future internet

MDPI

Article

Smart Fitness Solutions: An Edge-Computing Approach for Biometric Monitoring and Real-Time Muscle Fatigue Detection

Mohamed Ahmed El Ghazouly [1], Duaa Zuhair Al-Hamid [2,*], Omar Gadelmawla [1], Hamid GholamHosseini [3], Mirza Mansoor Baig [4] and Shereen Afifi [1]

[1] Computer Science and Engineering Department, Faculty of Media Engineering and Technology (MET), German University in Cairo (GUC), New Cairo 11835, Egypt; mohamedahmed4016@student.guc.edu.eg (M.A.E.G.); omarezzat2011@gmail.com (O.G.); shereen.moataz@guc.edu.eg (S.A.)
[2] Department of Computer and Information Sciences, Auckland University of Technology (AUT), Auckland 1010, New Zealand
[3] Department of Electrical and Electronic Engineering, Auckland University of Technology (AUT), Auckland 1010, New Zealand; hamid.gholamhosseini@aut.ac.nz
[4] Dedalus Healthcare New Zealand Limited, Auckland 6011, New Zealand; mirza.baig@dedalus.com
* Correspondence: duaa.alhamid@aut.ac.nz

Abstract

Real-time monitoring of muscle fatigue could support safer strength training, yet most wearable systems depend on cloud processing, which adds latency and exposes physiological data. This work presents and evaluates an edge-based prototype for biometric monitoring during biceps curls. A wearable node integrating a nine-axis IMU (MPU-9250) and a photoplethysmography heart-rate sensor (MAX30102) streams data from an ESP32 microcontroller over UDP to a Raspberry Pi 5, which performs feature scaling and inference with a compact feedforward neural network (two hidden layers of 64 and 32 units) trained on a publicly available biceps-curl dataset. The classifier takes a 10-element feature vector of IMU, heart-rate and acceleration-derived inputs and outputs three fatigue levels. Under leave-one-subject-out cross-validation on ten participants, the three-class classifier attained a mean accuracy of 0.77 (SD 0.25; 95% CI 0.59–0.95), with a strongly bimodal per-subject distribution (0.40–0.98). Deployed on the edge node and evaluated end-to-end over 5172 annotated windows under a binary no fatigue versus fatigue formulation, the model recovered 91% of non-fatigued but only 10% of fatigued windows (balanced accuracy 0.51, Matthews correlation 0.02), which is no better than chance and below a majority-class baseline. In its current form the classifier is therefore not a reliable fatigue detector; we analyse candidate causes, including training-side class skew and differences between the source-dataset and deployment sensors, and compare them against majority-class and chance reference points. On the systems side, the pipeline runs fully locally with 10 ms model inference per sample, 42 ms gateway processing and a mean wireless round-trip time of 180 ms after rate-limiting the link. The implemented physical, communication and edge-inference layers are designed as the sensing foundation for a future Digital Twin, which is not implemented in this work.

Keywords: muscle fatigue detection; edge computing; wearable sensors; IMU; feedforward neural network; real-time inference; Raspberry Pi; ESP32; IoT; class imbalance

Check for updates

Academic Editors: Yifan Guo and Paolo Bellavista

Received: 7 September 2026
Revised: 16 October 2026
Accepted: 27 October 2026
Published: 6 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Muscle fatigue is unavoidable. Whenever a muscle is asked to contract repeatedly under load, its ability to generate force drops, and the longer that goes on the higher the

chance of injury [1,2]. For someone doing biceps curls in a gym, this matters: the last set of an exercise is also the one where form deteriorates and strain injuries are most likely [3,4]. A system that could warn a lifter that they have hit a fatigued state in real time would be useful, both for safety and for adjusting training prescription.

The problem is that detecting fatigue is harder than it sounds. The gold-standard method, surface electromyography, gives a direct neuromuscular reading but requires skin prep and careful electrode placement and is sensitive to motion artefacts [5,6]. Blood lactate testing is invasive. VO_2 max requires a face mask and a lab. None of these survive contact with a normal gym session [7]. Inertial measurement units (IMUs) are a more practical alternative: a wrist-worn IMU can pick up changes in angular velocity and acceleration that correlate well with perceived fatigue, and Elshafei and Shihab [8] have shown that an IMU alone is enough to classify biceps curl fatigue with 88% accuracy across subjects.

What is less settled is how to deploy that kind of classifier in practice. Most wearable fatigue systems send raw sensor data to the cloud for inference, which works but introduces latency, costs bandwidth, and exposes physiological data on the wire [9,10]. Edge computing reverses that: the model runs on a device close to the sensor, so the data never leaves the user [11,12]. Recent work in TinyML has shown that even microcontroller-class hardware can host useful neural networks [13,14], and Raspberry Pi-class single-board computers can comfortably handle the more demanding feature pipelines [15].

This paper describes a working prototype that puts the pieces together. We built a wearable node around an MPU-9250 nine-axis IMU and a MAX30102 photoplethysmography sensor, both read by an ESP32 microcontroller that forwards the data over Wi-Fi to a Raspberry Pi 5. The Pi builds a 10-element feature vector and runs a two-layer feedforward neural network trained on the public Elshafei biceps curl dataset [8]. Per-sample inference is completed in 10 ms, gateway processing is completed in 42 ms, and the wireless round-trip time after tuning is 180 ms. CPU utilisation sits around 15%. The system does not need a cloud connection at any point. The classification results, however, are weak once the model is deployed: on 5172 windows scored end-to-end, only 10% of fatigued windows are detected. We report this result in full rather than the offline figures alone, because the gap between the two is the most informative finding of the study. The implemented layers are intended as the sensing and edge-inference foundation of a future Digital Twin pipeline [16,17]; no Digital Twin is implemented here.

The hardware itself is not novel: an ESP32 sensor node feeding a single-board-computer gateway is a common edge-computing pattern for the IoT [18], and we do not claim otherwise. The contribution lies in what is measured with it. Published IMU fatigue classifiers, including the one on which this work builds [8], are evaluated offline on the data they were trained on. Whether such a model keeps its accuracy when moved onto different, low-cost sensors and run live is rarely tested. The contributions of the paper are:

- An end-to-end, measured evaluation of a public-dataset-trained fatigue classifier transferred to separately built low-cost hardware (MPU-9250, MAX30102, ESP32, and Raspberry Pi 5), reporting both offline LOSO-CV and deployed performance on the same model. This exposes a large offline-to-deployment gap (accuracy 0.77 offline versus fatigue recall 0.10 deployed).
- An analysis of that gap with chance and majority-class reference points and confidence intervals, and a discussion of its candidate causes (training-side class skew, sensor and placement differences, and label definition).
- A systems characterisation of the edge pipeline: model and gateway latency, CPU, memory and temperature, and a wireless-link study showing that UDP transmission rate, not model inference, dominates the latency experienced by the user (round-trip time reduced from 15 s to 180 ms).

- A modular five-layer IoT architecture in which the physical, communication and edge-intelligence layers are implemented and evaluated, while the Digital Twin and application layers are only specified, as the intended path for future work.

The rest of the paper is organised as follows. Section 2 covers related work in fatigue detection, machine learning for activity recognition, edge AI, and the Digital Twin literature. Section 3 walks through the system design and methods. Section 4 reports results. Section 5 discusses what worked, what did not, and where the boundaries are. Section 6 concludes.

2. Related Work

The system proposed in this paper sits at the intersection of four research areas, and we review each in turn. We begin with the sensing side of muscle-fatigue detection, comparing the electromyographic, cardiorespiratory, and inertial modalities and explaining why a wrist-worn IMU is the most practical choice for an unsupervised gym setting. We then look at the machine-learning models used to turn those signals into a fatigue decision, from classical classifiers to deep networks, and at the cross-subject generalisation problem that recurs throughout this literature. Next we survey edge computing and embedded AI, which are what allow inference to happen on the device rather than in the cloud, and the hardware patterns that make this feasible on low-cost single-board computers. Finally, we place the work in the context of Digital Twin research for fitness and health monitoring, where a reliable edge-sensing layer is the prerequisite that much of the downstream modelling assumes but rarely builds. Each subsection below identifies the specific gap that motivates the corresponding design decision in our system.

2.1. Detecting Muscle Fatigue

Three sensing modalities dominate the fatigue-detection literature: surface electromyography (sEMG), cardiorespiratory measurements, and inertial sensing. sEMG is the most established. Spectral features such as median and mean power frequency drop in a fairly predictable way as a muscle fatigues, and Wang et al. [19] used wavelet packet denoising to clean those signals up enough for reliable RMS and IEMG extraction. Al-Mulla et al. [20] built one of the earliest autonomous wearable sEMG systems that could flag local fatigue without supervision. The catch is practicality: sEMG electrodes need skin preparation, sit awkwardly under clothing, and tend to pick up motion artefacts during heavy lifts [7].

IMU-based systems sidestep most of that. The dataset and approach we build on are due to Elshafei and Shihab [8], who showed that a single wrist-mounted IMU is enough to classify biceps curl fatigue across subjects, using statistical features of the inertial signals. Marotta et al. applied IMU-based fatigue detection to running and reviewed accelerometer-based identification of lower-limb fatigue [21,22], and Bangaru et al. [23] and Chang et al. [24] applied wearable sensing with deep learning to fatigue in construction workers and runners, respectively. Camomilla et al. [25] reviewed the in-field use of inertial sensors in sport more broadly, and their main complaint is the lack of standardisation in how features are extracted and reported. Fatigue has also been quantified with wearable EMG during surgery (Merbah et al. [26]) and with fused sEMG and ECG during exercise (Mu et al. [27]), although these approaches rely on electrophysiological rather than inertial sensing.

2.2. Machine Learning for Fatigue and Activity Recognition

For a feature vector of the kind used here, classical machine learning often does well enough. Random forest is a frequent baseline because it tolerates correlated features and is cheap to evaluate [28,29]. Logistic regression and generalised linear models are easier to interpret but tend to underperform on the kind of periodic signals that come from repetition-based exercises [8]. Neural networks have been applied successfully to exercise

recognition from wearable data [30], and Elshafei and Shihab [8] report 98% subject-specific and 88% cross-subject accuracy with a simple two-layer FNN—comfortably above their RF, decision tree, and logistic regression baselines on the same data. We use a similar architecture here for that reason. Elshafei et al. [31] also quantified what happens when an activity recognition model is run on fatigued data without being trained for it, and the answer is that accuracy degrades sharply if fatigue-specific features are not in the input. Nair et al. [32] and Otalora et al. [33] applied machine learning to physical-fatigue estimation from IMU and EMG sensors and to upper-limb fatigue estimation from wearable sensors, respectively, on small cohorts of the kind that also limits the present work (Section 4).

