Multi-Task Deep Learning Approach for Segmenting and Classifying Competitive Swimming Activities Using a Single IMU

By

Mark Shperkin

Bachelor Of Science
University of South Carolina, 2024

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Computer Science

Molinaroli College of Engineering and Computing

University of South Carolina

2025

DEDICATION

First and foremost, I thank my parents, Leonid, and Elnora Shperkin, for their unwavering support—waking before dawn to drive me to swim practices, encouraging both my athletic and academic pursuits, and enabling me to study abroad. I am also grateful to the coaches who shaped me into the competitive, dedicated person I am today: Leonid Sheichat, Yamit Libster, Dima Ribinsky, Mike Simpson, Moody McGee, Kevin Swander, Duncan Sherard, and Robert Pinter. My thanks extend to the University of South Carolina coaching staff—Jeff Poppell, Jason Calanog and Nils Wich-Glasen —for facilitating and validating this research with the Gamecocks Swim & Dive Team. I owe special appreciation to my academic advisor, Professor Marco Valtorta, for his guidance throughout my graduate studies, and to my research advisor, Professor Homayoun Valafar, for believing in me and supporting this thesis. I dedicate this work to all of you.

ABSTRACT

Competitive swimming performance analysis has traditionally relied on manual video review and multi-sensor systems, both of which are resource-intensive and impractical for everyday training use. This study investigates whether a single wrist-worn inertial measurement unit (IMU) can be used to automatically segment and classify swimming activities with high accuracy. We propose a multi-task deep learning pipeline based on the MTHARS (Multi-Task Human Activity Recognition and Segmentation) architecture introduced by Duan et al. to perform stroke classification, lap segmentation, stroke count estimation, and underwater kick count estimation. Data were collected from eleven collegiate-level swimmers wearing left-wrist-mounted IMUs, each performing five 100-yard sets per stroke (butterfly, backstroke, breaststroke, freestyle, and individual medley) in a 25-yard pool. This pipeline delivers a reliable multi-metric evaluation while significantly reducing the complexity and cost of sensor setups. In leave-one-subject-out validation, the accelerometer-only model achieved a micro-F_1 of 0.7405 (macro-F_1 0.5894), which improved to 0.7709 (macro-F_1 0.6565) when gyroscope data were added. This work contributes to the growing field of wearable-based athlete monitoring and has the potential to empower coaches and athletes with real-time, fine-grained performance feedback in competitive swimming using minimal hardware.

TABLE OF CONTENTS

Dedication iii

Abstract iv

List of Tables vii

List of Figures viii

Chapter 1: Introduction 1

Chapter 2: Review of Literature 3

Chapter 3: Dataset 8

- 3.1 Hardware Configuration 8
- 3.2 Protocol 8
- 3.3 Labeling & Annotations 8
- 3.4 Preprocessing 14

Chapter 4: Approach 16

- 4.1 MTHARS Methodology 16
- 4.2 Train Procedure 20
- 4.3 Training Hyperparameters 24

Chapter 5: Experiments and Results 25

- 5.1 Non-Maximum Suppression Algorithm 25
- 5.2 Evaluation Metrics 26
- 5.3 IoU Threshold Tuning 27
- 5.4 Hard Negative Mining Tuning 27
- 5.5 Loss Weights Balancing 27

5.6 Scale Factor Tuning 28
5.7 Input Study 28
Chapter 6: Conclusion 32
Chapter 7: Future Work 34
References 36

LIST OF TABLES

Table 4.1 Overview of the MTHARS Architecture Parameters (kernel size / stride / padding / dilation) 19

Table 5.1 Effects of τ IoU threshold on Mean IoU, Macro-$F1$, and Micro-$F1$ 27

Table 5.2 Effects of R ratio on Mean IoU, Macro-$F1$, and Micro-$F1$ 27

Table 5.3 Effects of α, β weights on Mean IoU, Macro-$F1$, and Micro-$F1$ 28

Table 5.4 Effect of scales s on Mean IoU, Macro-$F1$, and Micro-$F1$ 28

Table 5.5 Effects of using accelerometer vs. accelerometer & gyroscope as input data on Mean IoU, Macro-$F1$, Micro-$F1$, stroke and kick counts Mean Absolute Error (MAE) 28

Table 5.6 $F1$ score of all classes for accelerometer vs. accelerometer & gyroscope input data 29

Figure 5.1 Confusion matrix of only accelerometer input data 29

Figure 5.2 Confusion matrix of accelerometer & gyroscope input data 30

Figure 5.3 Training loss curve of accelerometer vs. accelerometer & gyroscope models showing steady convergence 30

Figure 5.4 Training and validation accuracy and IoU for accelerometer vs. accelerometer & gyroscope 31

LIST OF FIGURES

Figure 3.1 Example of raw tri-axial accelerometer and gyroscope signals during 100-yard individual medley set. 9

Figure 3.2 Example of accelerometer data stream of inter swimmer variability. (orange-butterfly; blue-backstroke; green-breaststroke; red-freestyle). Shaded regions indicate the window used for segmentation. 10

Figure 3.3 Example of accelerometer data stream of turn variability. Panels (a)–(d) show somersault, open-turn left, open-turn right, and backstroke-to-breaststroke-turn, respectively. Shaded regions indicate the window used for segmentation. 11

Figure 3.4 Example of accelerometer data stream of (a) push-off on stomach vs. (b) on back. Shaded regions indicate the window used for segmentation. 12

Figure 3.5 Example of accelerometer data stream of underwater kick during (1) laying on the back vs. (2) laying on the stomach, athlete who has (a) stable arms vs. athlete who uses his (b) arms for momentum. Shaded regions indicate the window used for segmentation. 12

Figure 3.6 Example of accelerometer data stream of (a) left hand touch vs. (b) right hand touch vs (c) both hand touch. Shaded regions indicate the window used for segmentation. 13

Table 3.1 Total count and average duration per class in the train and validation datasets. 14

Chapter 1: Introduction

Sensor-based activity recognition has rapidly matured over the past decade as a scalable, privacy-friendly alternative to camera-based methods [1]. In competitive swimming, however, practitioners still depend on manual video analysis, expensive pool-embedded timing systems, or coach-provided feedback to measure split times, stroke counts, and physiological load. These approaches are labor-intensive, non-portable, and impractical for monitoring multiple athletes simultaneously. Wrist-worn inertial measurement units (IMUs) offer a low-cost, mobile alternative, but to date no study has demonstrated an end-to-end solution that automatically detects lap boundaries, segments and classifies individual stroke cycles, and counts strokes and underwater kicks.

The objective of this research is to design and evaluate a multi-scale feature-extraction pipeline for single-IMU swimming data. We systematically tune and compare hyperparameters to achieve high segmentation and classification accuracy. Our approach is developed and validated on a dataset of eleven collegiate swimmers wearing wrist-mounted IMUs, recording triaxial accelerometer and gyroscope data at 52.63 Hz in a 25-yard pool. This pipeline presents an affordable, widely accessible method for collecting, analyzing, and evaluating high-level swimming performance. By leveraging devices many athletes already wear like smartwatches, our solution can deliver data-driven insights into daily workouts, support performance improvement, and help prevent overtraining or injury—thereby raising the overall standard of competitive swimming.

Despite their promise, wearable-based solutions face significant technical hurdles. Stroke cycles generate rich, high-frequency motion signals, whereas lap transitions such as turns and wall push-offs are sparse and vary by stroke. Reliable detection and segmentation of these events must contend with noisy sensor readings due to water resistance, sensor drift, and body rotation. Moreover, most existing IMU-based approaches either target a single task (e.g., activity classification) or rely on multiple sensors placed on different body locations, increasing user burden and system complexity. There remains a critical gap in the development of a lightweight, single-IMU solution capable of simultaneously classifying swimming activities and counting strokes and kick across all competitive strokes with high accuracy. Addressing this gap would enable scalable, real-time swim monitoring and personalized feedback using minimal hardware.

CHAPTER 2: REVIEW OF LITERATURE

Human activity recognition is one of the most promising research topics for a variety of areas of real-life applications, including ambient assisted living (AAL) where continuous, real-time monitoring of activities of daily living (ADLs) in smart-home [2-4] and institutional care setting enables personalized, context aware assistance of the elderly [5]. Human–computer interaction using sensor data [6, 7] advanced the field with real-time activity recognition capabilities. Context-aware computing [8], employing passive infrared motion detectors [9], reduces user intervention by inferring context from low-cost, privacy-preserving sensors and delivers scalable, robust activity recognition. Human-centric, complex activity problems in healthcare are tackled in [10, 11], where probabilistic sequence models (e.g., HMMs and CRFs) and unsupervised pattern-discovery and tracking methods enable continuous health monitoring and personalized assistance for individuals experiencing difficulties living independently.

Classical machine learning algorithms, such as Support Vector Machines (SVM [12], K-Nearest Neighbors (KNN) [13], and Random Forests [14], leverage handcrafted time and frequency domain features to characterize sensor data streams and have consistently demonstrated robust activity classification performance. However, handcrafted feature extraction demands extensive domain expertise and often fails to generalize across different activities.

