	−0.15	11 / 4 / 3
TCN	0.68	0.58	−0.10	9 / 4 / 5
CNN	0.68	0.63	−0.05	10 / 4 / 4

Table 7.1.: Impact of feature set reduction across architectures. The reduced set contains only 7 features (raw heart rate, session statistics, and workout structure), compared to the full set of 39 engineered features. "Full / Reduced / Tie" indicates the number of test sessions (out of 18) where each variant achieved a higher F_β score.

Across all four architectures, the full feature set consistently and substantially outperforms the reduced set (Table 7.1). The BiLSTM is most sensitive to feature reduction, losing 0.19 in F_β, with the full variant winning on 13 of 18 sessions and the reduced variant never winning a single session. The CNN is least affected (−0.05), suggesting that its convolutional filters can partially extract local temporal patterns even from the limited input.

Notably, none of the architectures can compensate for the absence of the engineered temporal features. This finding confirms that explicit feature engineering provides indispensable information that even sequence-aware architectures cannot reliably recover from raw heart rate data alone, at least at the current dataset scale. Consequently, the overall results table in Section 7.2 includes all ten configurations for completeness, while all further stratified analyses focus exclusively on the six full-feature configurations.

7.2. Comparative Results

All ten model configurations were evaluated on the same 18 held-out test sessions using the metrics and matching procedure described in Section 6.3. The results are presented in terms of overall performance, followed by stratification by sport, session complexity, and their combination. Visualisations of all 18 test sessions are provided in Appendix A.3.

7.2.1. Overall Performance

Model	F_β	**Precision**	**Recall**	**MAE (sec)**
BiLSTM	0.72	0.72	0.72	25.21
FFNN	0.71	0.71	0.71	23.92
XGBoost	0.71	0.71	0.71	20.23
TCN	0.68	0.68	0.68	24.47
CNN	0.68	0.68	0.68	23.30
U-Net	0.67	0.67	0.67	32.56
CNN (reduced)	0.63	0.63	0.63	44.87
TCN (reduced)	0.58	0.58	0.58	55.84
BiLSTM (reduced)	0.53	0.53	0.53	42.16
U-Net (reduced)	0.52	0.52	0.52	65.99

Table 7.2.: Overall performance of all ten model configurations, sorted by F_β score.

The ten model configurations fall into two distinct performance tiers (Table 7.2). The top tier comprises the six full-feature configurations - BiLSTM (0.72), FFNN (0.71), XGBoost (0.71), TCN (0.68), CNN (0.68), and U-Net (0.67) - spanning a range of 0.05 points. A clear gap separates these from the four reduced-feature variants, with F_β scores ranging from 0.52 to 0.63. Given the consistently lower performance of the reduced-feature variants, the remaining analyses in this section focus exclusively on the six full-feature configurations.

The three top-performing models - BiLSTM (0.72), FFNN (0.71), and XGBoost (0.71) - achieve closely comparable F_β scores. Since the neural network models are subject to minor variation across training runs due to stochastic optimization (random weight initialization, mini-batch sampling, and dropout), the differences among these three should

be interpreted with caution; their scores fall within a range where re-training could alter the ranking. XGBoost, as a deterministic algorithm with fixed random seeds, produces fully reproducible results and serves as a stable reference point.

Among the top three, XGBoost achieves the best mean absolute error (20.23 seconds), suggesting superior temporal precision when boundaries are correctly detected, followed by the FFNN (23.92 seconds). The TCN and CNN achieve identical F_β scores (0.68 each), forming a middle group with moderate MAE values (24.47 and 23.30 seconds respectively). The U-Net ranks sixth in F_β (0.67) and has a notably higher MAE (32.56 seconds), indicating that while it detects boundaries, their temporal placement is less precise.

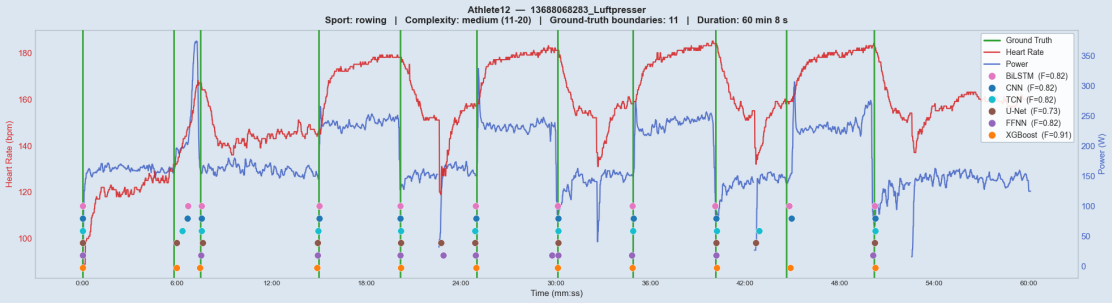

Figure 7.1.: Medium-complexity rowing session (11 boundaries, 60 min). While most models achieve high F_β scores (BiLSTM, CNN, TCN, FFNN: 0.82; XGBoost: 0.91), the U-Net scores notably lower (0.73). The horizontal spread of predicted dots around the ground-truth lines shows how MAE accumulates: even correctly matched boundaries contribute to temporal error when their placement is imprecise.

7.2.2. Performance by Sport

Sport	Metric	BiLSTM	FFNN	XGBoost	TCN	CNN	U-Net
Rowing (12)	F_β	0.74	0.72	**0.79**	0.72	0.74	0.70
cycling (6)	F_β	**0.69**	**0.69**	0.55	0.60	0.56	0.59
Sport gap (ΔF_β)		0.05	0.03	0.24	0.12	0.18	0.11

Table 7.3.: Performance of the six full-feature models by sport type. Bold indicates the best value in each row. The sport gap indicates the F_β difference between rowing and cycling for each model.

All models perform better on rowing than on cycling sessions (Table 7.3). For rowing, XGBoost leads clearly with an F_β of 0.79, followed by BiLSTM and CNN (both 0.74). For cycling, the BiLSTM and FFNN (both 0.69) outperform the remaining models, while XGBoost drops notably to 0.55. The FFNN exhibits the smallest sport gap (0.03), making

it the most consistent performer across sport types, while XGBoost shows the largest (0.24), indicating that its performance is strongly dependent on the structured patterns characteristic of rowing. It should be noted that approximately two thirds of all sessions are rowing, which may partly explain the consistent rowing advantage across models.

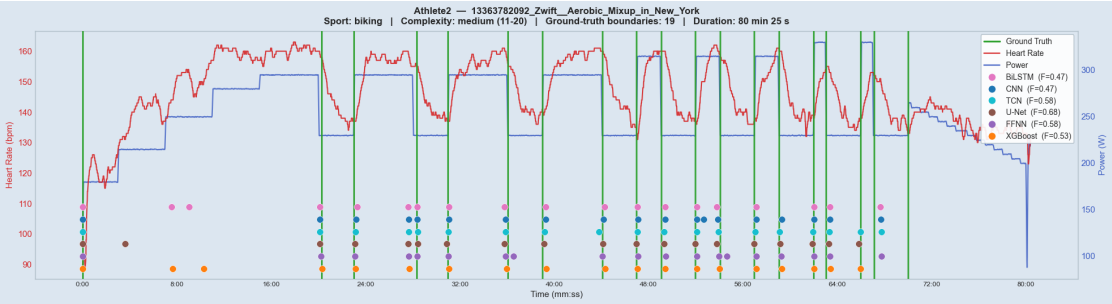

Figure 7.2.: Medium-complexity cycling session (19 boundaries, 80 min). Models diverge substantially, with BiLSTM and CNN scoring 0.47, and U-Net performing best at 0.68. The result contrasts with the medium-complexity rowing session in Figure 7.1, where all models exceed 0.92, illustrating the cross-sport performance gap reported in Table 7.3.