Deep learning gives the best accuracy on raw time-series sensor data if you can afford the parameter count. CNN, LSTM, and CNN-LSTM hybrids have all been applied to wearable HAR with strong results [34,35], though Bayat et al. [36] showed that classical ML on smartphone accelerometer data can also achieve high recognition accuracy. Zhang and Sawchuk [37] proposed a feature selection framework for multimodal HAR that aligns closely with the Spearman-rank reduction Elshafei and Shihab used; the correlation-based screening in Section 3.2 follows the same idea. Jiang et al. [38] predicted exercise fatigue from wearable motion data, and Li et al. [39] surveyed non-invasive techniques for muscle-fatigue monitoring.

The flip side of deep models is cost. Putting them on edge hardware usually requires quantisation or pruning [13,14], which is why a compact FNN running on pre-extracted features is often the more sensible choice in practice—the position we take here. Kim et al. [40] and Biró et al. [41] applied deep learning to fatigue prediction in sport, from GPS and perceived-exertion data and from IMU time series, respectively.

Cross-subject generalisation is the recurring problem. Fatigue looks different from one person to the next because biomechanics, muscle composition, and fitness all differ [42]. Leave-one-subject-out cross-validation is the standard way to test for this with small cohorts [8], and we use it here. Elshafei et al. [42] have shown that personalising the model improves on a generic cross-subject model, so without per-subject calibration cross-subject accuracy is limited—something our own results echo.

2.3. Edge Computing and Embedded AI

Running the model on the device, rather than in the cloud, helps with latency, privacy, and connectivity [9,12]. The combination of edge computing with trained ML models is what is now called Edge AI [11]. Figure 1 contrasts the two paradigms: in the cloud model, computation happens at a remote data centre, whereas in the edge model it happens next to the sensor.

Abdellatif et al. [10] argue that pushing feature extraction and inference to a mobile edge node cuts response times and wireless traffic compared with cloud-forwarded pipelines, which is the same trade-off we are making here. Srivastava et al. [13] ran HRV-based stress and sleep monitoring on a microcontroller, showing that models of this kind fit on very constrained hardware.

Raspberry Pi single-board computers have become a default for IoT health prototypes because they run a full Linux stack, support Python, and have enough GPIO and USB capability to handle multiple sensors at once [43,44]. Dewanto et al. [15] used edge computing for heart-rate monitoring with smart wearables, and our setup follows a similar pattern. Low-cost microcontrollers and single-board computers are widely used in IoT health-monitoring prototypes [45,46]; in our system the ESP32 fills the lower tier, being cheap, wireless-capable and easy to power from a battery. The two-tier pattern that the present system uses, a sensor node feeding a more capable gateway at the network edge, is a common architecture in edge computing for the IoT [18]. The specific sensors we

chose—MPU-9250 and MAX30102—have both been documented extensively as reliable low-cost options [47,48]. Background on edge versus cloud trade-offs and on real-time systems is covered in industry write-ups [49,50], and Seshadri et al. [51] review wearable physiological monitoring for athletes specifically. Baig et al. [52] survey wearable sensor and IoT-based monitoring more broadly and identify usability and interoperability as the persistent barriers, which is consistent with our design choice to keep the hardware off-the-shelf.

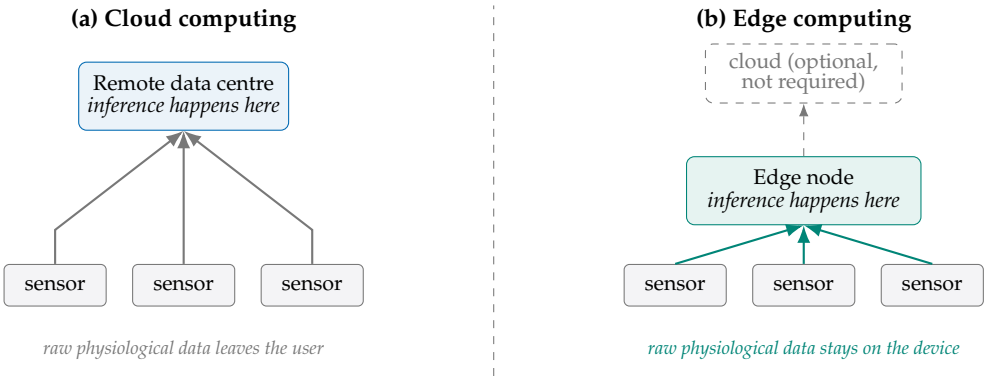


Figure 1. Cloud computing versus edge computing. In the edge paradigm adopted here, inference is performed locally on the device, so raw physiological data never leaves the user.

2.4. Digital Twins for Fitness Monitoring

A Digital Twin (DT) is, at its simplest, a virtual replica of a physical entity that stays synchronised with sensor data from that entity [16]. Related work by Afifi et al. has built an IoT-cloud DT system for remote patient monitoring [16] on top of an IoT-based mobile application for continuous patient monitoring [17], and has examined generative AI for personalised care plans [53] and the opportunities and open challenges of AI-driven patient monitoring with smart medical devices [54]. Jeong et al. [55] review wearable devices for precision medicine and continuous health-state monitoring, the kind of data stream on which a DT sensing layer depends.

3. System Design and Methods

This section details the development of the muscle fatigue detection system, encompassing the dataset characteristics, feature engineering pipeline, neural network architecture, and the edge-computing deployment strategy. The dataset, feature set (Table 1), network configuration and training settings (Table 2), and benchmarking procedure are reported below, and the source dataset is publicly available. Figure 2 shows an overview of the complete pipeline, from the wearable sensing node through the ESP32 gateway to on-device inference on the Raspberry Pi 5.

Table 1. Model input features. All features are instantaneous per-sample values (no temporal aggregation) and are standardised before inference. Raw axes are used in the units in which they are recorded; standardisation removes scale differences.

Feature	Sensor (Training/Deployment)	Computation	Unit
a_x, a_y, a_z	IMU accelerometer (source IMU/MPU-9250)	raw axis	sensor units
g_x	IMU gyroscope	raw axis	sensor units
m_x, m_y, m_z	IMU magnetometer	raw axis	sensor units
HR	smartwatch/MAX30102 PPG	beats per minute	bpm
A_{total}	derived	Equation (1)	accelerometer units
$\hat{F}$	derived	Equation (2)	proxy (a.u.)

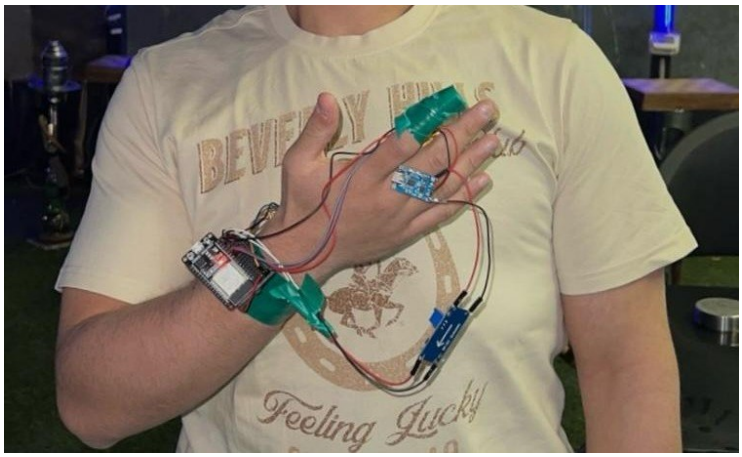


Figure 2. End-to-end system overview. A wrist-worn sensing node (MPU-9250 IMU and MAX30102 PPG sensor read by an ESP32) streams windowed measurements over a lightweight wireless link to a Raspberry Pi 5, which performs feature scaling and runs the feedforward neural network locally before displaying the fatigue state.

Table 2. Model and training configuration.

Setting	Value
Input	10 standardised features (Table 1)
Hidden layers	64 and 32 units, ReLU
Output	3 units, softmax
Trainable parameters	2883
Optimiser	Adam
Loss	categorical cross-entropy
Early stopping	patience 5, on a validation split of the training participants
Class weighting/resampling	none
Evaluation	leave-one-subject-out CV (10 folds)
Training hardware/software	workstation; Python 3.15 with Keras and scikit-learn
Deployment hardware	Raspberry Pi 5 (8 GB, 64-bit OS); ESP32; MPU-9250; MAX30102

3.1. Data Source and Acquisition Protocol

The model was trained on the public biceps-fatigue dataset of Elshafei and Shihab [8]. In the original study, 20 volunteers performed concentration biceps curls with a 4.5 kg dumbbell (five warm-up repetitions followed by five sets of 15 repetitions per arm), wearing a wrist-mounted nine-axis IMU sampled at 50 Hz (accelerometer, gyroscope and magnetometer) and a smartwatch heart-rate monitor. After each repetition participants reported their Rate of Perceived Exertion (RPE) on Borg's 6–20 scale, and heart rate was used to check the consistency of the reported RPE. The original data were collected under an ethics approval granted to that study (Concordia University, protocol 30008716).

This work uses a subset of 10 of the 20 participants (aged 29–33, BMI 18–28).

Two distinct data sources are used, and they must not be confused:

1. Training and offline validation data: the 10-participant subset of the public dataset above, used for model training and the LOSO-CV results in Section 4.1.2. These data were recorded with the original study's IMU and smartwatch, not with the hardware described in this paper.
2. Deployment evaluation data: the 5172 annotated windows acquired by the prototype's own sensing node (MPU-9250 and MAX30102 read by the ESP32) during the experimental trials and processed end-to-end by the deployed edge pipeline (Section 4.4). These data therefore differ from the training data in sensor hardware, sensor placement and heart-rate source.