To address this, recent deep learning methods automatically learn hierarchical features directly from raw sensor streams. In particular, convolutional neural networks

(CNNs) [15, 16] have demonstrated superior performance over traditional classifiers such as SVM by extracting robust temporal and spatial patterns in an end-to-end manner. Müller et al. [17] adapt three existing CNN architectures and introduce a Scaling-FCN classification head—alongside standard FCN variants—to evaluate how different convolutional feature extractors impact IMU-based fitness activity recognition. Zhao et al. [18] further extend this line of work by proposing a parallel CNN architecture that ingests both DWT and STFT representations in dual convolutional streams, achieving near-perfect precision in classifying a diverse set of sports activities using only minimal wearable sensor inputs.

With the introduction of the LSTM (Long Short-Term Memory) architecture [19], researchers were able to capture long-range temporal dependencies in IMU streams, leading to significant improvements in human activity recognition. Subsequent studies expanded upon the basic LSTM cell by stacking multiple LSTM layers and employing bidirectional and cascaded deep recurrent structures [20], leveraging ensembles of diverse deep LSTM learners to further improve recognition robustness and performance [21], and combining CNN-based feature extraction front ends with LSTM sequence models in hybrid architectures tuned via Bayesian optimization for smartphone-based HAR [22]. Vakacherla et al. [23] showed that combining CNNs with LSTMs to a hybrid CNN-LSTM architecture can achieve over 98% accuracy in five daily activities recorded via a single chest-mounted accelerometer. Khatun et al. [24] further enhance a CNN-LSTM framework by integrating a self-attention mechanism, achieving 99.93% accuracy on H-Activity, 98.76% on MHEALTH, and 93.11 % on UCI-HAR benchmarks.

Further improvements in RNN-based (Recurrent Neural Network) HAR leveraged hybrid GRU–CNN (Gated Recurrent Unit) architectures. DeepSense integrates [25] CNN modules for local feature extraction with stacked GRU layers to model temporal dynamics on mobile devices, AttnSense [26] augments this fusion with multi-level attention mechanisms over both modality-specific CNN outputs and GRU hidden states to prioritize salient spatial–temporal cues, and recent work using a stacking ensemble of 1D-CNN heads alongside a GRU head has achieved state-of-the-art accuracy in clinical balance assessment with single-IMU data [27].

The Transformer architecture [28], who is built entirely on self-attention rather than recurrence, has revolutionized sequence modeling and inspired application across diverse domains, including human activity recognition. By processing raw IMU streams through self-attention layers to learn spatial-temporal representations without recurrence, [29] outperform earlier CNN-RNN hybrids on multiple HAR benchmarks. Later works optimized Transformer configurations with modality-specific attention modules and efficient positional encoding, enabling mobile-friendly, real-time inference with accuracy exceeding 99% on diverse activity recognition tasks [30, 31]. Guo et al. [32] further enhance the Transformer for IMU-based HAR by integrating a convolutional feature-extractor block and a vector-based relative position embedding, yielding state-of-the-art accuracy across multiple benchmark datasets.

Recent work in swimming activity recognition has shown that sliding-window segmentation paired with both traditional and deep learning architectures can robustly distinguish continuous stroke cycles and sparse transitional events using only wearable IMUs. Delhaye et al. [33] segment sacrum-mounted IMU data into 90-frame (1.8 s) windows before passing each through a deep Bi-LSTM stack to predict eight classes—four strokes, wall pushes, turns, underwater phases, and rest—with an overall F1-score of 0.96 and lap-time MAPE below 4.1%. Zhang et al. [34] employ a 150 ms sliding window with 10 ms stride on six lower-limb IMUs, extract time-domain features (mean, standard deviation, extrema, sum of absolute differences), and classify four strokes via a stroke-dependent quadratic discriminant analysis model, achieving per-stroke accuracies from 97.24% to 99.10%. Building on these approaches, Chen and Hu [35] introduce a hybrid DCNN-BiLSTM architecture: ten body-worn IMUs are windowed into 200-sample segments with 50 % overlap at 200Hz, convolved to learn hierarchical features, and then sequenced through bidirectional LSTM layers—yielding balanced accuracies above 92% even when reduced to just two sensors. These studies collectively demonstrate that carefully chosen window lengths and overlaps, combined with classifiers ranging from QDA to end-to-end deep networks, can accurately segment and classify swimming activities. However, although sliding-window approaches yield promising classification performance, they cannot reliably estimate the number of activity repetitions, since the true start and end points of each swim action remain unknown.

Recent advances move beyond fixed-length windowing to directly infer true activity segments. Aminikhanghahi et al. [36] frame segmentation as an online change-point detection task: by computing a kernel-based density-ratio divergence between consecutive, short windows, their SEP algorithm flags real change points whenever the divergence exceeds a learned threshold, yielding precise segment boundaries in near real time. Li et al.'s P2LHAP [37] abandons sliding windows altogether, instead splitting each sensor channel into overlapping patches and feeding them through a Seq2Seq

Transformer to predict patch-level activity labels. Adjacent patches with identical labels are then merged to recover true start–end segments, unifying segmentation, and recognition in one end-to-end model.

CHAPTER 3: DATASET

3.1 Hardware Configuration

Triaxial accelerometer and gyroscope data were collected using a single wrist-worn smartwatch (Tic Watch Pro 5). The device was secured just proximal to the distal wrist crease on the left arm to minimize movement artifacts. Sensor streams were recorded at 52.63Hz and synchronized with a video recording for manual annotations.

3.2 Protocol

Eleven collegiate swimmers (ages 18-30) from the University of South Carolina Swim and Dive team completed the data-collection sessions. Each athlete performed five 100-yard sets, one per stroke (freestyle, backstroke, breaststroke, butterfly, and individual medley), at a comfortable moderate pace with approximately 30 seconds rest between sets. Participants were instructed to use their normal underwater-kick routine and to maintain consistent stroke technique. During initial data transfer, a small number of recordings were corrupted, so we conducted additional stroke-specific sessions with a subset of athletes to recover those data.

3.3 Labeling & Annotations

We defined nine hand-crafted activity labels—freestyle, backstroke, breaststroke, butterfly, underwater glide, underwater kick, push-off, turn, and wall touch—to capture all key phases of a swim session. Label definitions and segment boundaries were refined to reduce ambiguity and ensure as much as possible uniform data signal across all activities.

Examining the data from a single swimmer reveals highly uniform activity patterns (Figure 3.1), making model training on that individual essentially trivial. In contrast, comparing two athletes performing the same swim set uncovers substantial variability across all activity types. To ensure our model generalizes well, we recruited as many athletes as possible and evaluated its performance using a leave-one-subject-out cross-validation, with the author serving as the held-out participant. The following subsections explore these inter-athlete variations in detail.

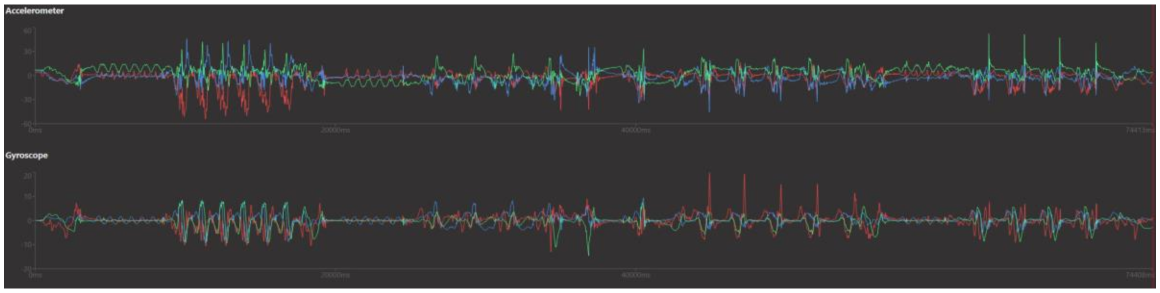


Figure 3.1 Example of raw tri-axial accelerometer and gyroscope signals during 100-yard individual medley set.

3.3.1 Swim Strokes

Because the smartwatch was mounted on the left wrist, strokes in which the arms move asynchronously (freestyle and backstroke) were defined with respect to left-hand motion: each stroke segment begins the instant the left hand initiates its pull phase and ends when it returns to its recovery position. This definition accommodates inter-swimmer differences in lead-hand preference, although it can introduce variability when athletes begin or end a cycle with their right hand. In contrast, synchronous strokes (butterfly and breaststroke) exhibit consistent bilateral motion, where segment boundaries are inherently more constant across swimmers. Figure 3.2 illustrates the segmentation intervals for each stroke style and overlays the sensor signals from two athletes across all four strokes. Although the temporal patterns closely align, the signal amplitudes differ noticeably between athletes.