7.2.3. Performance by session complexity

Complexity	BiLSTM	FFNN	XGBoost	TCN	CNN	U-Net
Low, ≤10 (8)	0.74	0.70	**0.77**	0.69	0.71	0.71
Medium, 11–20 (6)	0.72	0.73	0.72	**0.74**	0.70	**0.74**
High, >20 (4)	0.70	**0.72**	0.55	0.57	0.59	0.48
Change (Low → High)	−0.04	+0.02	−0.22	−0.12	−0.12	−0.23

Table 7.4.: Performance of the six full-feature models by session complexity. Bold indicates the best value in each column. Number of test sessions per complexity level shown in parentheses. The final row shows the F_β change from low- to high-complexity sessions.

Session complexity, defined by the number of intervals per session (low: ≤10, medium: 11–20, high: >20), reveals the most pronounced performance differences between models (Table 7.4).

For low-complexity sessions (≤10 intervals), XGBoost achieves the best F_β (0.77). These sessions feature longer intervals with clearer heart rate transitions, conditions that may favor XGBoost's explicit feature-based decision boundaries.

For medium-complexity sessions (11–20 intervals), the TCN and U-Net lead jointly with an F_β of 0.74, closely followed by the FFNN (0.73).

For high-complexity sessions (>20 intervals), the FFNN achieves the best F_β (0.72), followed by the BiLSTM (0.70). XGBoost drops to 0.55, and the U-Net falls to 0.48.

The change from low to high complexity varies substantially across models. Notably, the FFNN is the only model to *improve* with complexity, gaining 0.02 points, suggesting it generalises well to denser interval structures. XGBoost and U-Net show the steepest declines (−0.22 and −0.23 respectively), driven by their strong performance on simple sessions and marked weakness on complex ones. The BiLSTM (−0.04) and TCN and CNN (both −0.12) show intermediate degradation.

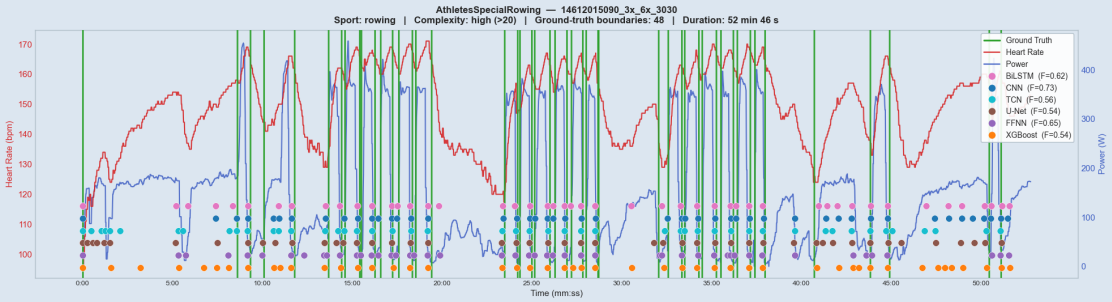

Figure 7.3.: High-complexity rowing session (48 boundaries, 53 min). The dense boundary structure leads to substantial performance degradation across all models compared to low- and medium-complexity sessions. Models diverge considerably, with CNN performing best (0.73) and XGBoost (0.54) and TCN (0.56) struggling most, consistent with the complexity-driven decline reported in Table 7.4.

7.2.4. Performance by session complexity x Sport

	Low (≤10)		Medium (11–20)		High (>20)	
Model	**Bike (1)**	**Row (7)**	**Bike (2)**	**Row (4)**	**Bike (3)**	**Row (1)**
BiLSTM	**0.75**	0.74	0.60	0.77	0.72	0.62
CNN	0.62	0.72	0.54	0.77	0.55	**0.73**
TCN	0.62	0.69	0.62	**0.80**	0.57	0.56
U-Net	**0.75**	0.70	**0.70**	0.75	0.46	0.54
FFNN	**0.75**	0.69	0.60	**0.80**	**0.74**	0.65
XGBoost	0.50	**0.81**	0.57	**0.80**	0.54	0.58

Table 7.5.: F_β scores broken down by complexity and sport. Session counts per subgroup are shown in parentheses. Note the small sample sizes, particularly for high-complexity cycling ($n = 3$) and high-complexity rowing ($n = 1$).

Table 7.5 further shows performance by both complexity and sport. The rowing advantage is most pronounced at low complexity, where XGBoost achieves an F_β of 0.81 on rowing

sessions but only 0.50 on the single cycling session. At high complexity, the FFNN (0.74) and BiLSTM (0.72) are the strongest performers on cycling, while CNN (0.73) leads on the single high-complexity rowing session. These patterns should be interpreted cautiously given the small subgroup sizes, particularly for high-complexity sessions ($n = 4$ total, with only one rowing session).

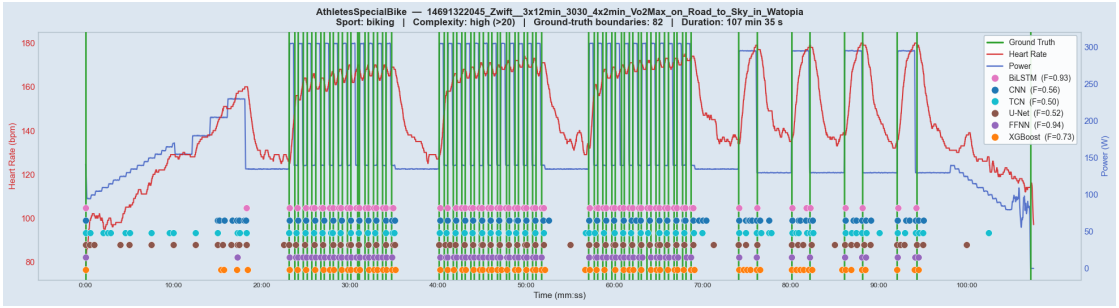

Figure 7.4.: High-complexity cycling session (82 boundaries, 108 min), the most boundary-dense session in the test set. FFNN (0.94) and BiLSTM (0.93) maintain strong performance despite the dense interval structure, while TCN (0.50) and U-Net (0.52) degrade substantially. The result directly illustrates the complexity × sport interaction reported in Table 7.5: high-complexity cycling represents the most challenging condition in the test set, and models differ markedly in their ability to generalise to it.

8. User-in-the-Loop Annotation Interface

8.1. Concept and Motivation

The models evaluated in Section 7.2 achieve F_β scores around 0.72 — informative enough to reduce annotation effort substantially, but not reliable enough for unsupervised deployment. The natural application is therefore as a set of candidate boundaries that a user can review and adjust, positioning the system as an annotation aid rather than a replacement for human judgement.

This motivates a User-in-the-Loop (UITL) approach, in which model predictions are surfaced as suggestions for a human reviewer. Rather than applied directly, each prediction can be accepted, dismissed, or corrected. The goal is to reduce manual annotation effort while preserving the reviewer's authority over the final output. To illustrate this concept, a prototype web-based interface was developed that visualises model predictions alongside the heart rate for each training session.[1]

8.2. Three-Tier Prediction Agreement System

The three top-performing models overall — BiLSTM, XGBoost, and FFNN — are selected as the ensemble for the annotation interface. Rather than selecting a single model, the interface combines all three by aggregating their predictions, using the degree of inter-model agreement to determine how each candidate boundary is handled.