3.2. Labels, Windowing and Features

Fatigue labels. The source dataset provides a per-repetition Borg RPE. The original authors used a binary split (6–16 non-fatigued, 17–20 fatigued) [8]. In this work the RPE annotations are grouped into three ordered levels: no fatigue (class 0), mid fatigue (class 1) and full fatigue (class 2). Every IMU sample within a repetition inherits that repetition's label. For the binary deployment evaluation, label merging is used: the predicted class is the softmax argmax, and classes 1 and 2, in both the predictions and the ground truth, are mapped to a single fatigue class.

Sampling and windowing. Training and LOSO-CV operate on individual 50 Hz samples: each sample is one feature vector and one classification. On the deployed node the ESP32 averages the samples acquired within each 200 ms interval on-device and transmits one packet every 200 ms (5 packets/s). A deployment "window" is therefore one 200 ms packet (non-overlapping, stride 200 ms) and receives one prediction and one ground-truth label.

Raw signals were cleaned by removing inconsistent timestamps and handling missing values through row deletion. Features were standardised to zero mean and unit variance with `StandardScaler`. To avoid information leakage between participants, the scaler was fitted on the training participants of each LOSO fold only and then applied unchanged to the held-out participant; the deployed model uses the scaler fitted on its training data. Two derived features were added:

- Total acceleration (A_{total}): the Euclidean norm of the tri-axial acceleration components:

$$A_{total} = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{1}$$

- Acceleration-derived force proxy ($\hat{F}$): the dumbbell mass $m = 4.5\,\text{kg}$ multiplied by the total acceleration, with A_{total} expressed in $\text{m}\,\text{s}^{-2}$:

$$\hat{F} = m \times A_{total} = 4.5 \times A_{total} \tag{2}$$

  $\hat{F}$ is not a measured force. The IMU is worn on the wrist and the accelerometer also senses gravity, so $\hat{F}$ is at best a scaled indicator of movement intensity. Because m is constant, $\hat{F}$ is perfectly linearly correlated with A_{total} and adds no information to the classifier. It was retained only for comparability with the source study; after standardisation the two inputs are identical, and removing either one is expected to leave the results unchanged.

Feature screening. Twelve candidate inputs were screened by their point-biserial correlation with the binary fatigue label the nine IMU axes, heart rate, A_{total} and $\hat{F}$. The two gyroscope components with near-zero correlation (g_y, g_z) were removed, leaving the ten inputs listed in Table 1.

Role of heart rate. Heart rate is one of the model inputs. In training it comes from the smartwatch used in the source study; in deployment it comes from the MAX30102 on the fingertip, from whose signal beats per minute are estimated on the ESP32. Heart rate changes slowly relative to a single repetition, so within a set it acts mostly as a context variable, and the two devices differ in placement and processing. This is one of the training/deployment differences discussed in Section 5.1.

3.3. Model Architecture and Training Strategy

A feedforward neural network with two hidden layers was implemented, following the FNN that performed best in the source study [8]. The input is the 10-element standardised feature vector (Table 1). The network has:

- Hidden layer 1: fully connected, 64 units, and ReLU activation (each unit receives all 10 inputs).
- Hidden layer 2: fully connected, 32 units, and ReLU activation.
- Output layer: three units with softmax activation, giving the probabilities of no fatigue (class 0), mid fatigue (class 1) and full fatigue (class 2); the predicted class is the one with the highest probability.

The network has $10 \times 64 + 64 + 64 \times 32 + 32 + 32 \times 3 + 3 = 2883$ trainable parameters. Its structure is shown in Figure 3.

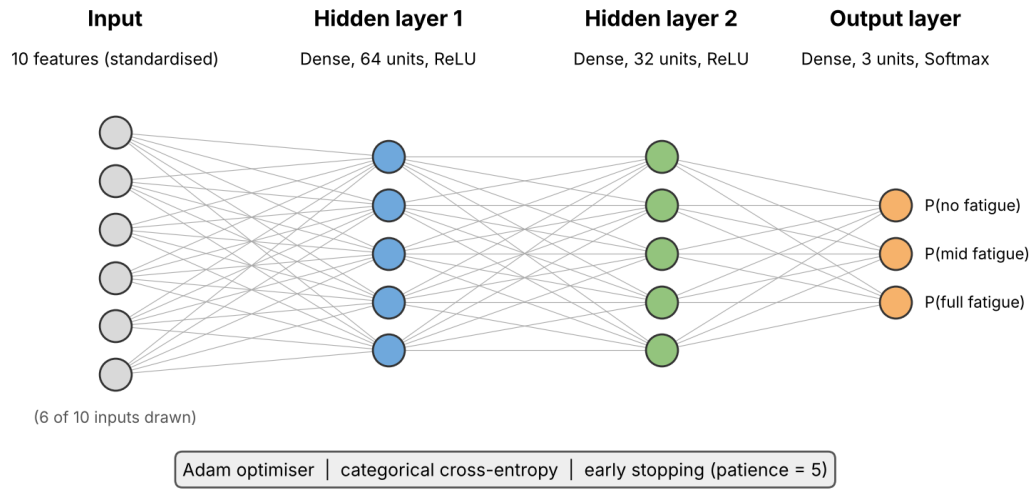


Figure 3. Architecture of the feedforward neural network. The 10-element feature vector is the input; it feeds a 64-unit and a 32-unit ReLU hidden layer, followed by a three-unit softmax output whose nodes give the probabilities of no, mid and full fatigue.

To estimate how the model generalises to unseen people, we employed leave-one-subject-out cross-validation (LOSO-CV). For each fold, data from one participant served as the test set, while the remaining nine were used for training. Within the nine training participants, a validation split was held out for early stopping only; the test participant was never used for scaling, early stopping or model selection. Training used the Adam optimiser and categorical cross-entropy loss, with early stopping (patience = 5). No class weighting or resampling was applied; the consequences of that choice are analysed in Section 5.1. The training configuration is summarised in Table 2.

3.4. Edge Deployment and Hardware Integration

The system was transitioned from a workstation environment to a portable edge-AI node. The hardware stack is divided into two primary units.

3.4.1. Portable Sensing Node

An ESP32 microcontroller serves as the primary data aggregator, interfaced with an MPU-9250 IMU and a MAX30102 pulse sensor. To prevent bus contention and clock stretching issues, the sensors are isolated on two independent I^2C buses (400 kHz). The node is powered by a 2000 mAh Li-ion battery, utilising a synchronous boost converter to maintain a stable 5 V rail. The power consumption and battery runtime of the node and the gateway are not characterised in this work (Section 5.5).

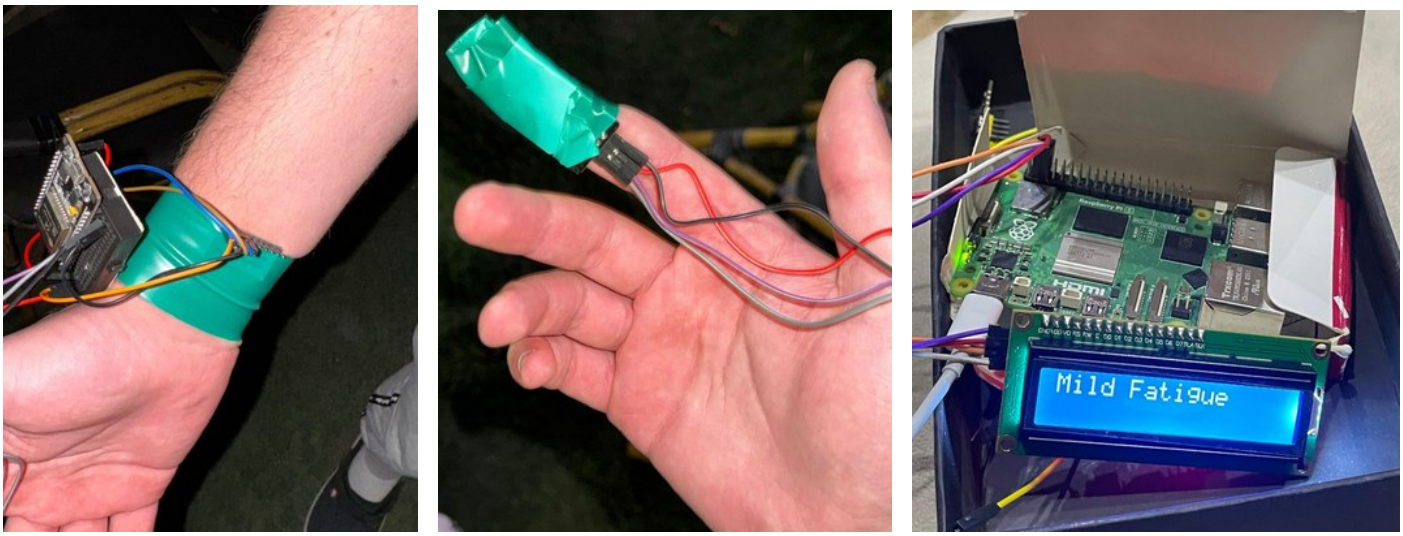
3.4.2. Inference Gateway

A Raspberry Pi 5 (8 GB RAM) acts as the inference engine. Data is streamed from the ESP32 via UDP packets over Wi-Fi every 200 ms. The Python-based inference daemon

performs real-time scaling and feeds the 10-element feature vector into the pre-trained FNN model. Three latency quantities are distinguished throughout this paper:

- Model inference latency (10 ms per sample): from the call to the model with a scaled feature vector to the returned class probabilities, timed on the gateway.
- Gateway processing latency (42 ms): from UDP packet receipt on the gateway to completion of the LCD write, covering deserialisation, scaling, inference and display.
- Wireless round-trip time (RTT; 180 ms mean after tuning): the round trip between the ESP32 and the gateway over the Wi-Fi link (Section 4.3).

The original manuscript also labelled the 180 ms figure as the "end-to-end sensor-to-display latency". That label was imprecise: the figure is the mean RTT, and it does not include the up to 200 ms during which samples accumulate on the node before a packet is sent. A dedicated sensor-to-display measurement is left for future work. Feedback is presented to the user through an I^2C character LCD. The assembled prototype is shown in Figure 4.