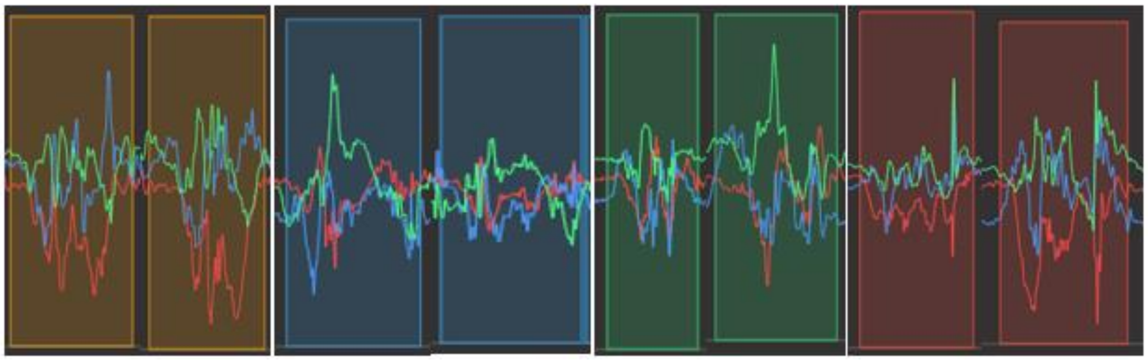


Figure 3.2 Example of accelerometer data stream of inter swimmer variability. (orange-butterfly; blue-backstroke; green-breaststroke; red-freestyle). Shaded regions indicate the window used for segmentation.

3.3.2 TURNS

Turn segments encompass all wall-interaction maneuvers performed between laps. The mechanics of each turn are heavily influenced by the stroke being swam and the sequence of strokes before and after the wall encounter. Swimming regulations specify distinct turn requirements for each stroke and stroke transition, which introduces both ambiguity and wide variability in our turn labels (Figure 3.3). The following sections describe the different turn activities that our system classifies.

Freestyle & Backstroke: Flip Turns

Segments begin just before the somersault's initiation and end upon feet contact with the wall. Although both freestyle and backstroke turns employ a somersault, their sensor signatures differ: backstroke turns consistently finish with the swimmer on their back, whereas freestyle turns vary by athlete—some land on their stomach, side, or back.

Butterfly & Breaststroke: Open Turns

Segments begin at hand-wall contact and terminate once the swimmer places both feet on the wall in a streamlined position. Variations in turning direction (left vs. right) introduce additional labeling nuance as left hand moves in different directions.

Individual Medley:

The backstroke-to-breaststroke transition begins with a single-hand wall touch and concludes once the swimmer places both feet against the wall in a streamlined position. We mark the segment start at the moment of hand-wall contact and the end when the feet are secured in streamline. Athletes may perform this transition either as an open turn or a somersault. Somersault can be executed as forward or backward, and open turns can be executed to the left or right. Furthermore, swimmers may use either hand to contact the wall, introducing additional variability.

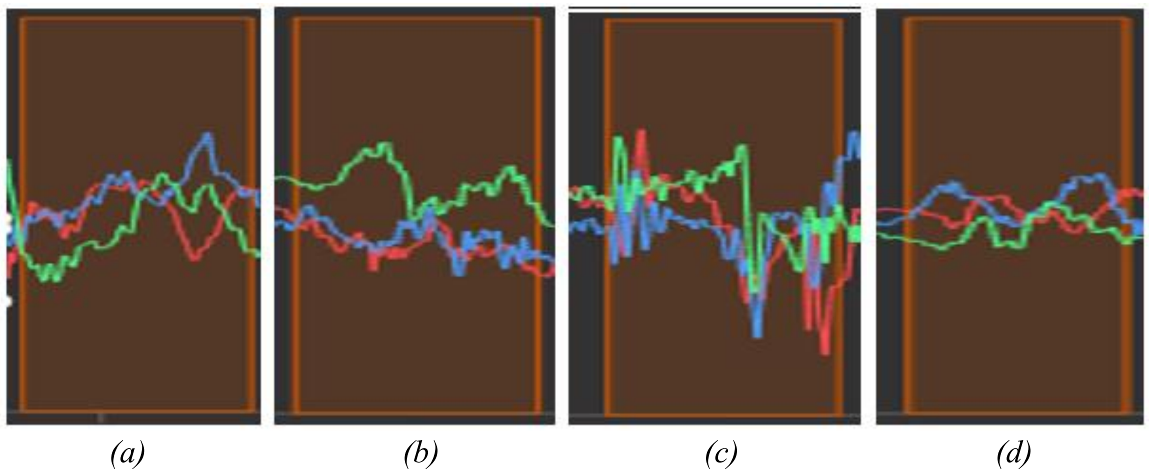


Figure 3.3 Example of accelerometer data stream of turn variability. Panels (a)–(d) show somersault, open-turn left, open-turn right, and backstroke-to-breaststroke-turn, respectively. Shaded regions indicate the window used for segmentation.

3.3.3 PUSH-OFF

Segments begin when both feet contact the wall in a streamlined position and end when the feet leave the wall. Push-off technique varies by stroke: swimmers may push off lying on their side or stomach, whereas backstroke push-offs always occur on the back (Figure 3.4).

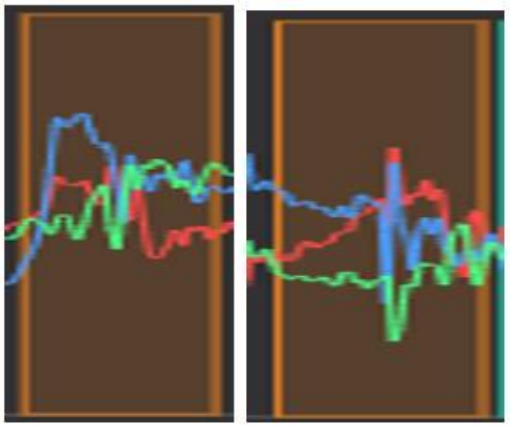

(a) (b)

Figure 3.4 Example of accelerometer data stream of (a) push-off on stomach vs. (b) on back. Shaded regions indicate the window used for segmentation.

3.3.4 Underwater Activities

Underwater glide and underwater kick phases are annotated separately, with orientation differences between strokes and swimmer preferences. Some athletes have more stable arms, while others use their arms to gain momentum.

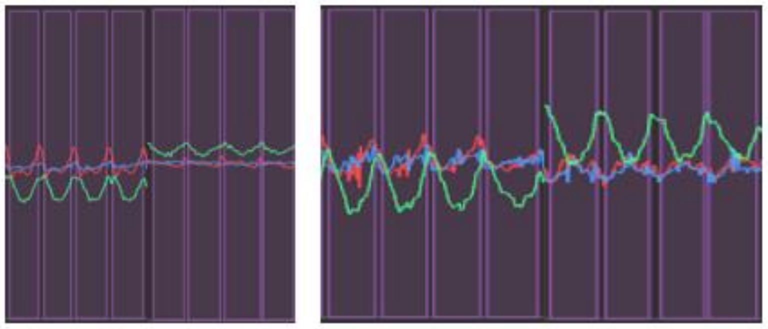

(1.a) (2.a) (1.b) (2.b)

Figure 3.5 Example of accelerometer data stream of underwater kick during (1) laying on the back vs. (2) laying on the stomach, athlete who has (a) stable arms vs. athlete who uses his (b) arms for momentum. Shaded regions indicate the window used for segmentation.

3.3.5 Wall Touch

Wall touches are the rarest class in our system, marking the end of a swim set. We define the segment start when the swimmer approaches the wall and the end at the moment of contact. These events reflect stroke specific rules: butterfly and breaststroke require both hand contacts, whereas freestyle and backstroke allow single hand touches which sometimes go unregistered if athlete touches the wall with right hand (Figure 3.6).

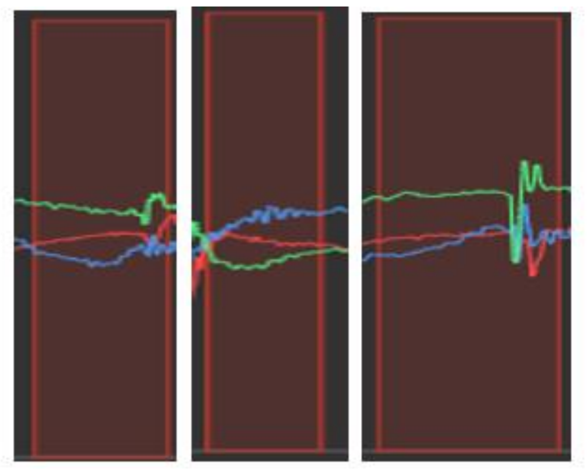


Figure 3.6 Example of accelerometer data stream of (a) left hand touch vs. (b) right hand touch vs (c) both hand touch. Shaded regions indicate the window used for segmentation.