Tier 1 — Full consensus (3/3 agreement). All boundary clusters supported by all three models are unconditionally accepted. No selection or ranking is applied: every 3/3 cluster represents a boundary the system is most confident about.

Tier 2 — Majority agreement (2/3 agreement). After Tier 1 is locked in, the remaining slots are filled from the pool of 2/3-agreement clusters. When the number of Tier 2 candidates exceeds the remaining slots, candidates are ranked by the average session-normalised output probability of their two contributing models, and only the highest-scoring are accepted (see Section 8.5).

[1] The prototype interface is publicly accessible at `https://kata-boehm.github.io/MockUpAirowHR/`.

Tier 3 — Best-model fallback. If Tier 1 and Tier 2 together still fall short of the expected boundary count, the interface falls back to the single model with the best empirical performance for the session type (see Section 8.4). Its predictions not already covered by an existing cluster are ranked by raw within-model output probability and the top candidates fill the remaining slots.

The reviewer may inspect all accepted boundaries and intervene where needed, dismissing individual predictions or adding new ones manually.

8.3. Adaptive Temporal Clustering

Raw model outputs are binary per-second labels. To group temporally close predictions from different models into a single boundary candidate, a temporal clustering step is applied: predictions are sorted chronologically, and a new cluster is seeded whenever a prediction falls further than a merge window w from the seed of every existing cluster; otherwise it is absorbed into the nearest cluster. Each cluster's seed is fixed as its first assigned prediction, while the reported centre, used as the final boundary candidate, is the mean of all member predictions, so the output position reflects the consensus of all contributing models. The merge window is set adaptively based on the number of expected boundaries (Table 8.1), following the rationale that predictions closer than the minimum possible separation between any two true boundaries must refer to the same boundary and should be merged.

Expected boundaries	**Merge window** w
< 10	30 sec
10–14	20 sec
15–19	15 sec
20–24	10 sec
25–34	8 sec
35–49	5 sec
≥ 50	3 sec

Table 8.1.: Adaptive merge window thresholds used during temporal clustering, determined by the number of expected interval boundaries.

8.4. Tier 3: Best-Model Routing as a Fallback

The fallback tier (Tier 3) requires designating one model as the authority when consensus fails. Since the ensemble is composed of the three overall top-performing models, the routing simply identifies which of the three achieves the highest F_β for the given sport and session complexity (Table 8.2).

Sport	Complexity	BiLSTM	FFNN	XGBoost	Best Model
Rowing	Low (≤10)	0.74	0.69	**0.81**	XGBoost
	Medium (11–20)	0.78	**0.80**	0.80	FFNN
	High (>20)	0.63	**0.65**	0.58	FFNN
Cycling	Low (≤10)	**0.75**	0.75	0.50	BiLSTM
	Medium (11–20)	**0.60**	0.60	0.57	BiLSTM
	High (>20)	0.72	**0.74**	0.55	FFNN

Table 8.2.: F_β scores of the three ensemble models by sport and session complexity. Bold indicates the best value per row; where values are equal at two decimal places, the model with the higher unrounded score was selected.

XGBoost is the strongest fallback for low-complexity rowing sessions ($F_\beta = 0.81$), where its feature-based decision boundaries are most effective. For medium- and high-complexity rowing, FFNN takes over, demonstrating superior performance as interval density increases. For cycling, BiLSTM serves as the fallback across low and medium complexity, while FFNN leads on high-complexity cycling sessions.

8.5. Candidate Selection and Ranking

When the number of boundary candidates within a tier exceeds the available slots, a ranking step is applied to select the most confident predictions. Tier 1 clusters are always accepted unconditionally, as full consensus already implies maximal confidence. For Tier 2, candidates are ranked by the average output probability of their two contributing models. Since raw probabilities are not directly comparable across models — a score of 0.7 from BiLSTM does not carry the same meaning as 0.7 from XGBoost due to differences in architecture, loss function, and regularisation — each model's probabilities are session-normalised via min-max scaling prior to averaging:

$$\tilde{p}_{m,t} = \frac{p_{m,t} - \min_s p_{m,s}}{\max_s p_{m,s} - \min_s p_{m,s}} \tag{8.1}$$

Candidates are then sorted descending by this score and the top $n = |\mathcal{B}_{\text{true}}| - |\mathcal{C}_{\text{tier1}}|$ are accepted. For Tier 3, within-model comparison is inherently valid, so candidates are ranked directly by raw output probability without normalisation.

8.6. Reviewer Workflow

The interface is structured around four tabs: an annotation view, a ground truth comparison view, a model overview tab, and a session summary. The annotation view constitutes the core reviewer-facing interface; the remaining tabs are included for demonstration and

evaluation purposes only, as they rely on ground truth annotations that would not be available in a real deployment scenario.

Upon loading a session, all Tier 1, Tier 2, and Tier 3 candidates are automatically accepted and pre-populated in the annotation view. The reviewer is presented with a progress indicator showing how many of the expected non-start boundaries have been confirmed, alongside a list of all suggested boundary candidates with their confidence tier, contributing models, and timestamp. Boundary markers in the heart rate chart are colour-coded by tier: green for Tier 1, orange for Tier 2, and yellow for Tier 3.

The reviewer can interact with individual candidates (accept or un-accept by clicking) or apply bulk operations by confidence tier using the tier-level toggle buttons. A reset action unaccepts all predictions, returning the session to a blank state with only the boundary suggestions visible. Manual boundaries can be added by clicking directly on the HR signal chart. Figure 8.1 shows an example session with all three tiers present.

The ground truth comparison tab provides a reference view placing raw model predictions alongside the true boundary annotations, supporting a qualitative assessment of where and how models disagree. The model overview tab summarises per-session F_β scores for all three individual models as well as the ensemble F_β computed from the UITL system's cluster centres against the true boundaries using a ± 10 seconds tolerance window and $\beta = \sqrt{2}$, matching the evaluation protocol of Section 7.2. Figure 8.2 shows an example of this comparison for the session depicted in Figure 8.1.