(**a**) Wrist-worn sensing node (**b**) MAX30102 PPG on fingertip (**c**) Raspberry Pi 5 inference gateway with LCD

Figure 4. Assembled hardware prototype: (**a**) the ESP32-based sensing node strapped to the wrist; (**b**) the MAX30102 photoplethysmography sensor on the fingertip; and (**c**) the Raspberry Pi 5 gateway displaying the inferred fatigue state on the character LCD.

3.5. Layered IoT Architecture

The hardware described above is deliberately organised as a layered IoT stack rather than as a single monolithic device, so that each tier can be replaced or extended without disturbing the others. Figure 5 sets out the resulting five-layer reference architecture and marks explicitly which layers are implemented and benchmarked in this work and which are specified but deferred. The present prototype implements only the physical, communication and edge-intelligence layers. The upper two layers describe how a Digital Twin could later be attached; they are not implemented or evaluated here.

The separation matters for two reasons. First, it isolates the two engineering problems that turned out to dominate the prototype: the sensing layer determines what can be measured, whereas the communication layer—not the model—determined the latency the user actually experiences, as Section 4.3 shows. Second, it makes the scope of the present contribution explicit. Layers 1 to 3 are delivered and measured here. Layers 4 and 5 define the interface that a Digital Twin backend would consume and are discussed in Section 5.4, but no virtual athlete model is instantiated in this work.

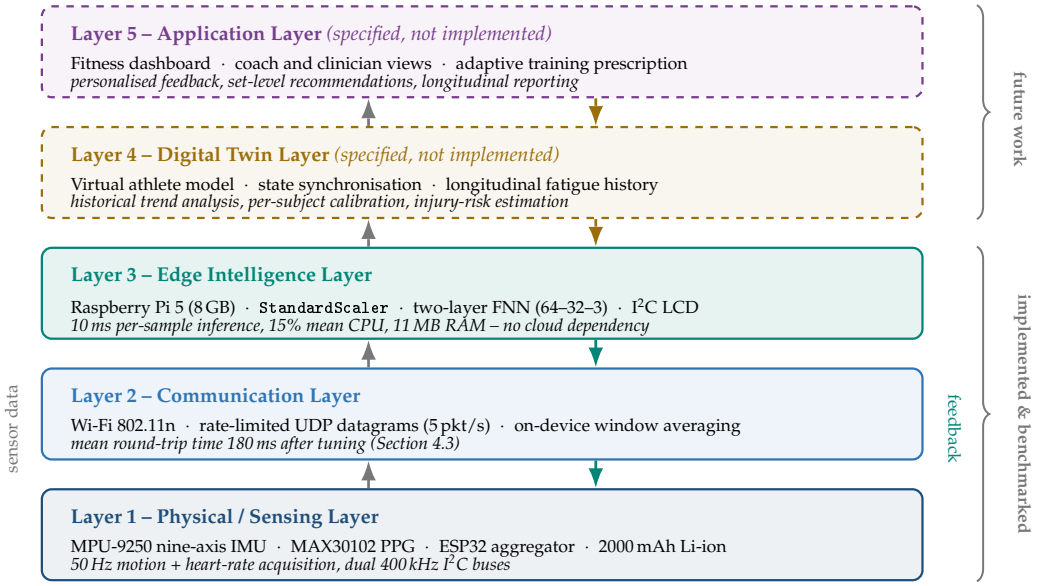


Figure 5. Proposed layered IoT architecture for the smart fitness system. Layers 1–3 (physical, communication and edge intelligence) are fully implemented and benchmarked in this work; the Digital Twin and application layers are specified here as the intended extension path and are drawn dashed to indicate that they are not part of the present prototype. Solid upward arrows carry sensor and inference data; downward arrows carry configuration and feedback.

3.6. Evaluation Metrics

Model performance was quantified using Accuracy, Precision, Recall, and the F_1-score. Specifically, Accuracy was calculated as the proportion of correctly classified instances across all fatigue states:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Because accuracy is misleading under class imbalance, the binary deployment results additionally report balanced accuracy (the mean of the per-class recalls) and the Matthews correlation coefficient (MCC), which is 0 for a classifier no better than chance and 1 for a perfect one:

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4}$$

Two reference points are reported with them: a majority-class predictor and a random predictor. For LOSO-CV, the mean, the standard deviation and a t-based 95% confidence interval across the ten folds are reported. For the deployed proportions, 95% Wilson score intervals are given; these treat windows as independent and are therefore optimistic, since consecutive windows from the same participant are correlated.

Two evaluation regimes are reported and kept distinct throughout. The first is the offline LOSO-CV protocol of Section 4.1.2, in which the three-class model is validated fold-by-fold on held-out participants. The second is the deployed end-to-end evaluation of Section 4.4, in which the trained model runs on the edge node and its outputs are scored under a binary *no fatigue*/*fatigue* collapse. These two regimes answer different questions and yield different numbers; conflating them would overstate deployed performance. Furthermore, edge performance was validated through *Inference Latency* (ms), *Throughput* (samples/s), and *CPU Utilisation* to ensure the system remains within real-time operational constraints.

4. Results

This section summarises the key empirical findings of the proposed fatigue-detection system, including model performance, edge deployment benchmarks, wireless optimisation, and full end-to-end evaluation.

4.1. Model Development Results

4.1.1. Feature Selection

Correlation analysis identified the most informative motion features with respect to fatigue state. Angular velocity components g_y and g_z showed minimal correlation and were excluded (Section 3.2). The feature correlation structure is shown in Figure 6. The strongest correlates are magnetometer components (m_z, $r = -0.68$; m_y, $r = 0.47$). A magnetometer measures orientation relative to the Earth's magnetic field, which depends on the room, the participant's heading and nearby metal, not on fatigue as such. Its high correlation here more plausibly reflects systematic changes in wrist orientation as sets progress, or a recording artefact, than a physiological signal, and it is unlikely to transfer to a different room and sensor. We return to this in Section 5.1.

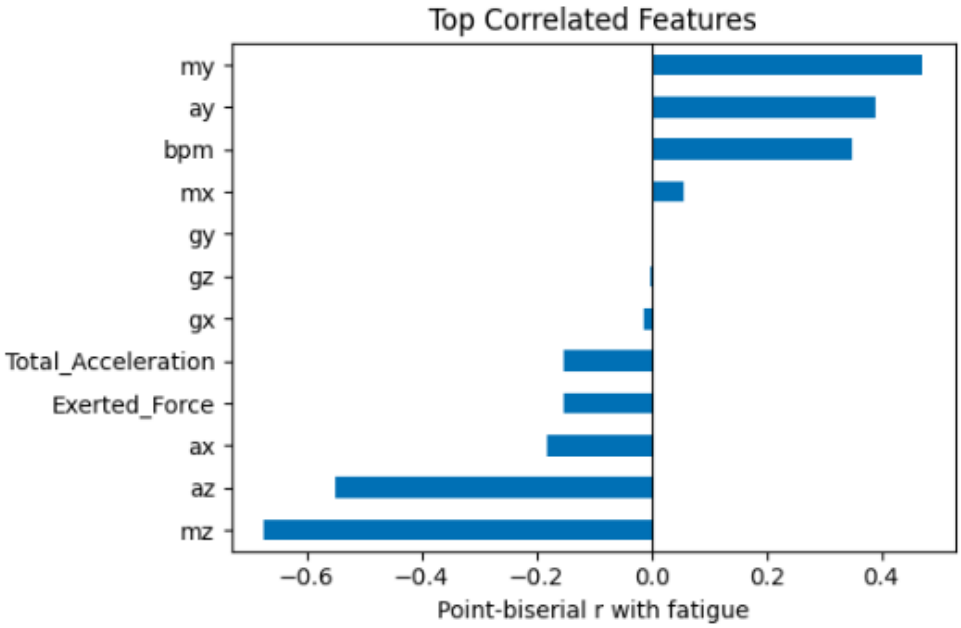


Figure 6. Point-biserial correlation of the candidate motion and physiological features with the fatigue label. Weakly correlated angular-velocity components (g_y, g_z) were dropped from the final feature vector (Table 1).

4.1.2. Leave-One-Subject-Out Cross-Validation

Table 3 summarises subject-wise performance under LOSO-CV. These are offline, three-class results obtained before deployment; the deployed binary results appear separately in another table.

Six participants achieved ≥90% accuracy, while three performed poorly (≤45%), indicating substantial inter-subject variability. The mean values in Table 3 are therefore an average over a strongly bimodal distribution and should be read as such rather than as a single expected operating point. The 95% confidence interval of the mean accuracy (0.59–0.95) is wide, and with ten participants the expected accuracy for a new user cannot be estimated more precisely than this. Per-class results within each fold were not retained from the original runs and are therefore not reported.

Table 3. LOSO-CV performance per subject (three-class, offline). Precision, recall and F_1 are support-weighted averages over the three classes within each fold. SD is the sample standard deviation across the ten folds; the 95% CI uses the t distribution with 9 degrees of freedom.

Subject	Accuracy	Precision	Recall	F_1
S1	0.95	0.95	0.95	0.95
S2	0.40	0.65	0.40	0.31
S3	0.91	0.92	0.91	0.91
S4	0.97	0.97	0.97	0.97
S5	0.40	0.53	0.40	0.28
S6	0.98	0.98	0.98	0.98
S7	0.90	0.92	0.89	0.89
S8	0.45	0.51	0.45	0.47
S9	0.81	0.73	0.81	0.77
S10	0.91	0.91	0.91	0.91
Mean	0.77	0.81	0.77	0.75
SD	0.25	0.19	0.25	0.28
95% CI of mean	0.59–0.95	0.68–0.94	0.59–0.94	0.54–0.94
Median	0.91	0.92	0.90	0.90

Figure 7 shows the training and validation accuracy over epochs, averaged across the LOSO-CV folds. The validation curve is not a repeated validation of a finished model. During training, the model is evaluated once at the end of every epoch on the validation split drawn from the training participants. This per-epoch curve is what early stopping monitors: training stops when the monitored validation metric has not improved for five consecutive epochs. The held-out test participant is evaluated only once, after training has finished.