3.3.6 Per Swimmer Variability

Across all labels, we observed substantial inter-swimmer variability in stroke technique, pacing and kick intensity. Some swimmers perform longer strokes with small “glide” breaks between cycles, while others constantly initiating stroke cycles. During kick activities, some athletes’ arms are more stable, while others use their arms to gain momentum. Some athletes perform different turn variations based on their overall technique.

We observed substantial inter-class variability, inter-swimmer variability, and stroke-dependent signal heterogeneity. Thus, a single activity label can span a wide range of sensor patterns. Capturing this full spectrum demands a larger, carefully balanced dataset; without sufficient diversity, unrepresented variations will inevitably erode both segmentation precision and classification accuracy.

Table 3.1 summarizes the total number of labeled segments per class and their average lengths in frames. Because our recordings come from collegiate swimmers competing in a 25-yard pool, where maximizing speed between laps relies heavily on streamlined underwater dolphin kicks, the underwater kick class exhibits the highest segment count and the lowest in average duration. In contrast, discrete events like wall touches and turns occur less frequently. We constructed the validation set using a leave-one-subject-out approach to assess the model's ability to generalize to new athletes.

Table 3.1 Total count and average duration per class in the train and validation datasets.

Class	Total count (train)	Average frames (train)	Total count (validation)	Average frames (validation)
Butterfly	317	71.59	56	87.59
Backstroke	219	119.98	50	142.68
Breaststroke	285	96.28	41	110.73
Freestyle	248	88.04	46	109.15
Underwater kick	818	34.77	102	32.60
Underwater glide	218	46.55	35	50.31
Push-off	206	32.73	31	32.23
Turn	152	65.86	17	68.06
Wall touch	54	42.15	11	50.18

3.4 Preprocessing

The initial step of our preprocessing pipeline converts all time-stamped sensor readings and annotations into a unified sample-index framework. Raw accelerometer and gyroscope streams, originally recorded at 52.63Hz, are first aligned by subtracting the initial timestamp from each measurement, yielding a zero-based relative time vector. Each annotated event, defined by its center time and duration, is then mapped to discrete sample indices by locating the nearest neighbors within this relative time array. Once all segment centers and lengths have been recomputed in integer sample units, both sensor streams are truncated at the highest labeled index to remove any unannotated tail.

Next, fixed-length windows are extracted from each indexed session to create uniform training examples. A sliding window of three hundred time points is advanced over the synchronized accel/gyro streams with a preset overlap of 50% yielding a total of 450 time points per window. For each window, only those annotations whose entire span falls within the window boundaries are retained. This was hand-crafted so each window contains on average five ground truth segments, and the 50% overlap ensures we include all the segments in our dataset. The original segment centers are adjusted to reflect their offset relative to the start of the window, producing a set of window-localized labels for every training example.

Prior to model ingestion, label coordinates are further down sampled to match the network's feature-level resolution by dividing both segment centers and lengths by an integer reduction factor. Textual activity names are simultaneously mapped to numeric class identifiers. To promote stochasticity during training, the resulting windows are shuffled randomly and then partitioned into mini batches of fixed size.

Together, these preprocessing stages guarantee that the multi-task model receives uniformly indexed sensor streams, precisely localized labels, and appropriately sized training batches—thereby facilitating accurate segmentation and classification of swimming activities.

CHAPTER 4: APPROACH

In this section, we first outline our end-to-end pipeline for extracting and interpreting swim events from raw IMU data, detailing both its core architectural components and the mechanism we use to propose and refine candidate segments. We then describe our training regimen, including how segments are matched to ground-truth labels and how we balance positive and negative examples. Finally, we summarize our optimization strategy—highlighting key loss terms and the hyperparameter choices that drive model convergence.

4.1 MTHARS Methodology

The MTHARS architecture is organized into several interconnected modules, each tailored to a specific function yet jointly optimized to improve overall performance.

4.1.1 SELECTIVE KERNEL CONVOLUTION

Gao et al. [38] introduced the Selective Kernel Convolution (SKConv) block to enhance multiscale feature extraction for sensor-based human activity recognition. SKConv implements a three-stage Split–Fuse–Select strategy: multiple convolutional branches with distinct receptive fields extract parallel feature representations. A global average-pooling and channel-reduction layer aggregates these into a compact descriptor. A SoftMax-based attention mechanism generates per-branch weights to fuse the branch outputs into an adaptive receptive field tailored to each input.

Moreover, this Split–Fuse–Select paradigm structurally parallels the transformer's multi-head attention mechanism, whereby parallel attention heads capture diverse aspects of the input and then recombine them via learned weights. By dynamically selecting and weighting convolutional kernels, SKConv is able to extract not only fine-grained local patterns but also broad, global dependencies across the entire IMU signal—effectively enriching the feature space in a manner akin to transformer architectures.

This dynamic kernel selection yields significant accuracy improvements on benchmark HAR datasets with only a modest increase in computational costs.

4.1.2 Multi-Scale Window Generator

Duan et al. [39] propose a unified anchor-based mechanism whereby, at each feature-sequence position, the network generates two windows per scale s∈{s1,…,s_m} where the window length is calculated by $L_1 = \lfloor N\sqrt{s} \rfloor$ and $L_2 = \left\lfloor \frac{N}{\sqrt{s}} \right\rfloor$, with N the backbone's output length. These windows serve as "temporal anchors" that together cover both short-duration impulses and long-duration activities. By mean-pooling each anchored slice and concatenating across scales, the model produces a rich, multi-scale embedding at every time step, which then feeds into joint classification and segmentation heads for end-to-end temporal activity detection. To maintain uniform window lengths at the sequence edges, we zero-pad any out-of-bounds slices and then apply mean pooling over each window, yielding exactly N consistent feature vectors for the downstream heads.

4.1.3 Classification Head

The classification head uses a pointwise (1×1) convolution which transforms the multi-scale feature representation at each time step into a vector of scores ("logits") for each activity category plus background. These logits are passed through a SoftMax function to produce confidence scores which are interpreted as the probability that each candidate window (anchor) contains a given activity. During inference, we combine these confidence scores with the boundary adjustments predicted by the regression head and apply non-maximum suppression. This ensures that only the most reliable, non-redundant segments are retained in the final activity predictions.

4.1.4 Segmentation Head

The segmentation head uses a pointwise (1×1) convolutional layer which operates on the multi-scale feature map to predict, for each predefined anchor window at each time step, two regression values: the temporal center offset (Δ_x) and the length adjustment factor (Δ_l). Here, Δ_x specifies how far to shift the anchor's midpoint in time, and Δ_l indicates the change to the anchor's duration. Applying these offsets to the default anchors yields refined start and end times that more precisely enclose the target activity. During inference, each refined window is paired with its classification confidence score and then passed to non-maximum suppression. This ensures only the most accurate, non-redundant activity predictions are retained.

4.1.5 MTHARS Module

The full MTHARS forward pass unfolds in several stages, each building on the previous one. First, the raw IMU input tensor X, containing B sequences of C sensor channels over T time steps, is fed into a one-dimensional convolutional backbone and expands the channel dimension from C to 64, followed by two Selective Kernel Convolution modules that produce 256-channel feature maps. Every convolution is accompanied by batch normalization and a ReLU activation, yielding a final feature map with 256 channels at N temporal positions for each example.

Next, the Multi-Scale Window Generator extracts fixed-length slices around each of those N positions at S different scales. For each scale, a predefined window length determines how many time steps to include. Slices that extend beyond the data boundaries are zero-padded. We then apply mean pooling over the time dimension of each slice, collapsing its length into a single vector. Because we generate two pooled maps per scale, we end up with $2S$ pooled maps, which are concatenated along the channel axis to form the unified multi-scale feature map M.

From M, the Classification Head applies a pointwise convolution to produce, at each of the N positions, a vector of K raw class scores, one for each activity category plus background. Passing these scores through a SoftMax function converts them into confidence probabilities, indicating how likely each candidate window contains a specific activity.

In parallel, the Segmentation Head also uses a pointwise convolution on M to predict two adjustment values per position. One value shifts the center of the default anchor in time, and the other scales its duration. Applying these offsets to the anchors yields refined start and end times that more precisely bound each activity proposal.

Finally, during inference we pair each refined window with its classification confidence and perform non-maximum suppression. Proposals are sorted by confidence score, and any window that overlaps a higher-scoring segment is discarded. The remaining windows, those most accurately localized and most confident, constitute the final set of detected swimming activities and their temporal extents.