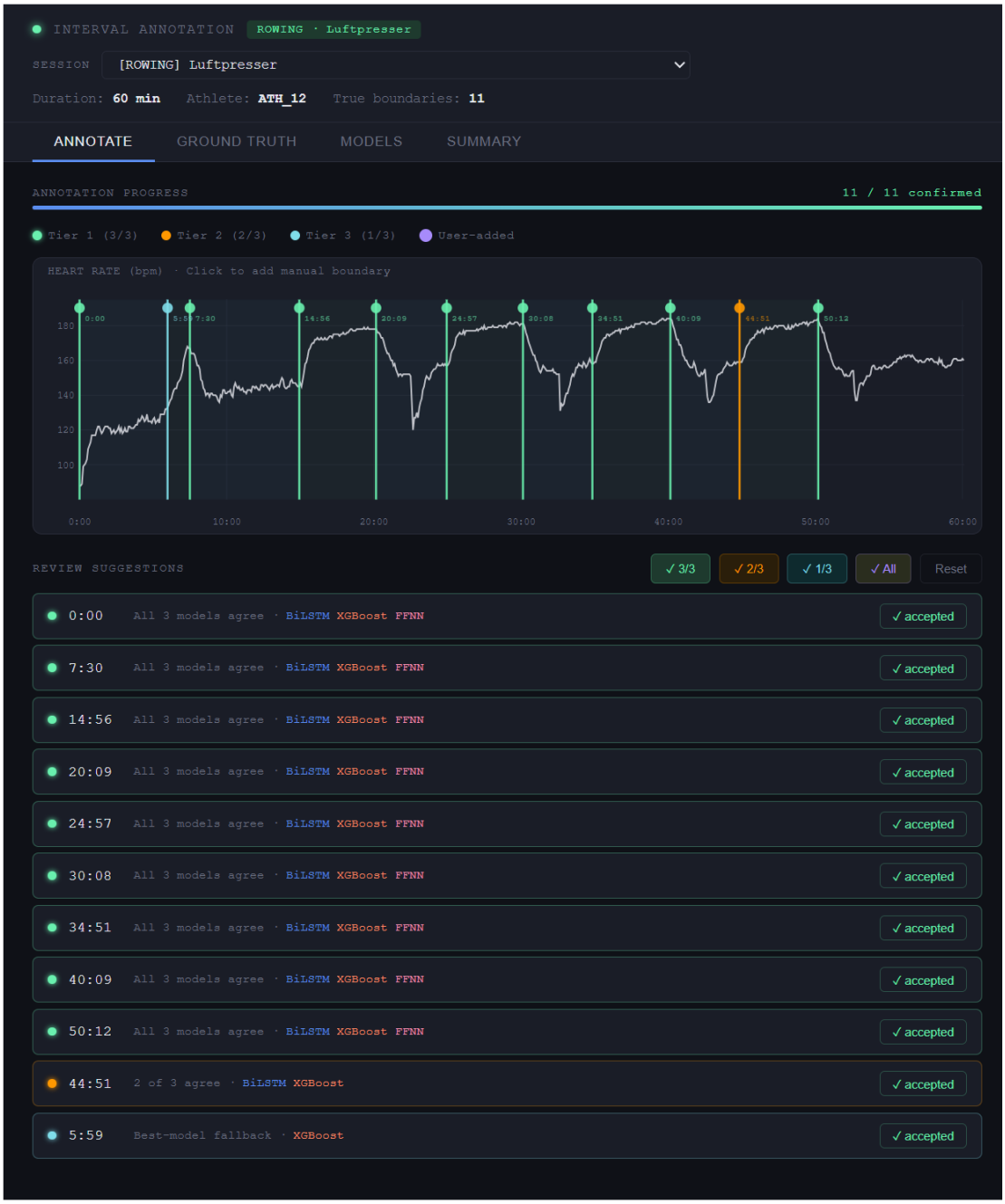

Figure 8.1.: Annotation view of the prototype UITL interface for a 60-minute rowing session. Boundary candidates are overlaid on the heart rate signal and colour-coded by confidence tier (green: Tier 1, orange: Tier 2, cyan: Tier 3). The candidate list below the chart shows each boundary's timestamp, contributing models, and acceptance status. Nine of eleven boundaries achieve full 3/3 consensus; one is supported by 2/3 models, and one is a best-model fallback.

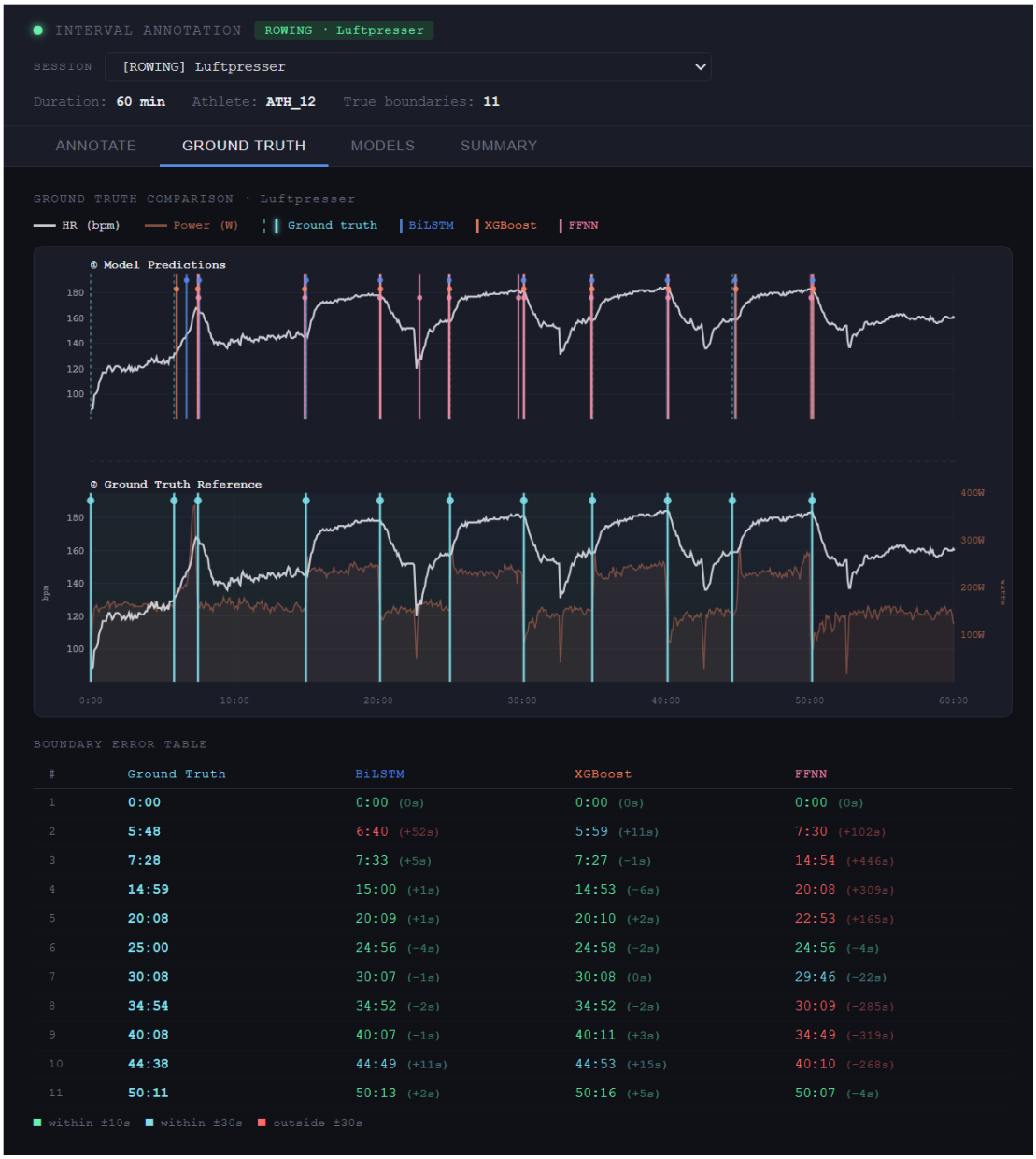

Figure 8.2.: Ground truth comparison tab for the same session, showing raw model predictions (BiLSTM, XGBoost, FFNN) alongside the true boundary annotations. This tab is included for demonstration purposes only; ground truth annotations would not be available in a real deployment scenario.

9. Discussion

The results demonstrate that heart rate data alone contains meaningful information for interval boundary detection, with the best models achieving F_β scores above 0.70. However, this level of performance is insufficient for fully automatic boundary detection in practice, as the models make too many errors to operate without human oversight.