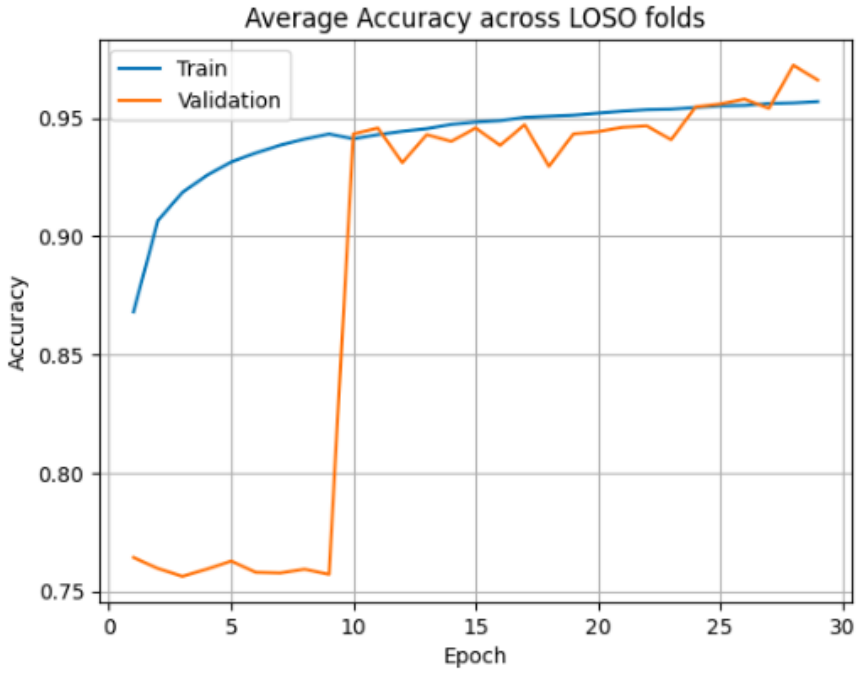


Figure 7. Training and validation accuracy across epochs, averaged across the LOSO-CV folds. The validation accuracy is computed once per epoch on a validation set held out from the nine training participants and is used only for early stopping (patience = 5).

4.2. Edge Deployment Results

Inference was benchmarked on a Raspberry Pi 5 (8 GB, 64-bit OS). Latencies are mean values. The throughput of 100 samples/s is the maximum rate at which the gateway can

classify samples sequentially at 10 ms each, not the sensor sampling rate. Fatigue develops over repetitions lasting about 2 s, so this rate is far above what the application needs. Its purpose is to show headroom: the deployed system produces 5 decisions/s, using about 5% of this capacity. Edge inference benchmark on Raspberry Pi 5 is summarised in Table 4.

Table 4. Edge inference benchmark on Raspberry Pi 5.

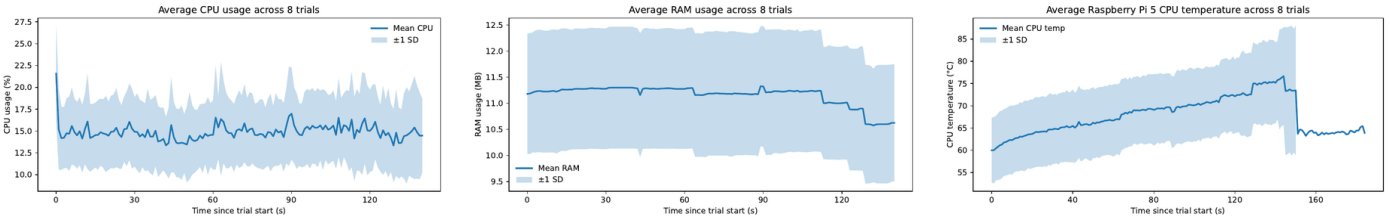
Metric	Value
Model inference latency (per sample)	10 ms
Gateway-side processing latency	42 ms
Wireless round-trip time (mean, after tuning)	180 ms
Throughput	100 samples/s

Resource utilisation across all trials is summarised in Table 5. CPU usage and temperature were logged on the gateway during continuous operation across the trials. The RAM figure refers to the memory used by the inference application (interpreter, model and scaler), not to the whole system including the operating system.

Table 5. Edge-computing performance metrics.

	Mean	Std	P95	Units
CPU usage	15.05	4.65	22.85	%
RAM usage	11.22	1.08	12.60	MB
CPU temperature	67.50	8.25	82.05	°C

The results confirm stable real-time inference with low CPU and memory overhead. Figure 8 plots the CPU utilisation, memory footprint, and CPU temperature averaged across all trials.



(**a**) CPU utilisation (**b**) RAM usage (**c**) CPU temperature

Figure 8. Edge resource usage on the Raspberry Pi 5 averaged across trials: (**a**) CPU utilisation, (**b**) memory footprint, and (**c**) CPU temperature. Utilisation and memory remain low throughout continuous operation.

4.3. Wireless Communication Results

Three optimisation trials were conducted to minimise feedback delay. Wireless-link optimisation results are summerised in Table 6.

Table 6. Wireless-link optimisation results.

Trial	Tx Rate (pkt/s)	Mean RTT (ms)
Baseline (unlimited)	25	15,000
Rate-limited	5	180
Rate-limited + averaging	5	180

Limiting transmission to 5 packets/s reduced round-trip time from 15 s to 180 ms. On-device averaging preserved signal resolution without increasing latency. The 15 s RTT

at the unlimited rate is caused by queueing: packets were produced faster than the link and receiver drained them, so each packet waited behind a growing backlog. Latency distributions, jitter and packet loss were not recorded in these trials and are not reported (Section 5.5).

4.4. End-to-End Experimental Results

For final evaluation, the problem was reformulated as binary classification: *no fatigue* vs. *fatigue* (mid and full fatigue merged, Section 3.2). The class distribution of the source dataset is shown in Figure 9, in which Class 0, Class 1 and Class 2 are no fatigue, mid fatigue and full fatigue, respectively, and the vertical axis counts 50 Hz samples. Within each subject the no fatigue label (class 0) dominates the training corpus, at roughly 60–70% of samples for eight of the ten participants. This skew is one candidate cause of the results below; others are discussed in Section 5.1.

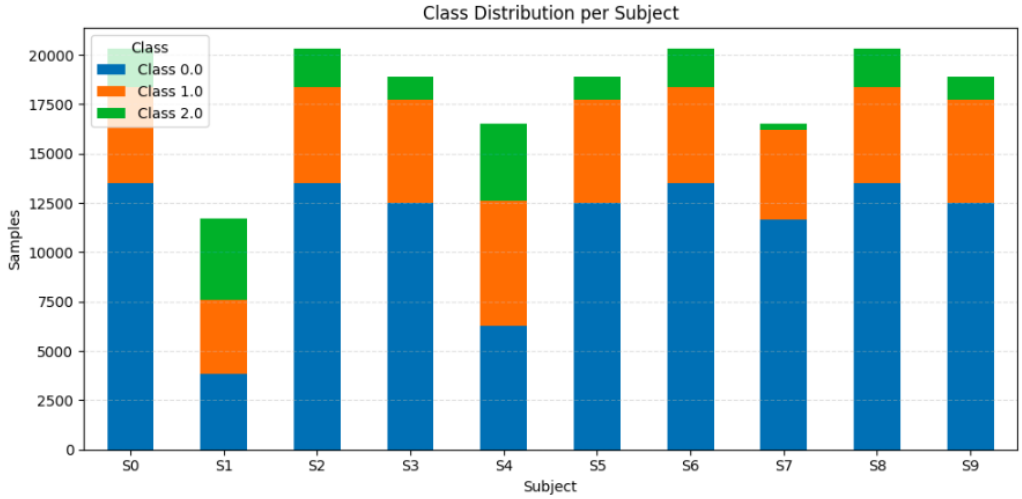


Figure 9. Class distribution per subject in the training data (10-participant subset of the source dataset; subjects indexed S0–S9 in this figure correspond to S1–S10 in Table 3). Class 0 = no fatigue, Class 1 = mid fatigue, and Class 2 = full fatigue; counts are 50 Hz samples. The no fatigue class dominates for most participants, which biases the learned prior once mid and full fatigue are merged into a single *fatigue* class at deployment.

Key observations.

- High recall for no fatigue (91%; 95% CI 90–92%) arises mainly because the model predicts *no fatigue* for 90% of all windows, not because it recognises rest.
- Very low recall for fatigue (10%; 95% CI 9–11%): nine out of ten fatigued windows are missed.
- Balanced accuracy is 0.51 and MCC is 0.02, i.e., the deployed classifier performs at chance. Its accuracy (0.44) is lower than that of a predictor that always outputs *fatigue* (0.58).
- Overall macro F_1 score: 0.38, against a mean LOSO-CV accuracy of 0.77 in the offline three-class setting (Table 3). The two regimes are not directly comparable, and the gap between them is analysed in Section 5.1.

The model exhibits conservative behaviour biased toward predicting the non-fatigued class. Notably, this is not simple majority-class bias with respect to the evaluation set: in Table 7 the fatigued class is the *larger* of the two (2988 versus 2184 windows). The bias matches the training distribution, but this study cannot separate it from the sensor differences discussed in Section 5.1. This behaviour is visible in the confusion matrix in Figure 10, where most fatigued windows are misclassified as non-fatigued.

Table 7. Deployed binary classification performance across all trials (5172 annotated windows).

	Precision	Recall	F_1	Support
No fatigue	0.426	0.912	0.580	2184
fatigue	0.610	0.101	0.173	2988
Macro Avg.	0.518	0.506	0.377	5172

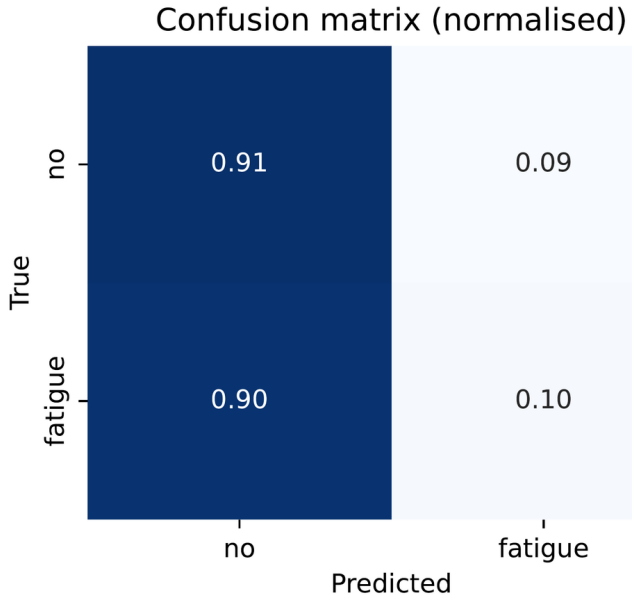


Figure 10. Confusion matrix for the binary *no fatigue* vs. *fatigue* task across all trials. The model correctly identifies most non-fatigued windows but misses the majority of fatigued ones, consistent with the training-side class skew and with differences between training and deployment sensors.