Table 4.1 Overview of the MTHARS Architecture Parameters (kernel size / stride / padding / dilation)

Layers	Parameters		
Layer 1	Conv2D (5 / 3 / 1)		
SKConv	Conv2D (3 / 1 /1 / 1)	Conv2D (3 / 1 / 2 / 2)	Conv2D (3 / 1 / 3 / 3)
	Conv2D (1 / 1)		
	Conv2D (1 / 1)	Conv2D (1 / 1)	Conv2D (1 / 1)
Recognition and Segmentation	Conv1D (3 / 1 / 1)	Conv1D (3 / 1 / 1)	

4.2 Train Procedure

In training MTHARS, we leverage several interlocking components to ensure precise temporal localization and robust classification. First, a one-dimensional Intersection-over-Union (IoU) metric quantifies the overlap between each anchor window and ground-truth segment. Next, the assign window to labels routine uses IoU matching to generate per-anchor class labels and regression targets. To balance positive and negative samples, we apply hard negative mining, which selects the most challenging and informative background anchors to ensure the model can discriminate between real activities and background. The optimization itself relies on two dedicated loss functions: a localization loss (Smooth-L1) for offset regression and a classification loss (cross-entropy) for activity prediction. These are combined via a weighted sum normalized by the number of positives into a single multi-task loss. Finally, the end-to-end training procedure integrates these elements within each mini batch, iteratively updating the model parameters to minimize the combined loss. Subsequent subsections will describe each of these components in detail.

4.2.1 Intersection-over-Union (IoU)

The IoU metric, also known as the Jaccard index, quantifies the overlap between a predicted window W = (w_x, w_l) and a ground truth segment T = (t_x, t_l) along a one-dimensional timeline. Here w_x and t_x denote the center times for the predicted window and ground-truth segment, respectively, and w_l and t_l denote their durations.

We define the interval endpoints as follows: for the predicted window,

$$w_{start} = w_x - \frac{w_l}{2}, w_{end} = w_x + \frac{w_l}{2},$$

And for the ground-truth segment,

$$t_{start} = t_x - \frac{t_l}{2}, t_{end} = t_x + \frac{t_l}{2}.$$

The intersection length is defined as

$$I = \max(0, \min(w_{end}, t_{end}) - \max(w_{start}, t_{start})),$$

And the union length is defined as

$$U = (w_{end} - w_{start}) + (t_{end} - t_{start}).$$

Thus, IoU is defined as

$$IoU(W, T) = \frac{I}{U}.$$

The IoU ranges from 0 (no overlap) to 1 (perfect alignment). This scalar score is used both to assign ground-truth labels to anchors during training and to prune redundant proposals via non-maximum suppression at inference time.

4.2.2 Assign Windows to Labels

The assign windows to labels routine implements a two-stage anchor–ground-truth matching similar to SSD-style assignment in [40]. Let $\{W_i\}_{i=1}^{n_a}$ be the set of n_a anchor windows, each defined by its center time w_x^i and duration w_l^i, and $\{T_i\}_{j=1}^{n_b}$ the set of n_b ground-truth segments with centers t_x^j and lengths t_l^j. We begin by computing the IoU matrix,

$$M \in R^{n_a n_b}, M_{ij} = IoU(W_i, T_j),$$

which measures the temporal overlap between anchor W_i and segment T_j.

In stage one, each ground-truth segment T_j is paired with the single anchor W_i that maximizes M_{ij}, producing exactly one positive match per segment. In stage two, we examine all remaining anchor-segment pairs and greedily match any pair whose IoU exceeds the threshold τ, a tunable hyperparameter, to include additional positives. Anchors not matched in either stage are assigned to the background class label.

For every positive match anchor, we compute regression targets Δx and Δl to refine their temporal boundaries:

$$\Delta x = \frac{t_x - w_x}{w_l}, \Delta l = \ln\left(\frac{t_l}{w_l}\right),$$

where (w_x, w_l) are the matched anchor's center and duration and (t_x, t_l) those of the corresponding ground-truth segment. This assignment procedure yields per-anchor class labels (positive vs. background) and offset targets, forming the supervisory signals for our joint classification and localization losses.

4.2.3 Hard Negative Mining

To address the severe imbalance between background and foreground anchors, we implement hard negative mining. Let each anchor i have an assigned label y_i, where $y_i = 0$ indicates a background anchor and $y_i > 0$ a positive (matched) anchor. We first collect all the background anchors $\{i \mid y_i = 0\}$. For each background anchor i, we compute a "hardness" score h_i defined as

$$h_i = \max_{c>0} \ell_{i,c},$$

Where $\ell_{i,c}$ is the logit output by the classification head for class c at anchor i, and the maximum is taken over the K non-background activity classes. Intuitively, h_i measures how strongly the model, but incorrectly, predicts some activity for background which is a negative anchor.

We then sort these background anchors in descending order of h_i and retain only the top,

$$\lfloor R \times N_{pos} \rfloor$$

Anchors, where N_{pos} is the total number of positive anchors and R is the user-defined negative-to-positive ratio. By concentrating the classification loss on these hardest negatives, those most easily mistaken for true activities, the model avoids being swamped by trivial background examples and instead leans features that better distinguish subtle, confusion cases.

4.2.4 Loss Functions

a) Localization Loss

Let P be the set of indices corresponding to positive anchors. For each $i \in P$, the model predicts an offset vector $\widehat{\Delta}_i = (\widehat{\Delta}_{x,i}, \widehat{\Delta}_{l,i})$ and the ground truth is $\Delta_i = (\Delta_{x.i}, \Delta_{l,i})$. We define the localization loss as

$$L_{loc} = \sum_{i \in p} SmoothL1(\widehat{\Delta}_i - \Delta_i),$$

Where the smooth-L1 (Huber) function is applied elementwise to the difference $d = \widehat{\Delta} - \Delta$:

$$SmoothL1(d) = \begin{cases} 0.5d^2, & |d| < 1, \\ |d| - 0.5, & otherwise. \end{cases}$$

b) Classification Loss

let $A = P \cup N_{hard}$ be the set of anchors used for classification, where N_{hard} is the set of hard-mined negatives and K denotes the number of actual activity categories. For anchor $i \in A$, the ground-truth label is encoded as one-hot vector,

$$a_i = [a_{i,0}, \dots, a_{i,K}],$$

which exactly one entry of 1 (including class 0 for background). The model's predicted probability distribution,

$$\hat{a}_i = [\hat{a}_{i,0}, \dots, \hat{a}_{i,K}],$$

obtained via SoftMax over the $K + 1$ logits. The classification loss over the selected anchors A is then,

$$L_{conf} = -\sum_{i \in A} \sum_{c=0}^{K} a_{i,c} \log(\hat{a}_{i,c}).$$

c) Combined Loss

We balance the two objectives with weights α and β, and normalize by the number of positives $N_{pos} = |P|$:

$$L = \frac{1}{N_{pos}} \left(\alpha L_{conf} + \beta L_{loc} \right).$$

This multi-task loss trains the network simultaneously to classify each anchor correctly and to localize its temporal boundaries with high precision.

4.3 Training Hyperparameters

We train MTHARS end-to-end by integrating IoU-based anchor assignment (threshold $\tau = 0.5$), hard negative mining (neg-pos ratio $R = 0.4$), and a combined loss (weights $\alpha = 1$ for cross-entropy and $\beta = 2$ for Smooth-L1). Sensor streams are broken into sliding windows of 450 samples with 50% overlaps and a frame reduction of three, and the Multi-Scale Window Generator uses scales $\{2,3,4\}$. Optimization is performed with Adam with learning rate = 1×10^{-3}, over 50 epochs with batch size 8. At Epoch 30, we reduced the learning rate by a factor of 0.1. During inference, we apply non-maximum suppression on the classification confidences and offset regressions to produce the final temporally localized activity segments.

CHAPTER 5: EXPERIMENTS AND RESULTS

In this section, we present our experimental framework and key results for the proposed multi-task swimming activity recognition pipeline. We begin by detailing the non-maximum suppression (NMS) algorithm used during inference to select and refine high-confidence segment proposals. Next, we introduce the evaluation metrics, employed to quantify segmentation and classification performance. We then conduct a series of hyperparameter experiments, trained on five epochs, to assess their impact on model accuracy: tuning the IoU threshold for proposal selection, adjusting the hard negative mining ratio, balancing the segmentation and classification loss weights, and varying the multi-scale feature extractor's scale factors. Finally, we examine how different input data modalities (accelerometer only, gyroscope only, and combined streams) affect overall performance. Together, these experiments illuminate the sensitivity and robustness of our pipeline across a range of design choices.

5.1 Non-Maximum Suppression Algorithm

We refine the model's raw segment proposals into a final, non-overlapping set via a two-stage 1D non-maximum suppression (NMS). First, for each predicted class (excluding background), we sort its proposals by confidence and then greedily select the highest-scoring windows, suppressing any other windows of that class that overlap it. This yields a class-wise "prelim keep" list. Second, we merge across all classes: we sort those preliminarily kept windows by confidence and, in descending order, retain each window only if it does not overlap any previously selected window, suppressing overlaps by always keeping the highest-confidence proposal. The result is a final set of temporally localized, non-overlapping segments.

