International Journal of
PHYSICAL EDUCATION, FITNESS AND SPORTS

Electronic Supplementary Information

IoT-Enabled Multimodal Approach for Low-Latency Prediction of Elite Athlete Performance Dynamics

Mensah Sitti [a, *], Prince James Adeti [b]

[a] Department of Computer Science and Engineering University of Mines and Technology, Tarkwa, Ghana
[b] Ghana Atomic Energy Commission, Legon, Accra, Ghana
* E-mail: msitti@umat.edu.gh

DOI: https://doi.org/10.54392/ijpefs2618
Received: 29-11-2025; Revised: 17-02-2026; Accepted: 24-02-2026; Published: 02-03-2026

TVAE–Vanilla RNN Multimodal Fusion for Elite Athlete Performance Prediction

Input:

Streaming multimodal data:

EEG(t) ∈ ℝ32 // 32-ch, 256 Hz

ECG(t) ∈ ℝ // HRV from R–R intervals

IMU(t) ∈ ℝ6 // 3-axis accel + 3-axis gyro

Session metadata:

sport_id ∈ {1,…,6}, session_type ∈ {Training, Competition, Recovery}

Labels per 4-s epoch:

performance_score ∈ [0,1]

fatigue_level ∈ {Fresh, Moderate, High, Critical}

Output:

ŷ_perf – predicted performance score

ŷ_fat – predicted fatigue class

Deployed edge model with sub-5 ms inference latency

Stage 1 – Data acquisition and synchronisation

---

1 FOR each athlete a and session s DO
2 Stream EEG, ECG, IMU via BLE 5.0 → edge gateway
3 Time-stamp and buffer EEG(t), ECG(t), IMU(t)
4 Resample all streams to base_sync_rate = 256 Hz
5 Segment into non-overlapping 4-s epochs:
6 epoch_EEG, epoch_ECG, epoch_IMU
7 Attach labels performance_score, fatigue_level,

8 sport_id, session_type, context metadata

9 END FOR

Stage 2 – Signal pre-processing and feature extraction

2.1 EEG processing (per epoch)

10 FOR each epoch_EEG DO

11 EEG_hp ← high-pass Butterworth(0.5 Hz)

12 EEG_notch ← notch_filter(50 Hz)

13 EEG_clean ← artefact_removal_ICA(EEG_notch)

14 FOR each band b ∈ {Δ, θ, α, β, γ} DO

15 EEG_b ← bandpass_filter(EEG_clean, b)

16 P_b ← RMS_power_per_channel(EEG_b)

17 END FOR

18 θ/α ← mean(P_θ) / mean(P_α)

19 FAA ← frontal_alpha_asymmetry(F4, F3)

20 EEG_feat ← [Δ, θ, α, β, γ, θ/α, FAA]

21 END FOR

2.2 HRV processing (per epoch)

22 FOR each epoch_ECG DO

23 RR ← detect_RR_intervals(epoch_ECG)

24 RR_clean← correct_ectopics(RR)

25 MeanHR ← 60000 / mean(RR_clean)

26 SDNN ← std(RR_clean)

27 RMSSD ← rms_diff(RR_clean)

28 pNN50 ← percent_diff_gt_50ms(RR_clean)

29 RR_res ← resample(RR_clean, 4 Hz)

30 PSD_LS ← lomb_scargle(RR_res)

31 LF ← integrate(PSD_LS, 0.04–0.15 Hz)

32 HF ← integrate(PSD_LS, 0.15–0.40 Hz)

33 LF_HF ← LF / HF

34 SDNN_RMSSD ← SDNN / RMSSD

35 HRV_feat← [MeanHR, SDNN, RMSSD, pNN50, LF, HF, LF_HF, SDNN_RMSSD]

36 END FOR

2.3 IMU processing (per epoch)

37 FOR each epoch_IMU DO

38 IMU_cal ← apply_calibration(epoch_IMU)

39 IMU_lp ← lowpass_filter(IMU_cal, 40 Hz)

```
    IMU_dyn  ← remove_gravity(IMU_lp)
    Accel_RMS_xyz ← rms_per_axis(Accel)
    Gyro_RMS_xyz  ← rms_per_axis(Gyro)
    Accel_mag     ← mean_norm(Accel)
    Jerk_mean     ← mean_norm(d/dt Accel)
    Symmetry_idx  ← bilateral_symmetry(Accel, sport_id)
    Stride_freq   ← dominant_freq(Accel_vertical)
    Peak_accel    ← max_norm(Accel)
    IMU_feat      ← [Accel_RMS_xyz, Gyro_RMS_xyz,
                     Accel_mag, Jerk_mean,
                     Symmetry_idx, Stride_freq, Peak_accel]
END FOR
```