The feature ablation confirms that explicit feature engineering is indispensable at the current dataset scale. All four tested architectures performed worse with only 7 features, and none could recover the discriminative information provided by the full 39-feature set — even architectures specifically designed to learn temporal representations from sequential input.

Beyond overall performance, the results reveal pronounced variation across session characteristics. All models performed better on rowing than on cycling, likely partly reflecting the imbalance in training data (51 rowing vs. 24 cycling sessions). Performance also degraded with session complexity for most models, with the notable exception of the FFNN, which was the only architecture to maintain stable performance across complexity levels. These findings suggest that no single architecture dominates universally, and that model behaviour is sensitive to the structural properties of the input.

This motivates the User-in-the-Loop approach explored in Chapter 8, where model predictions serve as an initial suggestion that users can review and correct, substantially reducing manual annotation effort while retaining human oversight over the final output.

9.1. Limitations

Several limitations of this work should be acknowledged when interpreting the results.

Dataset Size. With only 75 training sessions and 18 held-out test sessions, the results may not be statistically robust, and performance estimates across subgroups should be interpreted with caution. Collecting additional annotated sessions would be the most direct path to more reliable conclusions.

Annotation Subjectivity. The ground truth labels were produced by a single annotator, which introduces an inherent subjectivity bias. Boundary placement is not always unambiguous: the exact timing of a transition is a matter of judgment, and higher-level segmentation decisions, such as whether a cluster of short, closely spaced intervals should be treated as individual boundaries or as a single composite block, may reasonably differ

between annotators. Without multi-annotator agreement, this labeling subjectivity cannot be disentangled from model error. Furthermore, it remains an open question whether heart rate alone can ever reliably detect boundaries within such dense interval clusters, given the physiological lag and the limited recovery time between efforts.

Tolerance Window. The tolerance window of ± 10 seconds is a design choice motivated by the application context and the physiological lag of heart rate behind effort changes. Results should be interpreted within this evaluation framework rather than as absolute performance measures.

User-in-the-Loop Validation. The annotation tool and inter-model agreement strategy, while central to the proposed workflow, have not been evaluated with real users. The utility of the confidence tiers and the practical reduction in annotation effort remain to be validated in a user study. Furthermore, even with model assistance, interpreting and correcting boundary predictions may not be straightforward for users: heart rate data is physiologically lagged, and users without experience in reading heart rate traces may find it difficult to judge whether a suggested boundary is correctly placed.

9.2. Future Work

Several directions emerge naturally from the limitations identified above.

The most immediate priority is expanding the dataset, both in total volume and in balance across sport types and session complexities, which would allow more reliable subgroup comparisons and provide the architectural differentiation that the current scale cannot support.

A related goal would be to remove the dependency on the known interval count as an input feature: in the current implementation, the number of boundaries per session is provided to the post-processing pipeline as metadata, which is a practical constraint that limits deployment to contexts where this information is available in advance. With a sufficiently large and varied dataset spanning diverse athletes, sports, and session structures, it may become feasible to train models that can infer boundary count from the data itself, removing this requirement.

Multi-annotator labeling would address the subjectivity limitation identified above: Having multiple annotators independently label the same sessions would allow inter-annotator agreement to be measured, providing a more principled upper bound on achievable model performance. If human annotators themselves disagree on boundary placement in ambiguous cases, this sets a natural ceiling on what any model can reasonably be expected to achieve, and would help distinguish genuine model errors from inherently ambiguous cases.

Finally, a formal user study evaluating the annotation interface with real users would validate the practical utility of the User-in-the-Loop workflow and quantify the actual

reduction in annotation effort relative to fully manual labeling.

9.3. Conclusion

This thesis investigated the feasibility of automatic interval boundary detection in athletic training sessions using heart rate data, comparing six machine learning architectures across a dataset of 93 annotated rowing and cycling sessions. The results demonstrate that heart rate data contains sufficient information for meaningful boundary detection, with the best models achieving F_β score of 0.72. However, the results show that performance is not yet sufficient to justify fully automatic deployment without human oversight.

A key finding is that performance depends heavily on extensive feature engineering: all architectures benefited substantially from the 39 engineered features, and none could compensate for their absence. Beyond this, the results reveal that no single architecture dominates universally – model behaviour varies considerably across sport type and session complexity, with data imbalance likely contributing to these differences.

These findings motivated the design of a User-in-the-Loop workflow, in which model predictions serve as an initial suggestion that users can review and correct. The three-tier confidence system and prototype annotation interface developed as part of this thesis represent a practical path toward reducing manual annotation effort while retaining the flexibility to handle the cases that automated methods struggle with. Taken together, the results suggest that at the current dataset scale, the most productive role for these models is as an assistive tool rather than a replacement for human judgment.

10. AI Use Disclaimer

In the preparation of this thesis, AI-supported applications were used as auxiliary tools. Specifically, Claude (Anthropic) was employed in two capacities: first, as a coding assistant during the implementation of machine learning models and data processing pipelines, providing suggestions for code structure, debugging, and technical problem-solving; and second, to support the linguistic revision of draft texts, including rephrasing passages for clarity and readability. All AI-generated code suggestions were critically reviewed, tested, and adapted by the author. All AI-assisted text revisions were evaluated and contextualised by the author to ensure academic accuracy. The selection, design, and evaluation of methods, the analysis and interpretation of results, and the substantive content of this thesis were carried out independently by the author, who bears sole responsibility for all content.

Bibliography

[AIR22] AIROW Project. AIROW – Artificial Intelligence in Rowing: Online Brochure. `https://airow.univie.ac.at/fileadmin/user_upload/p_airow/Documents/airow_online_brochure_220902.pdf`, 2022. Research Network Data Science @ Uni Vienna, Institute of Sports Science, University of Vienna, and Austrian Rowing Federation. Accessed: 2026-03-01.

[And25] Stefano Andriolo. Interval IQ™: Automatic Interval Detection. `https://athletica.ai/interval-iq-automatic-interval-detection/`, 2025. Athletica.ai. Accessed: 2026-03-01.

[BFdST+15] Rhenan Bartels-Ferreira, Élder D. de Sousa, Gabriela A. Trevizani, Lilian P. Silva, Fábio Y. Nakamura, Cláudia L. M. Forjaz, Jorge Roberto P. Lima, and Tiago Peçanha. Can a first-order exponential decay model fit heart rate recovery after resistance exercise? *Clinical Physiology and Functional Imaging*, 35(2):98–103, 2015.

[BHL+20] Hans Bühler, Blanka Horvath, Terry Lyons, Imanol Perez Arribas, and Ben Wood. A data-driven market simulator for small data environments, 2020.

[BKK18] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[CJL+23] Adam M. Coates, Michael J. Joyner, Jonathan P. Little, Andrew M. Jones, and Martin J. Gibala. A perspective on high-intensity interval training for performance and health. *Sports Medicine*, 53(S1):85–96, 2023.

[GA18] Damien Garreau and Sylvain Arlot. Consistent change-point detection with kernels. *Electronic Journal of Statistics*, 12(2):4440 – 4486, 2018.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GS05] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005. IJCNN 2005.

[Han03] David J. Hand. *Introduction*, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[Int26] Intervals.icu Ltd. Intervals.icu – Free Cycling, Running, Triathlon Training Platform. `https://www.intervals.icu/`, 2026. Accessed: 2026-03-01.

[JF15] In Cheol Jeong and Joseph Finkelstein. Comparative utility of time and frequency hrv domains for automated classification of exercise exertion levels. In *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 983–989, 2015.