5.2 Evaluation Metrics

We evaluate each hyperparameter setting, trained for five epochs, using mean IoU, and both micro and macro averaged F_1 scores. Let TP_c, FP_c, and FN_c be the true positives, false positives, and false negatives for class c. The segment-level detection metrics are defined as follows:

- True Positive (TP): a proposed segment who's temporal IoU with a ground-truth segment exceeds zero and whose predicted class matches the ground-truth class.
- False Positive (FP): any proposed segment that either (i) overlaps a ground-truth segment (IoU > 0) but is assigned the wrong class, or (ii) does not overlap any ground-truth segment (IoU = 0).
- False Negative (FN): any ground-truth segment that either (i) overlaps a proposal of incorrect class, or (ii) is not overlapped by any proposal (IoU = 0).

Let P_c be the per-class precision and R_c be the recall. Then we define

$$P_c = \frac{TP_c}{TP_c + FP_c}, R_c = \frac{TP_c}{TP_c + FN_c},$$

And the class-level F_1 is,

$$F_{1,c} = 2\frac{P_c R_c}{P_c + R_c}.$$

Then the macro-F_1 averages equally over all C classes:

$$Macro - F_1 = \frac{1}{C}\sum_{c=1}^{C} F_{1,c}.$$

Micro-precision and micro-recall are defined as:

$$P_{micro} = \frac{\sum_c TP_c}{\sum_c (TP_c + FP_c)}, R_{micro} = \frac{\sum_c TP_c}{\sum_c (TP_c + FN_c)}.$$

Thus micro-F_1 is,

$$Micro - F_1 = 2\frac{P_{micro} R_{micro}}{P_{micro} + R_{micro}}.$$

Macro-F_1 provides an unweighted average of per-class F_1 scores, while micro-F_1 reflects overall detection and classification performance weighted by the frequency of each class.

5.3 IoU Threshold Tuning

Adjusting the IoU threshold τ directly governs how stringent the anchor–ground-truth matching is: a lower τ assigns more anchors as positives—potentially introducing noisy or weak matches—whereas a higher τ ensures only highly overlapping anchors are used, which can reduce recall.

Table 5.1 Effects of τ IoU threshold on Mean IoU, Macro-F_1, and Micro-F_1

τ	Mean IoU	Macro-F_1	Micro-F_1
0.3	0.5736	0.4998	0.6760
0.5	0.5603	0.5146	0.7064
0.7	0.5586	0.4746	0.7046

5.4 Hard Negative Mining Tuning

The negatives to positives ratio R determine how many of the easiest background anchors are discarded versus how many "hard" negatives are retained. Tuning R ensures the model focuses on the most informative negative examples without being overwhelmed by trivial background windows, improving its ability to distinguish true events from noise.

Table 5.2 Effects of R ratio on Mean IoU, Macro-F_1, and Micro-F_1

R	Mean IoU	Macro-F_1	Micro-F_1
0.1	0.5662	0.5043	0.7121
0.4	0.5731	0.4974	0.7091
0.8	0.4148	0.5290	0.8458
1.5	0.1129	0.2173	0.9571

5.5 Loss Weights Balancing

The relative weighting of classification versus localization loss (α, β) affects whether the network prioritizes recognizing activity classes or precisely localizing segment boundaries.

Table 5.3 Effects of α, β weights on Mean IoU, Macro-F_1, and Micro-F_1

$\alpha; \beta$	Mean IoU	Macro-F_1	Micro-F_1
$\alpha = 1; \beta = 1$	0.5724	0.5127	0.7200
$\alpha = 2; \beta = 1$	0.5802	0.5126	0.7171
$\alpha = 3; \beta = 1$	0.5688	0.5397	0.7152
$\alpha = 1; \beta = 2$	0.5715	0.5384	0.7248
$\alpha = 1; \beta = 3$	0.5583	0.5101	0.7164
$\alpha = 2; \beta = 3$	0.5725	0.5337	0.7239
$\alpha = 3; \beta = 2$	0.5744	0.5074	0.6978

5.6 Scale Factor Tuning

Our multi-scale window generator relies on a predefined set of temporal scales s to cover events of varying durations. Testing scale sets reveal how the choice of scales influences the richness of the feature embedding and the model's sensitivity to both brief impulses and extended strokes.

Table 5.4 Effect of scales s on Mean IoU, Macro-F_1, and Micro-F_1

s	Mean IoU	Macro-F_1	Micro-F_1
$s = \{0.3, 0.5\}$	0.5042	0.5126	0.7332
$s = \{0.3, 0.5, 0.8\}$	0.5510	0.4855	0.7048
$s = \{0.5, 1.5\}$	0.5208	0.5310	0.7506
$s = \{2,3\}$	0.5656	0.5079	0.6978
$s = \{2,3,4\}$	0.5498	0.5381	0.7559

5.7 Input Study

Finally, with the optimal hyperparameters fixed, we compare two input modalities—accelerometer only versus combined accelerometer and gyroscope—to assess the marginal benefit of gyroscope data. This experiment demonstrates whether the added sensor axis measurably improves activity recognition and segmentation performance.

Table 5.5 Effects of using accelerometer vs. accelerometer & gyroscope as input data on Mean IoU, Macro-F_1, Micro-F_1, stroke and kick counts Mean Absolute Error (MAE)

Input	Mean IoU	Macro-F_1	Micro-F_1	Stroke-count MAE	Kick-count MAE

Accel	0.5900	0.5894	0.7405	3.57	4.21
Accel + Gyro	0.5735	0.6565	0.7709	3.35	4.35

Table 5.6 F_1 score of all classes for accelerometer vs. accelerometer & gyroscope input data

Class	Accelerometer	Accelerometer + gyroscope
Butterfly	0.946	0.957
Backstroke	0.966	0.946
Breaststroke	0.833	0.864
Freestyle	0.878	0.882
Underwater kick	0.762	0.797
Underwater glide	0.156	0.324
Push-off	0.062	0.455
Turn	0.500	0.606
Wall touch	0.200	0.077

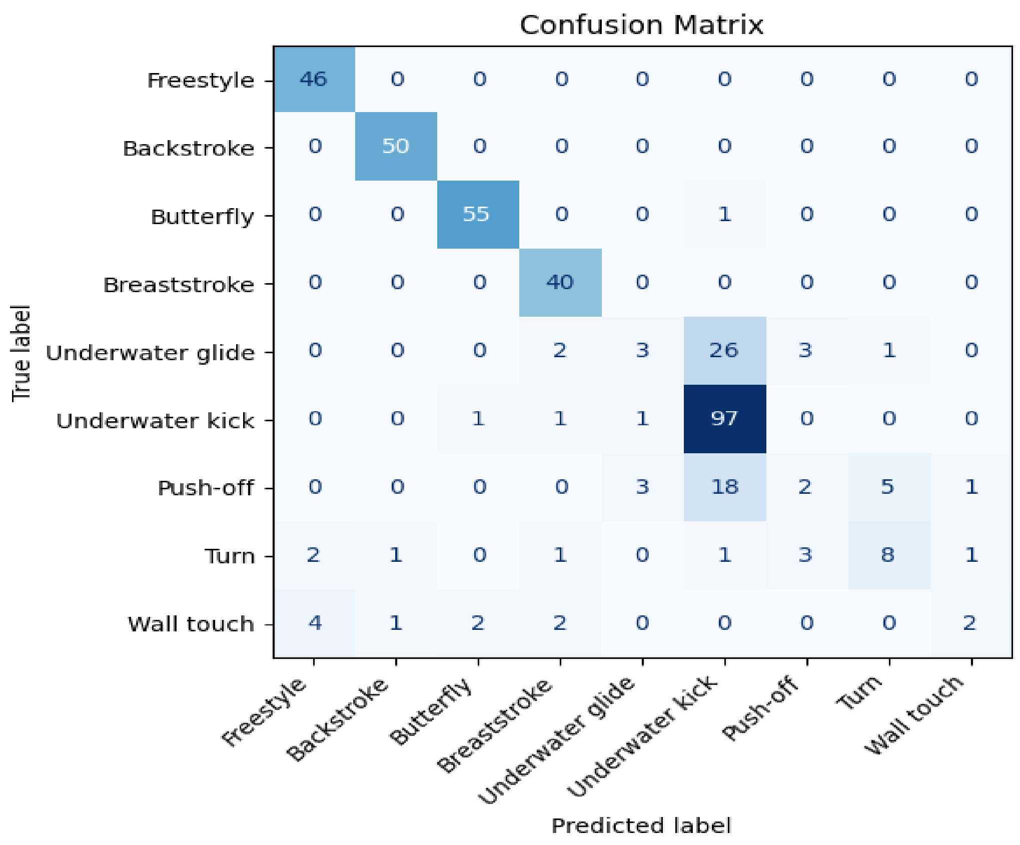
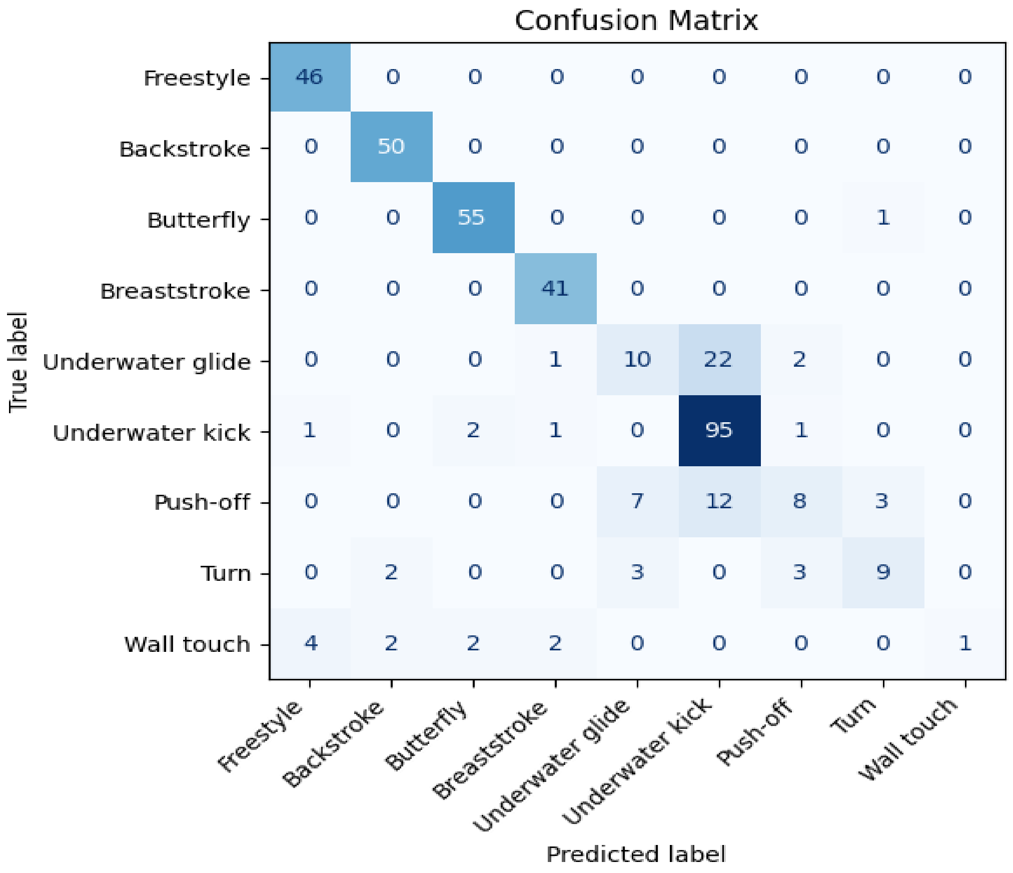