4.5. Comparison with Classical Models

On this dataset, the source study compared the FNN with random forest, decision tree and logistic regression classifiers using repetition-level features, and the FNN performed best in both the cross-subject and the subject-specific settings [8]. That comparison motivated the choice of model here. Classical baselines and class-weighted training were not re-run under the present per-sample protocol and deployment setting; Table 8 gives the majority-class and chance reference points against which the deployed model is judged, and Section 5.5 lists the missing comparisons.

Table 8. Deployed binary performance with reference points. Counts reconstructed from Table 7: TN = 1992, FP = 192, FN = 2686, TP = 302 (fatigue = positive). Intervals are 95% Wilson intervals.

Predictor	Accuracy	Balanced acc.	Macro F_1	MCC
Deployed FNN	0.444 (0.430–0.457)	0.507	0.377	0.02
Majority class (always *fatigue*)	0.578	0.500	0.366	0
Random (uniform)	0.500	0.500	≈0.50	0

4.6. Summary of Findings

- Offline, three-class subject-independent accuracy averaged 77% under LOSO-CV (mean precision 0.81, recall 0.77, F_1 0.75), with strong inter-subject variance (SD 0.25; 95% CI 0.59–0.95).
- Deployed, binary end-to-end performance over 5172 windows reached macro precision 0.52, recall 0.51 and F_1 0.38, with balanced accuracy 0.51 and MCC 0.02, i.e., chance level.

- Real-time inference achieved 10 ms model latency per sample and 42 ms gateway-side processing latency.
- Wireless optimisation reduced the round-trip time from 15 s to 0.18 s.
- The deployed classifier does not detect fatigue reliably: 90% of fatigued windows are missed.

5. Discussion

The results of this study are mixed. From an engineering perspective, the system achieves its main objective: real-time inference is performed entirely on the device using low-cost consumer hardware, with model and gateway latencies well below 200 ms. The classification results do not support the same conclusion. Offline, the model generalises to most but not all held-out participants; deployed, it performs at chance on the fatigue class. To answer directly the question of whether the deployed system is a reliable fatigue monitor: in its current form it is not. What the study demonstrates is a working, measured edge pipeline, and a quantified failure of a public-dataset model to transfer to it.

5.1. Classification Performance

Cross-subject accuracy varied a lot. Six of the ten subjects in the LOSO-CV folds came in above 90% accuracy. Three came in around 40–45%, which is barely above chance for a three-class problem. This kind of inter-subject spread is well documented in the fatigue literature—people have different biomechanics, fitness levels, and fatigue tolerance, and a model trained on nine other people will not always cover the tenth [42]. The original Elshafei and Shihab study [8] reported 88% cross-subject accuracy on the full 20-subject dataset; we worked with roughly half that cohort, which directly limits the diversity available for training in each LOSO-CV fold. Elshafei et al. [31] have separately shown that fatigue-induced kinematic changes degrade activity recognition models unless those models have fatigue-specific features built in, and our experience supports that.

The larger discrepancy is between the offline LOSO-CV numbers and the deployed end-to-end numbers, and it is worth being precise about it because the two are easy to conflate. Table 3 reports a three-class model validated on held-out participants offline: mean accuracy 0.77, mean F_1 0.75. Table 7 reports the same trained models running on the edge node, scored after mild and severe fatigue, are merged into one class: macro F_1 0.38. The edge port itself is unlikely to be the cause: the same trained model and preprocessing run on the Pi as on the workstation. The drop must therefore come mainly from the data. Three explanations are possible, and this study cannot fully separate them because the offline and deployed evaluations used different windows from different recordings:

1. Training-side class skew, discussed in the next paragraph.
2. Sensor and placement shift. The model was trained on the source study's IMU and smartwatch heart rate and deployed on an MPU-9250 and a fingertip MAX30102. Axis orientation, mounting position, calibration, full-scale range and the heart-rate algorithm all differ. The two most fatigue-correlated inputs in training are magnetometer axes (Figure 6), which depend on the local magnetic environment and heading and are the least likely of all inputs to transfer between laboratories.
3. Label shift. Fatigue labels in the deployment trials need not align exactly with the per-repetition RPE labels of the training data, particularly near the boundary between non-fatigued and fatigued repetitions.

Replaying the deployment recordings through the offline pipeline, and the reverse, would separate (1) from (2); retraining without the magnetometer axes would test the specific concern about them. These checks were not performed in this study and are listed in the Section 5.

When we collapsed the three-class problem to binary, the picture became sharper but also more lopsided. The model hit 91% recall on the non-fatigued class and 10% on the fatigued class: a classifier that has effectively learned to predict "not tired" most of the time. The obvious explanation, majority-class bias with respect to the evaluation set, is the wrong one here. In the evaluation set the fatigued class is actually the larger of the two, with roughly 2988 fatigued windows against 2184 non-fatigued. The bias matches the training side instead. In the source dataset the no fatigue label dominates within each subject (Figure 9), so a network trained with unweighted categorical cross-entropy learns a prior that no longer matches the deployment-time mix once mild and severe fatigue are merged into a single positive class. That is a train/deploy distribution shift layered on top of an imbalance problem, and training with unweighted cross-entropy commonly produces this failure mode. Class imbalance was not addressed in the original training because the first objective was to reproduce the unweighted FNN of the source study [8] as a reference before modifying it; in retrospect, given the known label distribution, a class-weighted variant should have been part of the original evaluation. It is identified as the first step of future work.

Weighted cross-entropy or resampling, the standard remedies for class imbalance, addresses only the first of the three explanations, and how much of the gap it would close has not been measured. Whatever part of the gap is due to sensor or label shift cannot be fixed by rebalancing. These require calibration data recorded on the deployment hardware, and subject-adaptive calibration [42], which the bimodal LOSO-CV distribution in Table 3 also points to.

Generalisation to new users. With ten participants, the LOSO-CV estimate is imprecise (95% CI 0.59–0.95), and the per-subject results are bimodal rather than scattered around the mean: the model either works well for a person or performs near chance. Ten participants are not enough to tell whether the three poorly predicted participants are rare cases or a sizeable fraction of users. The current evidence therefore does not support a claim of reliable generalisation to new users.

5.2. Edge Performance and Latency

The hardware side of the system held up well. Model inference came in at 10 ms per sample on the Raspberry Pi 5, the full gateway-side pipeline—deserialisation, scaling, inference and LCD update—completed in 42 ms, and the mean wireless round-trip time after tuning was 180 ms. That puts the system inside the responsiveness window that matters for real-time feedback during exercise [9,10,50]. Edge health-monitoring pipelines on comparably low-cost hardware have been reported by Srivastava et al. [13] and Dewanto et al. [15].

The wireless story is more interesting than we expected. Running UDP at its default rate produced round-trip times around 15 s, which made the system unusable. We tracked this down to buffer queue saturation, a known failure mode on shared Wi-Fi [9], and rate-limiting transmission to 5 packets/s collapsed RTT to 180 ms. The model never changed. What this said to us is that the bottleneck in this kind of edge deployment is rarely the model—it is the link. That matches the wider observation from Patel et al. [7] and Majumder et al. [43] that wireless data management is the persistent weakness in wearable systems. CPU and RAM utilisation on the Pi sat at 15% and 11 MB, respectively, across all trials. The Pi 5 has a lot of headroom for what we are asking it to do. Edge-computing surveys describe the same two-tier arrangement of sensor nodes and more capable gateways [18], and we end up with a consistent conclusion: the architecture is right; the protocol-level tuning is what does the work.

5.3. Implications

On the systems side, the results show that live inference with a model of this size does not need the cloud: the compute fits comfortably on a single-board computer, latency is set by the wireless link rather than the model, and the data need not leave the local network. This does not yet mean the system improves exercise safety, which would require a classifier that detects fatigue reliably and a user study. The remaining work is on the modelling side: more subjects, class-balanced training, online per-user calibration, and—where the dataset supports it—multimodal sensor fusion that brings sEMG or ECG into the input alongside the IMU [20,23,27,38]. The hardware also matters less than people sometimes assume. The ESP32 + Raspberry Pi 5 + MPU-9250 + MAX30102 stack is built from parts that anyone can order, with no custom PCB, which is part of the point: a usable wearable fatigue monitor should not require a hardware team [44,45,47,48]. Baig et al. [52] make a similar point: accessibility tends to matter as much as raw performance for wearable systems that are actually used. As noted in the Introduction, this architecture is a standard one; its value here is as a reproducible, low-cost testbed for measuring how fatigue models behave outside the dataset they were trained on.

5.4. Relation to Digital Twin Research

No Digital Twin is implemented or validated in this work, and we make no Digital Twin claim. The system has no virtual athlete model, no state maintained across sessions, and no bidirectional synchronisation between a physical and a virtual entity. The most precise description is that the implemented layers are *designed for* later Digital Twin integration. We use the term "Digital-Twin-ready" only in this narrow sense, and only for three properties that the sensing layer of a twin requires and that this prototype actually has and measures:

1. A continuous, timestamped stream of physiological and kinematic state at a known rate (5 updates/s);
2. A measured, bounded latency from sensing to state update (Tables 4 and 6); and
3. Local state estimation (the classifier output) that a twin could consume as an observation.

These properties are necessary for a twin but far from sufficient. Property 3 is currently weak: a state estimate that misses 90% of fatigued windows would give a twin a misleading picture of the athlete. Three further steps would be needed before the system could be called a Digital Twin: a persistent per-subject state model that survives across sessions; bidirectional synchronisation, so that twin-side calibration can update the edge classifier at runtime; and a validation protocol that scores the twin's predictions against measured outcomes.