[KAA+21] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398, 2021.

[KW19] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 2019.

[RF20] Alen Rajšp and Iztok Fister. A systematic literature review of intelligent data analysis methods for smart sport training. *Applied Sciences*, 10(9), 2020.

[RRKF25] Alen Rajšp, Patrik Rek, Peter Kokol, and Iztok Fister. The role of intelligent data analysis in selected endurance sports: A systematic literature review. *Applied Sciences*, 15(18), 2025.

[RSS+21] Sofia Romagnoli, Agnese Sbrollini, Alessio Scalese, Ilaria Marcantoni, Micaela Morettini, and Laura Burattini. Signal processing for athletic cardiovascular monitoring with wearable sensors: Fully automatic detection of training phases from heart rate data. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1491–1494, 2021.

[SED18] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation, 2018.

[SHD+21] Amine Souissi, Monoem Haddad, Ismail Dergaa, Helmi Ben Saad, and Karim Chamari. A new perspective on cardiovascular drift during prolonged exercise. *Life Sciences*, 287:120109, 2021.

[Vek25] Vekta. Intervals, session & race classification, 2025. Accessed: 2026-04-17.

[VPP+07] M. Vaglio, A. Porta, P. Pizzinelli, S. Di Marco, D. Lucini, F. Badilini, and M. Pagani. A fully automatic algorithm for the analysis of heart rate changes and cardiac recovery during exercise. In *2007 Computers in Cardiology*, pages 309–312, 2007.

A. Appendix

A.1. Code Availability

The code developed for this thesis is publicly available in the following repositories:

Power-Based Annotation Tool The web application used to label the training data via power visualization: `https://github.com/kata-boehm/AirowPowerAnnotator`. The deployed tool is accessible at `https://airow-smm5.onrender.com/`.

Model Implementations All ten model architectures, the feature engineering pipeline, and the evaluation framework: `https://github.com/kata-boehm/airow-model-implementations`

Annotation Interface The prototype User-in-the-Loop annotation interface (MockUpAirowHR): `https://github.com/kata-boehm/MockUpAirowHR`. The deployed interface is accessible at `https://kata-boehm.github.io/MockUpAirowHR/`.

A.2. Preliminary Project Results

The following summarizes the evaluation setup and results of the preliminary project described in Section 3.3.

Predicted boundaries were matched against ground truth starts within a tolerance window of ± 5 seconds. Performance was measured using the F_β-score with $\beta = \sqrt{2}$, consistent with the evaluation framework of this thesis, alongside Precision and Recall. The dataset comprised 71 sessions in total: 60 training sessions (52 rowing, 8 cycling) and 11 test sessions (9 rowing, 2 cycling).

The heuristic threshold model and the CNN determined the number of intervals alone, without this information being provided as input. The kernel CPD, by contrast, was provided the ground truth number of intervals as input, representing an informational advantage not available to the other two models.

Model	Precision	Recall	F_β-Score
Heuristic Threshold	0.59	0.84	0.74
Kernel CPD	0.53	0.53	0.53
CNN	0.06	0.12	0.09

Table A.1.: Performance of power-based models on the test set (5-second tolerance window, $\beta = \sqrt{2}$).

A.3. Test Session Visualisations

All 18 held-out test sessions are shown below. Each plot displays the heart rate signal (red), power output where available (blue), ground-truth interval boundaries (green vertical lines), and per-model predicted boundary positions (coloured dots). Session metadata and per-model F_β scores are shown in the plot title and legend.

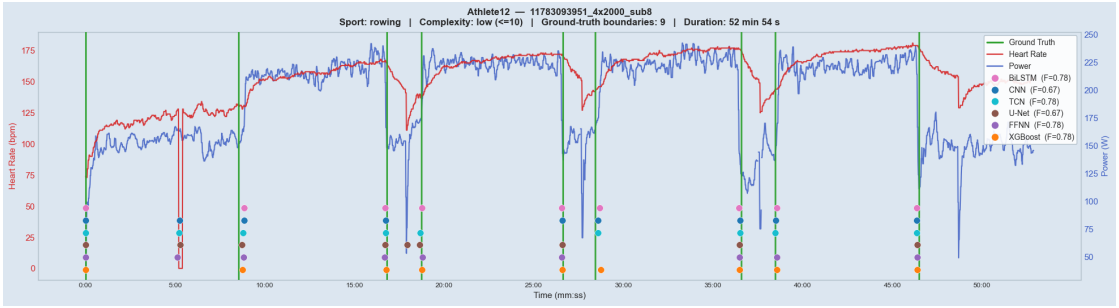

Figure A.1.: Low-complexity rowing, 9 boundaries, 53 min.

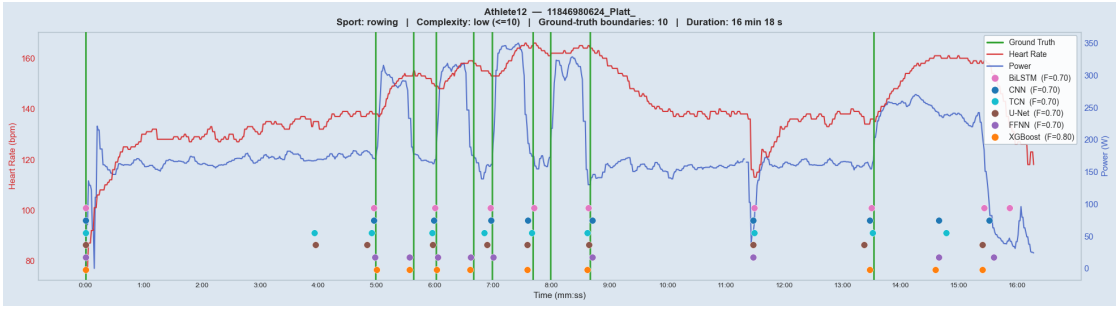

Figure A.2.: Low-complexity rowing, 10 boundaries, 16 min.

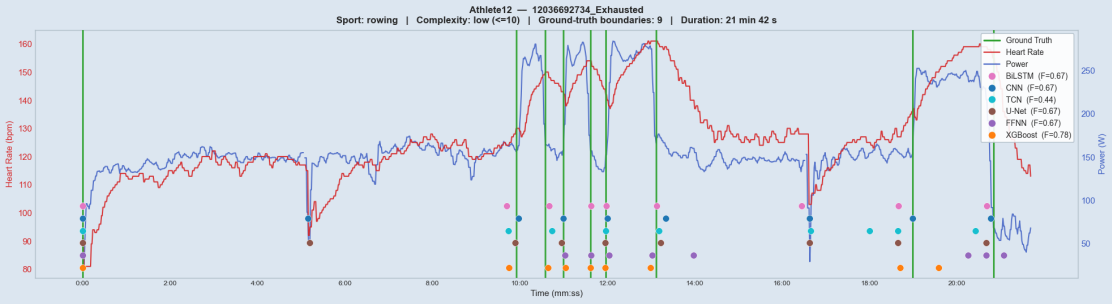

Figure A.3.: Low-complexity rowing, 9 boundaries, 22 min.