Figure 5.1 Confusion matrix of only accelerometer input data



True label \ Predicted label	Freestyle	Backstroke	Butterfly	Breaststroke	Underwater glide	Underwater kick	Push-off	Turn	Wall touch
Freestyle	46	0	0	0	0	0	0	0	0
Backstroke	0	50	0	0	0	0	0	0	0
Butterfly	0	0	55	0	0	0	0	1	0
Breaststroke	0	0	0	41	0	0	0	0	0
Underwater glide	0	0	0	1	10	22	2	0	0
Underwater kick	1	0	2	1	0	95	1	0	0
Push-off	0	0	0	0	7	12	8	3	0
Turn	0	2	0	0	3	0	3	9	0
Wall touch	4	2	2	2	0	0	0	0	1

Figure 5.2 Confusion matrix of accelerometer & gyroscope input data

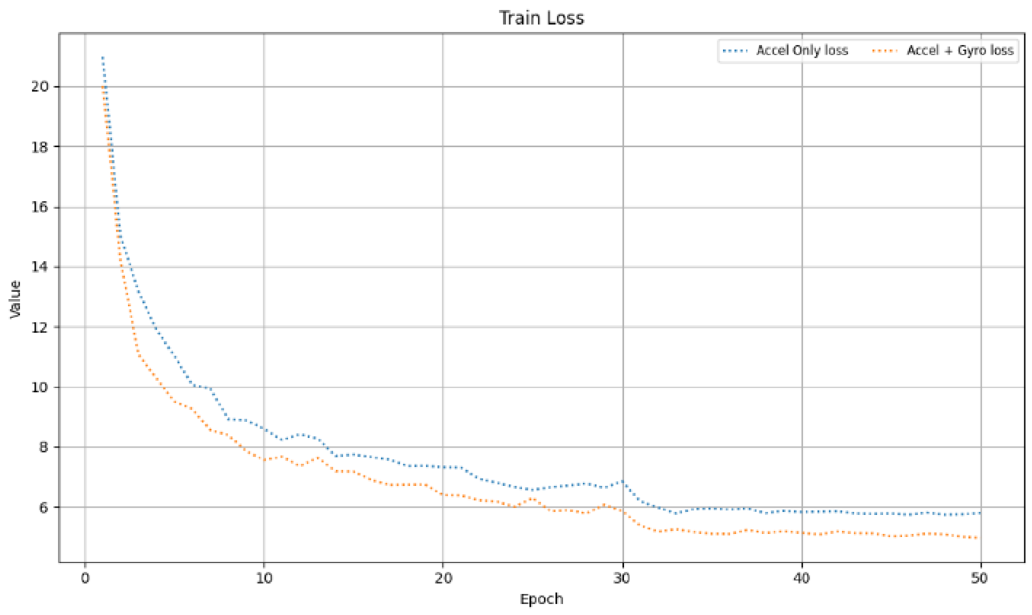


Figure 5.3 Training loss curve of accelerometer vs. accelerometer & gyroscope models showing steady convergence.

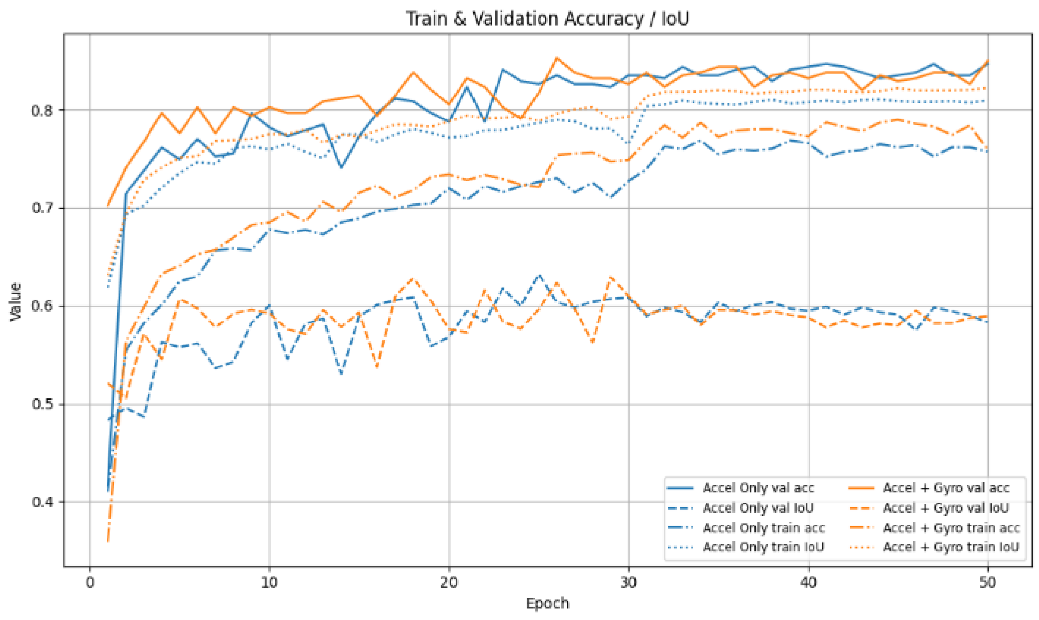


Figure 5.4 Training and validation accuracy and IoU for accelerometer vs. accelerometer & gyroscope

CHAPTER 6: CONCLUSION

Our experiments underscore the challenges of single-IMU swimming analysis—namely severe class imbalance and high inter-swimmer variability in stroke, turn, and kick patterns. In leave-one-subject-out validation, the accelerometer-only MTHARS pipeline achieved a micro-F_1 of 0.7405 and a macro-F_1 of 0.5894. Incorporating gyroscope data improved performance to a micro-F_1 of 0.7709 and a macro-F_1 of 0.6565, demonstrating that adding gyroscopic measurements enhances both overall and class-balanced detection without degrading temporal localization (mean IoU remained above 0.57 for both).

These results confirm that a single wrist-worn IMU, recording accelerometer and gyroscope measurements, can deliver accurate segmentation and classification across all competitive swim activities. When evaluated solely on stroke classification and underwater-kick detection, our accelerometer-only model performs on par with specialized pipelines focused exclusively on those tasks, with the added benefit of simultaneously counting the number of strokes and kicks.