5.5. Limitations

There are a few things worth being honest about. First, ten subjects is not many. Cross-subject accuracy will improve with a larger and more diverse cohort, but we cannot claim with confidence that the present numbers generalise beyond this dataset [8]; only half of the available 20-participant dataset was used. Second, we did not include sEMG. IMU kinematics are a good proxy for fatigue but not a direct measurement of it; adding sEMG would likely close some of the gap on the minority class [5,6]. Third, the binary reformulation throws away the three-class fatigue staging that is more useful for actual training prescription [4], and, as Section 5.1 argues, it is also the point at which the train/deploy prior mismatch becomes damaging. Fourth, the model was trained without class weighting, and neither class-weighted training nor classical baselines (logistic regression, SVM, and random forest) were re-evaluated under the present protocol; how far rebalancing would recover fatigue recall is therefore unknown. Fifth, the model was trained on one sensor setup and deployed on another; this sensor and placement shift, including the reliance

on magnetometer axes, was not isolated experimentally. Sixth, classification is per sample, using instantaneous values, whereas the source study used repetition-level statistical features; temporal features over a repetition may be more robust and were not explored. Seventh, testing was done indoors with reliable Wi-Fi. Outdoor deployment with weaker connectivity and variable temperature would change the latency and thermal numbers, and we have not stress-tested for that. Eighth, the work focuses on biceps concentration curls specifically. Whether the same pipeline ports cleanly to other exercises and other muscle groups needs its own evaluation [25,56]. Ninth, the systems evaluation reports mean latencies only: latency distributions, packet loss, a dedicated sensor-to-display measurement and the power consumption and battery runtime of the node and gateway were not characterised, so no claim is made about portability or energy efficiency. Tenth, per-class results for each LOSO fold were not retained. Eleventh, no user study was conducted, so no claim is made about effects on exercise safety or training outcomes. Finally, Layers 4 and 5 of the architecture in Figure 5 are unimplemented, so the paper makes no claim of demonstrated Digital Twin behaviour.

6. Conclusions

This work presented and evaluated a fully edge-based system for real-time biceps fatigue classification using wearable motion and pulse sensing. A nine-axis IMU and MAX30102 PPG sensor stream physiological data via an ESP32 gateway to a Raspberry Pi 5 inference engine, which runs a compact FNN classifier at 10 ms per-sample inference latency, 42 ms gateway-side processing latency and a 180 ms mean wireless round-trip time, consuming a mean of 15% CPU and 11 MB RAM.

The key findings are:

- Real-time inference with model and gateway latencies far below the 200 ms update interval is achievable on consumer-grade edge hardware with minimal computational resources, without any cloud connection.
- Protocol-level wireless optimisation (rate-limited UDP at 5 packets/s) reduces round-trip time from 15 s to 180 ms; in this pipeline, transmission management, not the model, dominates latency.
- Offline subject-independent LOSO-CV accuracy averaged 77% on the three-class task (95% CI 0.59–0.95), with three of ten participants near chance.
- Deployed on new hardware, the same model performs at chance on the binary task (balanced accuracy 0.51, fatigue recall 0.10). The system is therefore not yet a reliable fatigue monitor. Training-side class skew, sensor and placement shift, and label shift are candidate causes.
- The implemented physical, communication and edge-intelligence layers provide a measured sensing foundation that a future Digital Twin could build on; no Digital Twin is implemented here.

The prototype shows that real-time on-device fatigue inference is feasible as a systems problem, and that the open problem is the transfer of the model to new sensors and new users. The next steps are calibration data recorded on the deployment hardware, class-balanced and repetition-level models, a larger cohort, and subject-adaptive personalisation [42]. Only once fatigue detection is reliable does it make sense to build the Digital Twin and application layers on top [16,53].

Author Contributions: Conceptualization, S.A.; methodology, S.A. and M.A.E.G.; software, M.A.E.G. and O.G.; validation, M.A.E.G. and D.Z.A.-H.; formal analysis, M.A.E.G. and O.G.; investigation, M.A.E.G.; data curation, M.A.E.G.; writing—original draft preparation, M.A.E.G. and O.G.; writing—review and editing, D.Z.A.-H., M.M.B., H.G. and S.A.; visualization, M.A.E.G.; project

administration, D.Z.A.-H.; and supervision, S.A., H.G. and M.M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The training data are the public dataset of Elshafei and Shihab [8], collected under the approval of the Concordia University Human Research Ethics Committee (protocol 30008716); their secondary use required no new approval. The deployment trials were conducted in accordance with the Declaration of Helsinki with consenting volunteers in a controlled laboratory environment at the German University in Cairo.

Informed Consent Statement: Informed consent for the public training dataset was obtained by the original investigators [8]. Informed consent was obtained from all participants in the deployment trials.

Data Availability Statement: The biceps muscle fatigue dataset used in this study is publicly available from Elshafei and Shihab [8] at https://zenodo.org/record/3698242 (accessed on 1 June 2025). The training, evaluation and edge-deployment code, the trained model and scaler, and the ESP32 firmware are available from the corresponding author on reasonable request.

Acknowledgments: The authors thank the anonymous reviewers, whose comments substantially improved the evaluation in this paper. The authors thank the German University in Cairo for laboratory and computational support.

Conflicts of Interest: Author Mirza Mansoor Baig is employed by Dedalus, Auckland, New Zealand. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BLE	Bluetooth Low Energy
CI	confidence interval
DT	Digital Twin
FNN	feedforward neural network
HAR	Human Activity Recognition
IMU	inertial measurement unit
IoT	Internet of Things
LOSO-CV	leave-one-subject-out cross-validation
MCC	Matthews correlation coefficient
MEC	Multi-Access Edge Computing
PPG	Photoplethysmography
ReLU	Rectified Linear Unit
RF	random forest
RPE	Rate of Perceived Exertion
RTT	round-trip time
SBC	single-board computer
sEMG	surface electromyography
UDP	User Datagram Protocol

References

1. Enoka, R.M.; Duchateau, J. Translating Fatigue to Human Performance. *Med. Sci. Sports Exerc.* **2016**, *48*, 2228–2238. [CrossRef] [PubMed] [PubMed Central]
2. Mair, S.D.; Seaber, A.V.; Glisson, R.R.; Garrett, W.E., Jr. The Role of Fatigue in Susceptibility to Acute Muscle Strain Injury. *Am. J. Sport. Med.* **1996**, *24*, 137–143. [CrossRef] [PubMed]
3. Kellmann, M. Preventing Overtraining in Athletes in High-Intensity Sports and Stress/Recovery Monitoring. *Scand. J. Med. Sci. Sport.* **2010**, *20*, 95–102. [CrossRef] [PubMed]
4. Halson, S.L. Monitoring Training Load to Understand Fatigue in Athletes. *Sports Med.* **2014**, *44*, 139–147. [CrossRef] [PubMed]
5. Cifrek, M.; Medved, V.; Tonković, S.; Ostojić, S. Surface EMG Based Muscle Fatigue Evaluation in Biomechanics. *Clin. Biomech.* **2009**, *24*, 327–340. [CrossRef] [PubMed]