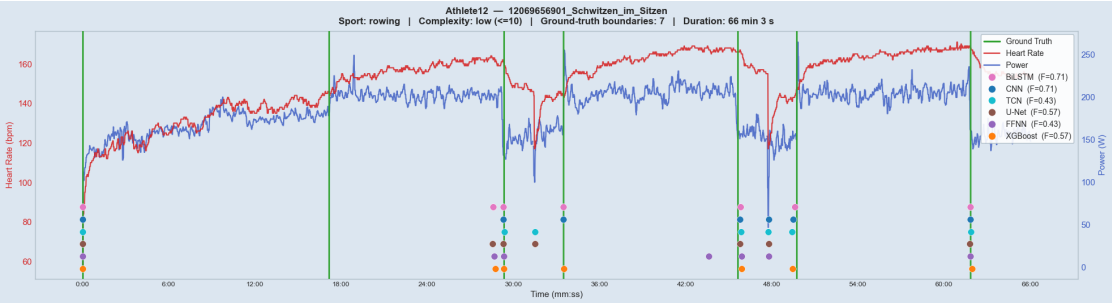

Figure A.4.: Low-complexity rowing, 7 boundaries, 66 min.

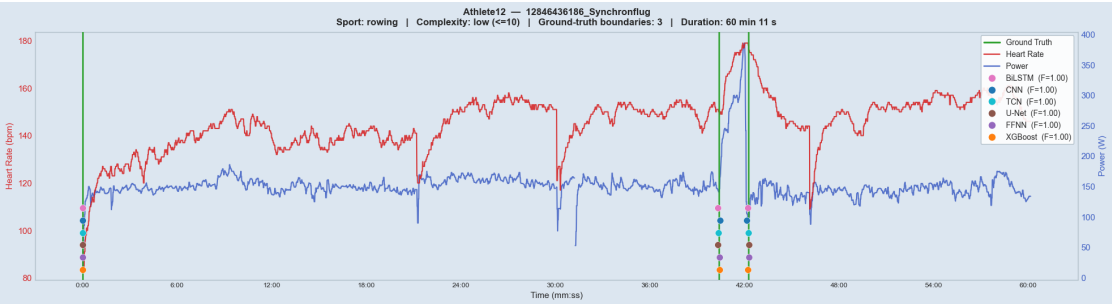

Figure A.5.: Low-complexity rowing, 3 boundaries, 60 min.

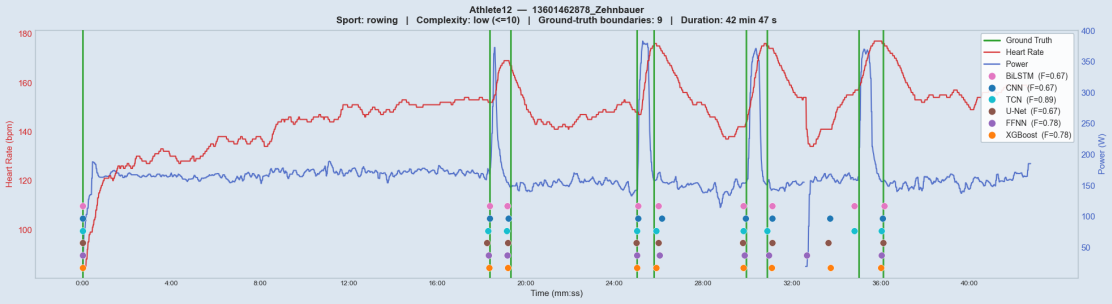

Figure A.6.: Low-complexity rowing, 9 boundaries, 43 min.

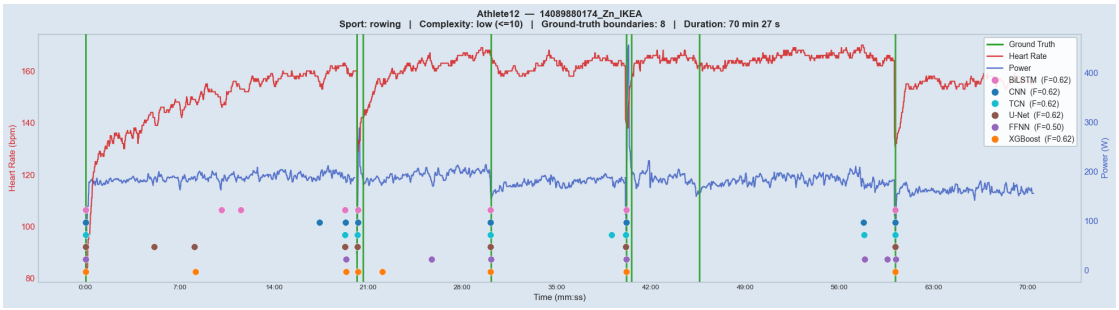

Figure A.7.: Low-complexity rowing, 8 boundaries, 70 min.

Figure A.8.: Medium-complexity rowing, 12 boundaries, 59 min.

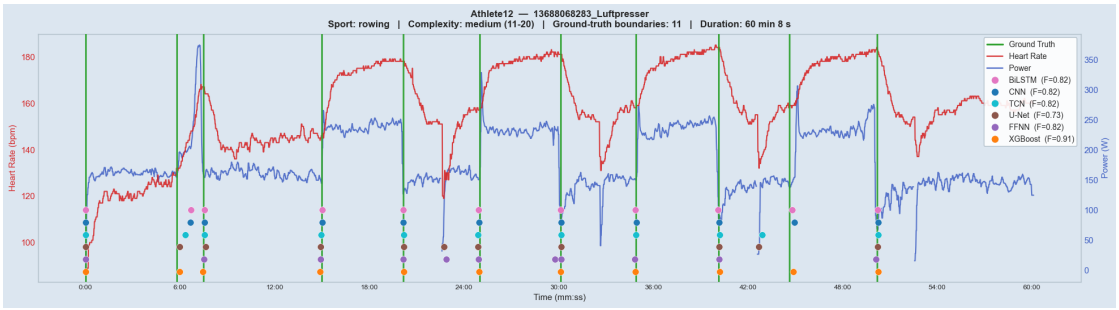

Figure A.9.: Medium-complexity rowing, 11 boundaries, 60 min.

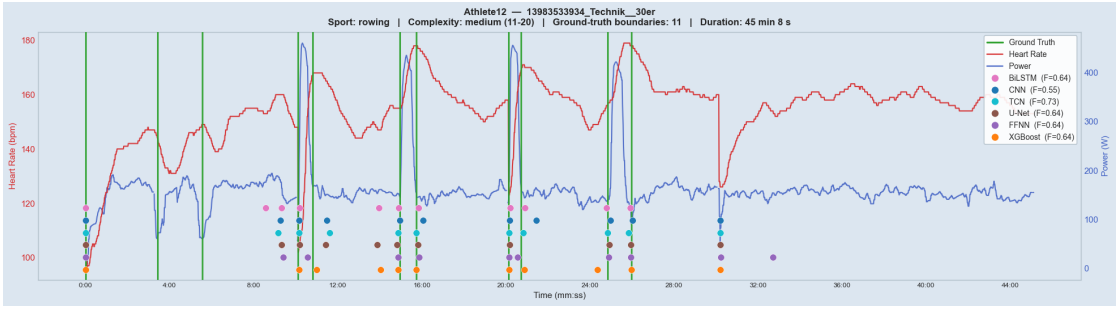

Figure A.10.: Medium-complexity rowing, 11 boundaries, 45 min.

Figure A.11.: Medium-complexity rowing, 11 boundaries, 62 min.