However, the MTHARS pipeline's architectural complexity comes at a steep computational cost. The multi-scale window generator must extract, zero-pad, mean-pool, and concatenate feature slices at several scales for every time step. Then, anchor-based predictions with non-maximum suppression add further overhead. Although our segmentation and classification accuracies are strong, this exhaustive feature-extraction and proposal-pruning workflow incurs significant memory and latency penalties, making the current implementation unsuitable for true real-time inference on resource-constrained devices such as smartwatches or embedded swim monitors without additional optimization or architectural simplification.

CHAPTER 7: FUTURE WORK

Future work will begin by tackling the pronounced class imbalance in our dataset. We plan to collect additional examples of the rarer events, particularly wall touches and various turn types, to equalize the number of samples per class as much as possible. Once a more balanced corpus is assembled, we will explore targeted data-augmentation techniques to further expand the effective dataset size and smooth out residual imbalances. New models will then be trained and evaluated on these enriched datasets to quantify the benefits of both increased sample diversity and balanced class distributions.

In parallel, we will extend our performance evaluation to include lap-time and full-set-time prediction tasks. By leveraging the wall-push and turn classes, we can construct automated estimators of split times, and by incorporating both the initial wall-push and final wall-touch events, we can predict the total duration of each swim set. These temporal-prediction capabilities would transform our pipeline from purely recognition and segmentation toward a comprehensive timing and analytics tool for coaches and athletes.

Finally, we will investigate alternative sensor placements to reduce signal ambiguity and variability. Head or goggle mounted IMUs could provide more stable reference frames for wall-contact detection and streamline the interpretation of underwater-kick and stroke cycles. By comparing wrist, head, and hybrid configurations, we aim to identify the optimal mounting strategy that maximizes accuracy while minimizing disruption to an athlete’s natural technique.

Although MTHARS delivers strong accuracy, its multi-scale window generator incurs substantial computational overhead, making real-time inference challenging. To address this, we will investigate fully sequential, memory-based architectures, such as LSTM or GRU networks, which ingest raw IMU streams end-to-end without explicit window slicing and pooling. By inherently modeling temporal dependencies, these recurrent models eliminate the expensive multi-scale feature extraction step, significantly reducing intermediate computations and latency. This shift promises a simpler pipeline and faster inference on embedded hardware, paving the way for live, low-latency feedback during swimming training.

REFERENCES

[1] Chen, Liming, et al. "Sensor-based activity recognition." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42.6 (2012): 790-808.

[2] Chen, Liming, Chris D. Nugent, and Hui Wang. "A knowledge-driven approach to activity recognition in smart homes." IEEE Transactions on Knowledge and Data Engineering 24.6 (2011): 961-974.

[3] Cook, Diane J., Juan C. Augusto, and Vikramaditya R. Jakkula. "Ambient intelligence: Technologies, applications, and opportunities." Pervasive and mobile computing 5.4 (2009): 277-298.

[4] Philipose, Matthai, et al. "Inferring activities from interactions with objects." IEEE pervasive computing 3.4 (2004): 50-57.

[5] Chen, Datong, Jie Yang, and Howard Wactlar. "A study of detecting social interaction with sensors in a nursing home environment." Computer Vision in Human-Computer Interaction: ICCV 2005 Workshop on HCI, Beijing, China, October 21, 2005. Proceedings. Springer Berlin Heidelberg, 2005.

[6] Krishnan, Narayanan C., and Diane J. Cook. "Activity recognition on streaming sensor data." Pervasive and mobile computing 10 (2014): 138-154.

[7] Wang, Zhelong, et al. "Human-human interactional synchrony analysis based on body sensor networks." IEEE Transactions on Affective Computing 10.3 (2017): 407-416.

[8] Van Laerhoven, Kristof, and Kofi Aidoo. "Teaching context to applications." Personal and Ubiquitous Computing 5 (2001): 46-49.

[9] Wren, Christopher R., and Emmanuel Munguia Tapia. "Toward scalable activity recognition for sensor networks." International Symposium on Location-and Context-Awareness. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[10] Kim, Eunju, Sumi Helal, and Diane Cook. "Human activity recognition and pattern discovery." IEEE pervasive computing 9.1 (2009): 48-53.

[11] Rashidi, Parisa, et al. "Discovering activities to recognize and track in a smart environment." IEEE transactions on knowledge and data engineering 23.4 (2010): 527-539.

[12] Chathuramali, KG Manosha, and Ranga Rodrigo. "Faster human activity recognition with SVM." International conference on advances in ICT for emerging regions (ICTer2012). IEEE, 2012.

[13] Paramasivam, Kalaivani, Mohamed Mansoor Roomi Sindha, and Sathya Bama Balakrishnan. "KNN-based machine learning classifier used on deep learned spatial motion features for human action recognition." Entropy 25.6 (2023): 844.

[14] Feng, Zengtao, Lingfei Mo, and Meng Li. "A Random Forest-based ensemble method for activity recognition." 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2015.

[15] Ronao, Charissa Ann, and Sung-Bae Cho. "Human activity recognition with smartphone sensors using deep learning neural networks." Expert systems with applications 59 (2016): 235-244.

[16] Panwar, Madhuri, et al. "CNN based approach for activity recognition using a wrist-worn accelerometer." 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2017.

[17] Müller, Philipp Niklas, et al. "Imu-based fitness activity recognition using cnns for time series classification." Sensors 24.3 (2024): 742.

[18] Zhao, Huipeng. "A parallel CNN architecture for sport activity recognition based on minimal movement data." Scientific Reports 14.1 (2024): 31697.

[19] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[20] Murad, Abdulmajid, and Jae-Young Pyun. "Deep recurrent neural networks for human activity recognition." Sensors 17.11 (2017): 2556.

[21] Guan, Yu, and Thomas Plötz. "Ensembles of deep lstm learners for activity recognition using wearables." Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies 1.2 (2017): 1-28.

[22] Mekruksavanich, Sakorn, and Anuchit Jitpattanakul. "Lstm networks using smartphone data for sensor-based human activity recognition in smart homes." Sensors 21.5 (2021): 1636.

[23] Vakacherla, Sai Siddarth, et al. "Single accelerometer to recognize human activities using neural networks." Journal of Biomechanical Engineering 145.6 (2023): 061005.

[24] Khatun, Mst Alema, et al. "Deep CNN-LSTM with self-attention model for human activity recognition using wearable sensor." IEEE Journal of Translational Engineering in Health and Medicine 10 (2022): 1-16.

[25] Yao, Shuochao, et al. "Deepsense: A unified deep learning framework for time-series mobile sensing data processing." Proceedings of the 26th international conference on world wide web. 2017.

[26] Ma, Haojie, et al. "AttnSense: Multi-level attention mechanism for multimodal human activity recognition." IJCAI. 2019.

[27] Kim, Yeon-Wook, et al. "Wearable IMU-based human activity recognition algorithm for clinical balance assessment using 1D-CNN and GRU ensemble model." Sensors 21.22 (2021): 7628.

[28] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[29] Mahmud, Saif, et al. "Human activity recognition from wearable sensor data using self-attention." ECAI 2020. IOS Press, 2020. 1332-1339.

[30] Shavit, Yoli, and Itzik Klein. "Boosting inertial-based human activity recognition with transformers." IEEE Access 9 (2021): 53540-53547.

[31] Dirgová Luptáková, Iveta, Martin Kubovčík, and Jiří Pospíchal. "Wearable sensor-based human activity recognition with transformer model." Sensors 22.5 (2022): 1911.

[32] Guo, Xin, et al. "Enhancing the Transformer Model with a Convolutional Feature Extractor Block and Vector-Based Relative Position Embedding for Human Activity Recognition." Sensors (Basel, Switzerland) 25.2 (2025): 301.

[33] Delhaye, Erwan, et al. "Automatic swimming activity recognition and lap time assessment based on a single IMU: a deep learning approach." Sensors 22.15 (2022): 5786.

[34] Zhang, Zhendong, et al. "IMU-based underwater sensing system for swimming stroke classification and motion analysis." 2017 IEEE International Conference on Cyborg and Bionic Systems (CBS). IEEE, 2017.

[35] Chen, Longwen, and Dean Hu. "An effective swimming stroke recognition system utilizing deep learning based on inertial measurement units." Advanced Robotics 37.7 (2023): 467-479.

[36] Aminikhanghahi, Samaneh, Tinghui Wang, and Diane J. Cook. "Real-time change point detection with application to smart home time series data." IEEE Transactions on Knowledge and Data Engineering 31.5 (2018): 1010-1023.

[37] Li, Shuangjian, et al. "P2LHAP: Wearable sensor-based human activity recognition, segmentation and forecast through Patch-to-Label Seq2Seq Transformer." IEEE Internet of Things Journal (2024).

[38] Gao, Wenbin, et al. "Deep neural networks for sensor-based human activity recognition using selective kernel convolution." IEEE Transactions on Instrumentation and Measurement 70 (2021): 1-13.

[39] Duan, Furong, et al. "A multitask deep learning approach for sensor-based human activity recognition and segmentation." IEEE Transactions on Instrumentation and Measurement 72 (2023): 1-12.

[40] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Cham: Springer International Publishing, 201