6. Subasi, A.; Kiymik, M.K. Muscle Fatigue Detection in EMG Using Time–Frequency Methods, ICA and Neural Networks. *J. Med. Syst.* **2010**, *34*, 777–785. [CrossRef] [PubMed]
7. Patel, S.; Park, H.; Bonato, P.; Chan, L.; Rodgers, M. A Review of Wearable Sensors and Systems with Application in Rehabilitation. *J. Neuroeng. Rehabil.* **2012**, *9*, 21. [CrossRef] [PubMed]
8. Elshafei, M.; Shihab, E. Towards Detecting Biceps Muscle Fatigue in Gym Activity Using Wearables. *Sensors* **2021**, *21*, 759. [CrossRef] [PubMed]
9. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
10. Abdellatif, A.A.; Mohamed, A.; Chiasserini, C.F.; Tlili, M.; Erbad, A. Edge Computing for Smart Health: Context-Aware Approaches, Opportunities, and Challenges. *IEEE Netw.* **2019**, *33*, 196–203. [CrossRef]
11. Yeung, T. What Is Edge AI and How Does It Work? 2022. Available online: https://blogs.nvidia.com/blog/what-is-edge-ai/ (accessed on 22 May 2025).
12. Deng, S.; Zhao, H.; Fang, W.; Yin, J.; Dustdar, S.; Zomaya, A.Y. Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence. *IEEE Internet Things J.* **2020**, *7*, 7457–7469. [CrossRef]
13. Srivastava, P.; Shah, N.; Jaiswal, K. Microcontroller-Based EdgeML: Health Monitoring for Stress and Sleep via HRV. *Eng. Proc.* **2024**, *78*, 3. [CrossRef]
14. David, R.; Duke, J.; Jain, A.; Janapa Reddi, V.; Jeffries, N.; Li, J.; Kreeger, N.; Nappier, I.; Natraj, M.; Wang, T.; et al. TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems. In Proceedings of the Machine Learning and Systems, Online, 5–9 April 2021; Volume 3, pp. 800–811.
15. Dewanto, S.; Alexandra, M.; Surantha, N. Heart Rate Monitoring with Smart Wearables Using Edge Computing. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 141–148. [CrossRef]
16. Afifi, S.; ElGhazouly, M.; Leon, A.E.-m.; Elmeligy, H.; GholamHosseini, H.; Baig, M.M. Towards Personalized Digital Healthcare: An AI-Driven IoT-Cloud Digital Twin System for Remote Patient Monitoring. In Proceedings of the 2026 6th International Conference on Electrical, Computer and Energy Technologies (ICECET), Rome, Italy, 6–9 July 2026; pp. 1–5. [CrossRef]
17. Afifi, S.; Naem, M.; ElGhazouly, M.; Baig, M.M.; GholamHosseini, H. Smart IoT-Based Mobile App for Continuous Patient Monitoring. In Proceedings of the 2025 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA), Antalya, Türkiye, 7–9 August 2025; pp. 1–7. [CrossRef]
18. Yu, W.; Liang, F.; He, X.; Hatcher, W.G.; Lu, C.; Lin, J.; Yang, X. A Survey on the Edge Computing for the Internet of Things. *IEEE Access* **2018**, *6*, 6900–6919. [CrossRef]
19. Wang, J.; Sun, S.; Sun, Y. A Muscle Fatigue Classification Model Based on LSTM and Improved Wavelet Packet Threshold. *Sensors* **2021**, *21*, 6369. [CrossRef] [PubMed]
20. Al-Mulla, M.R.; Sepulveda, F.; Colley, M. An Autonomous Wearable System for Predicting and Detecting Localised Muscle Fatigue. *Sensors* **2011**, *11*, 1542–1557. [CrossRef] [PubMed]
21. Marotta, L.; Buurke, J.H.; van Beijnum, B.J.F.; Reenalda, J. Towards Machine Learning-Based Detection of Running-Induced Fatigue in Real-World Scenarios: Evaluation of IMU Sensor Configurations to Reduce Intrusiveness. *Sensors* **2021**, *21*, 3451. [CrossRef] [PubMed]
22. Marotta, L.; Scheltinga, B.L.; van Middelaar, R.; Bramer, W.M.; van Beijnum, B.J.F.; Reenalda, J.; Buurke, J.H. Accelerometer-Based Identification of Fatigue in the Lower Limbs during Cyclical Physical Exercise: A Systematic Review. *Sensors* **2022**, *22*, 3008. [CrossRef] [PubMed]
23. Bangaru, S.S.; Wang, C.; Aghazadeh, F. Automated and Continuous Fatigue Monitoring in Construction Workers Using Forearm EMG and IMU Wearable Sensors and Recurrent Neural Network. *Sensors* **2022**, *22*, 9729. [CrossRef] [PubMed]
24. Chang, P.; Wang, C.; Chen, Y.; Wang, G.; Lu, A. Identification of Runner Fatigue Stages Based on Inertial Sensors and Deep Learning. *Front. Bioeng. Biotechnol.* **2023**, *11*, 1302911. [CrossRef] [PubMed]
25. Camomilla, V.; Bergamini, E.; Fantozzi, S.; Vannozzi, G. Trends Supporting the In-Field Use of Wearable Inertial Sensors for Sport Performance Evaluation: A Systematic Review. *Sensors* **2018**, *18*, 873. [CrossRef] [PubMed]
26. Merbah, J.; Carè, B.R.; Gorce, P.; Gadea, F.; Prince, F. A New Approach to Quantifying Muscular Fatigue Using Wearable EMG Sensors during Surgery: An Ergonomic Case Study. *Sensors* **2023**, *23*, 1686. [CrossRef] [PubMed]
27. Mu, D.; Li, F.; Yu, L.; Du, C.; Ge, L.; Sun, T. Study on Exercise Muscle Fatigue Based on sEMG and ECG Data Fusion and Temporal Convolutional Network. *PLoS ONE* **2022**, *17*, e0276921. [CrossRef] [PubMed]
28. Xu, L.; Yang, W.; Cao, Y.; Li, Q. Human Activity Recognition Based on Random Forests. In Proceedings of the 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, China, 29–31 July 2017; pp. 548–553. [CrossRef]
29. Cust, E.E.; Sweeting, A.J.; Ball, K.; Robertson, S. Machine and Deep Learning for Sport-Specific Movement Recognition: A Systematic Review of Model Development and Performance. *J. Sport. Sci.* **2019**, *37*, 568–600. [CrossRef] [PubMed]

30. Soro, A.; Brunner, G.; Tanner, S.; Wattenhofer, R. Recognition and Repetition Counting for Complex Physical Exercises with Deep Learning. *Sensors* **2019**, *19*, 714. [CrossRef] [PubMed]
31. Elshafei, M.; Costa, D.E.; Shihab, E. On the Impact of Biceps Muscle Fatigue in Human Activity Recognition. *Sensors* **2021**, *21*, 1070. [CrossRef] [PubMed]
32. Nair, S.P.; Sica, M.; Tedesco, S.; Visentin, A. Machine Learning Physical Fatigue Estimation Approach Based on IMU and EMG Wearable Sensors. In Proceedings of the 32nd Irish Conference on Artificial Intelligence and Cognitive Science (AICS), Dublin, Ireland, 9–10 December 2024.
33. Otálora, S.; Segatto, M.E.V.; Monteiro, M.E.; Múnera, M.; Díaz, C.A.R.; Cifuentes, C.A. Data-Driven Approach for Upper Limb Fatigue Estimation Based on Wearable Sensors. *Sensors* **2023**, *23*, 9291. [CrossRef] [PubMed]
34. Hammerla, N.Y.; Halloran, S.; Plötz, T. Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables. In Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI), New York, NY, USA, 9–15 July 2016; pp. 1533–1540.
35. Ronao, C.A.; Cho, S.-B. Human Activity Recognition with Smartphone Sensors Using Deep Learning Neural Networks. *Expert Syst. Appl.* **2016**, *59*, 235–244. [CrossRef]
36. Bayat, A.; Pomplun, M.; Tran, D.A. A Study on Human Activity Recognition Using Accelerometer Data from Smartphones. *Procedia Comput. Sci.* **2014**, *34*, 450–457. [CrossRef]
37. Zhang, M.; Sawchuk, A.A. A Feature Selection-Based Framework for Human Activity Recognition Using Wearable Multimodal Sensors. In Proceedings of the 6th International Conference on Body Area Networks (BodyNets), Beijing, China, 7–8 November 2011; pp. 92–98. [CrossRef]
38. Jiang, Y.; Hernandez, V.; Venture, G.; Kulić, D.; Chen, B.K. A Data-Driven Approach to Predict Fatigue in Exercise Based on Motion Data from Wearable Sensors or Force Plate. *Sensors* **2021**, *21*, 1499. [CrossRef] [PubMed]
39. Li, N.; Zhou, R.; Krishna, B.; Pradhan, A.; Lee, H.; He, J.; Jiang, N. Non-Invasive Techniques for Muscle Fatigue Monitoring: A Comprehensive Survey. *ACM Comput. Surv.* **2024**, *56*, 1–40. [CrossRef]
40. Kim, J.; Kim, H.; Lee, J.; Lee, J.; Yoon, J.; Ko, S.K. A Deep Learning Approach for Fatigue Prediction in Sports Using GPS Data and Rate of Perceived Exertion. *IEEE Access* **2022**, *10*, 103056–103064. [CrossRef]
41. Biró, A.; Cuesta-Vargas, A.I.; Szilágyi, L. AI-Assisted Fatigue and Stamina Control for Performance Sports on IMU-Generated Multivariate Times Series Datasets. *Sensors* **2024**, *24*, 132. [CrossRef] [PubMed]
42. Elshafei, M.; Costa, D.E.; Shihab, E. Toward the Personalization of Biceps Fatigue Detection Model for Gym Activity: An Approach to Utilize Wearables' Data from the Crowd. *Sensors* **2022**, *22*, 1454. [CrossRef] [PubMed]
43. Majumder, S.; Mondal, T.; Deen, M.J. Wearable Sensors for Remote Health Monitoring. *Sensors* **2017**, *17*, 130. [CrossRef] [PubMed]
44. Kodali, R.K.; Jain, V.; Bose, S.; Boppana, L. IoT Based Smart Security and Home Automation System. In Proceedings of the International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 29–30 April 2016; pp. 1286–1289. [CrossRef]
45. Islam, S.M.R.; Kwak, D.; Kabir, M.H.; Hossain, M.; Kwak, K.S. The Internet of Things for Health Care: A Comprehensive Survey. *IEEE Access* **2015**, *3*, 678–708. [CrossRef]
46. Naik, K.S.; Sudarshan, E. Smart Healthcare Monitoring System Using Raspberry Pi on IoT Platform. *ARPN J. Eng. Appl. Sci.* **2019**, *14*, 872–876.
47. Mischianti, R. MPU9250-DMP Arduino Library. 2024. GitHub Repository. Available online: https://github.com/xreef/MPU-9250-DMP_Library/blob/master/library.properties (accessed on 23 May 2025).
48. SparkFun Electronics. SparkFun MAX3010x Sensor Library. 2025. GitHub Repository. Available online: https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library (accessed on 23 May 2025).
49. Fastly. Edge Computing vs. Cloud Computing. Available online: https://www.fastly.com/learning/edge-computing-vs-cloud-computing (accessed on 22 May 2025).
50. Intel Corporation. What Is a Real-Time System? Available online: https://www.intel.com/content/www/us/en/learn/what-is-a-real-time-system.html (accessed on 22 May 2025).
51. Seshadri, D.R.; Li, R.T.; Voos, J.E.; Rowbottom, J.R.; Alfes, C.M.; Zorman, C.A.; Drummond, C.K. Wearable Sensors for Monitoring the Physiological and Biochemical Profile of the Athlete. *npj Digit. Med.* **2019**, *2*, 72. [CrossRef] [PubMed]
52. Baig, M.M.; Afifi, S.; GholamHosseini, H.; Mirza, F. A Systematic Review of Wearable Sensors and IoT-Based Monitoring Applications for Older Adults—A Focus on Ageing Population and Independent Living. *J. Med. Syst.* **2019**, *43*, 233. [CrossRef] [PubMed]
53. Baig, M.M.; Hobson, C.; GholamHosseini, H.; Ullah, E.; Afifi, S. Generative AI in Improving Personalized Patient Care Plans: Opportunities and Barriers Towards Its Wider Adoption. *Appl. Sci.* **2024**, *14*, 10899. [CrossRef]

54. GholamHosseini, H.; Baig, M.M.; Afifi, S.; Ullah, E. The Role of AI in Patient Monitoring Using Smart Medical Devices: Opportunities and Challenges. In *Handbook on Smart Health*; Augusto, J.C., Ed.; Studies in Health Technology and Informatics; IOS Press: Amsterdam, The Netherlands, 2025; Volume 330. [CrossRef] [PubMed]
55. Jeong, I.C.; Bychkov, D.; Searson, P.C. Wearable Devices for Precision Medicine and Health State Monitoring. *IEEE Trans. Biomed. Eng.* **2019**, *66*, 1242–1258. [CrossRef] [PubMed]
56. Adão Martins, N.R.; Annaheim, S.; Spengler, C.M.; Rossi, R.M. Fatigue Monitoring Through Wearables: A State-of-the-Art Review. *Front. Physiol.* **2021**, *12*, 790292. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.