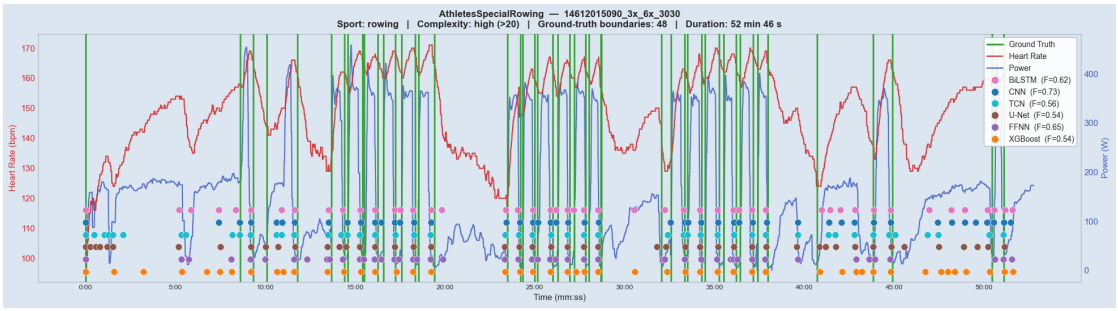

Figure A.12.: High-complexity rowing, 48 boundaries, 53 min.

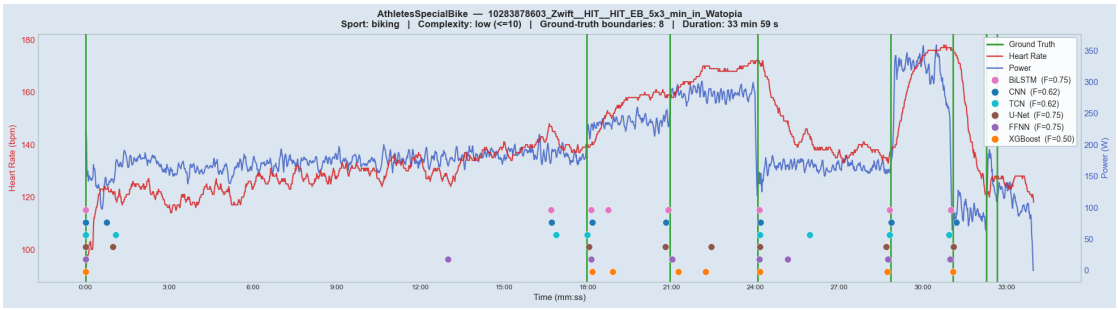

Figure A.13.: Low-complexity cycling, 8 boundaries, 34 min.

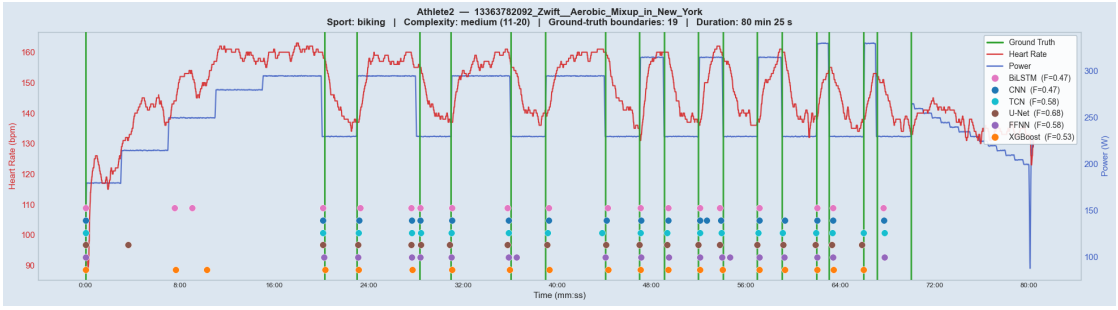

Figure A.14.: Medium-complexity cycling, 19 boundaries, 80 min.

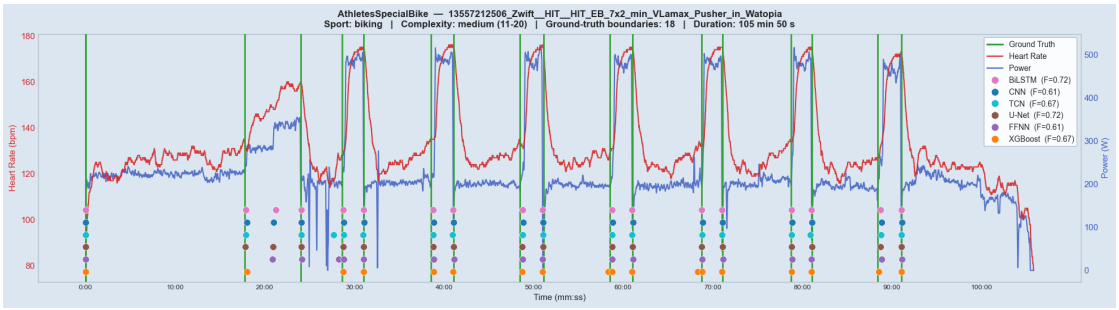

Figure A.15.: Medium-complexity cycling, 18 boundaries, 106 min.

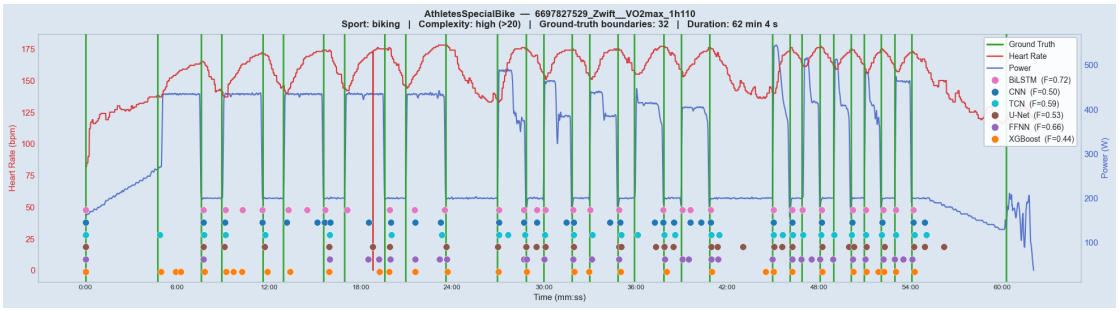

Figure A.16.: High-complexity cycling, 32 boundaries, 62 min.

Figure A.17.: High-complexity cycling, 40 boundaries, 115 min.

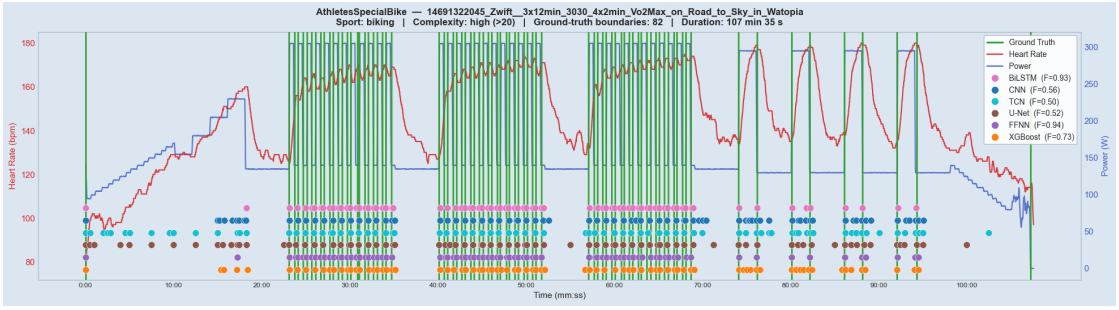

Figure A.18.: High-complexity cycling, 82 boundaries, 108 min.