Stage 3 – Feature-Level Fusion And Normalisation

```
FOR each epoch e DO
    Context_feat ← [session_duration, reaction_time,
                    decision_accuracy, RPE, cognitive_load,
                    one_hot(sport_id), one_hot(session_type)]
    Fused_feat_e ← CONCAT(EEG_feat, HRV_feat, IMU_feat, Context_feat)
END FOR

[Train, Val, Test] ← stratified_split_by_fatigue(Fused_feat_e, labels)
[μ, σ] ← zscore_stats(Train.features)
NORMALISE all splits: x_norm = (x − μ) / σ
```

Stage 4 – Temporal Variational Autoencoder (Tvae)

Encoder:

```
h1      = Dense(128, ReLU) → BN → Dropout(0.30)(x_norm)
h2      = Dense(64,  ReLU) → BN → Dropout(0.25)(h1)
z_mean  = Dense(32, Linear)(h2)
z_logv  = Dense(32, Linear)(h2)
ε       ~ N(0, I)
z       = z_mean + exp(0.5 * z_logv) ⊙ ε
```

Decoder:

```
d1      = Dense(64, ReLU)(z)
```

```
d2      = Dense(128, ReLU)(d1)
x_hat   = Dense(33, Linear)(d2)

Loss:
recon_loss = MSE(x_norm, x_hat)
KL_loss    = −0.5 · mean(1 + z_logv − z_mean² − exp(z_logv))
L_TVAE     = recon_loss + β · KL_loss

Training:
FOR epoch = 1…E_TVAE DO
    FOR mini-batch Xb in Train.features_norm DO
        COMPUTE L_TVAE(Xb)
        UPDATE encoder, decoder parameters (Adam)
    END FOR
    EARLY_STOP(Val) IF no improvement
END FOR

Latent extraction:
FOR each sample x_norm in {Train, Val, Test} DO
    z_mean = Encoder(x_norm)
    STORE z_mean as latent feature
END FOR
```

Stage 5 – Vanilla RNN Predictive Head (Multi-Task)

```
// TVAE–RNN hybrid for performance + fatigue prediction
```

```
Sequence construction:
FOR each latent z_mean DO
    z_pad   ← zero_pad_to_length(33, z_mean)
    seq_in  ← reshape(z_pad, (T_seq=3, D_seq=11))
    sport_emb ← embed(sport_id)
    STORE (seq_in, sport_emb, perf_label, fatigue_label)
END FOR

RNN head:
h1 = SimpleRNN(128, return_sequences=True)(seq_in) → Dropout(0.25)
h2 = SimpleRNN(64,  return_sequences=True)(h1)     → Dropout(0.25)
h3 = SimpleRNN(32,  return_sequences=False)(h2)    → Dropout(0.20)
z_proj   = Dense(32, ReLU)(flatten(seq_in))        // skip from TVAE latent
h_skip   = h3 + z_proj
```

```
95  sport_vec= Flatten(Embedding(6, 8)(sport_emb))
96  m        = CONCAT(h_skip, sport_vec)              // 40-D
97  fc1      = Dense(64, ReLU)(m) → BN → Dropout(0.20)
98  fc2      = Dense(32, ReLU)(fc1)
99  ŷ_perf   = Dense(1,  Sigmoid)(fc2)               // regression
100 ŷ_fat    = Dense(4,  Softmax)(fc2)               // 4-class fatigue

Loss:
101 L_perf = MSE(performance_score, ŷ_perf)
102 L_fat  = CE(fatigue_level, ŷ_fat)
103 L_total = w_perf·L_perf + w_fat·L_fat

Training (TVAE encoder frozen):
104 FREEZE(encoder)
105 FOR epoch = 1…E_RNN DO
106     FOR mini-batch (seq_b, sport_b, y_perf_b, y_fat_b) in Train DO
107         (ŷ_perf_b, ŷ_fat_b) = RNN_head(seq_b, sport_b)
108         COMPUTE L_total
109         UPDATE RNN parameters (Adam)
110     END FOR
111     EARLY_STOP(Val) IF no improvement
112 END FOR
```

Stage 6 – Evaluation And Edge Deployment

```
113 COMPUTE RMSE, MAE, R², r for performance on Test
114 COMPUTE accuracy, macro-F1, confusion matrix, AUC for fatigue
115 MEASURE inference latency per sample on edge (TVAE_encoder+RNN_head)

116 IF (accuracy, F1, RMSE) outperform unimodal and single-architecture baselines
117    AND latency_per_sample < 5 ms THEN
118       DEPLOY model to Edge_Node for real-time athlete monitoring
119 END IF

END
```