Smart Pacer

Lorenzo Gandini
Internet of Things
University La Sapienza
A.Y. 2024/2025

June 4, 2025

Contents

1 Abstract

Running coaches increasingly rely on data-driven feedback to tailor training load in real time. *Smart Pacer* is a reinforcement-learning system that learns a pacing policy able to suggest—second by second—whether an athlete should *accelerate*, *keep the pace*, or *slow down*. The agent observes heart-rate and power zones, instantaneous slope extracted from a GPX track, workout phase, and a cumulative fatigue score; it then maximises long-term reward that simultaneously favours zone compliance and physiological coherence while penalising excessive fatigue. We implemented the agent with Q-learning, trained it on four canonical workouts (fartlek, progressions, endurance, recovery) and three athlete profiles (elite, runner, amateur), and streamed live cues via MQTT as a smartwatch surrogate. Preliminary experiments on realistic tracks confirm the emergence of sensible pacing patterns and smooth fatigue management, laying the groundwork for future field validation and richer physiological models.

2 Methodology

2.1 Markov Decision Process

The pacing problem is formalised as a finite Markov Decision Process

$$\langle \mathcal{S},\ \mathcal{A},\ \mathcal{P},\ r,\ \gamma \rangle,$$

where

- $\mathcal{S}$ is the Cartesian product of **(i)** heart-rate zone ($Z_1 - Z_5$), **(ii)** power zone ($Z_1 - Z_5$), **(iii)** categorical fatigue level (low, medium, high), **(iv)** workout phase (warm-up, push, recover, cool-down), **(v)** target zones, and **(vi)** slope label (uphill, flat, downhill);
- $\mathcal{A} = \{\text{slow down}, \text{keep going}, \text{accelerate}\}$;
- $\mathcal{P}$ is implicit in the physiological update rules of `RunnerEnv`;
- r is a shaped reward that combines zone-matching bonuses, fatigue penalties, slope/action coherence, and phase-specific incentives;
- $\gamma = 0.99$ discounts future returns.

2.2 Learning Algorithm

We employ tabular Q-learning with an ϵ-greedy exploration schedule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big],$$

using $\alpha \in [0.05, 0.10]$ and logarithmically decaying ϵ ($0.4 \rightarrow 0.01$). Training runs are executed off-line for 2 000 episodes per athlete–workout pair; the best performing Q-tables are persisted and loaded by `main.py` for on-line inference.

2.3 Fatigue Model

Fatigue is updated every second through a dual process:

1. **Accumulation** during warm-up and push phases scales with HR zone, power zone, time spent above threshold, workout typology, and the athlete's FTP / weight ratio.
2. **Decay** during recover and cool-down follows an exponential–sigmoid law capped by a fitness-dependent floor.

The resulting score is clipped to $[0, 10]$ and mapped to qualitative levels that condition subsequent rewards.

3 Environment

The `RunnerEnv` simulator couples a discrete workout plan with metre-accurate elevation data:

Athlete profiles JSON templates store resting/max HR, FTP, mass, and fitness factor for `elite`, `runner`, and `amateur`.

Training plans Each plan is defined by ordered segments (duration, target zones, repeat count) which are expanded to a per-second schedule.

Track data GPX files are parsed with `track.py`; slope is discretised via a $\pm 0.5\%$ threshold to curb noise.

State transition Heart-rate drifts toward the power-implied target with a first-order lag; power changes instantaneously on action. Segment advancement and slope lookup are strictly time-indexed.

Reward computation Besides zone compliance and fatigue, the reward adds (i) physiological coherence between HR and power, (ii) slope-aware pacing penalties, (iii) a dynamic "funnel" bonus that tightens tolerance as the workout progresses.

4 Communication

During live sessions the agent publishes guidance through MQTT over the public broker `broker.emqx.io`. Each second a JSON payload such as


```
{
  "second": 732,
  "phase" : "push",
  "fatigue": "medium",
  "action": "accelerate"
}
```


is emitted on topic `smartpacer/action`. A lightweight subscriber (see `mqtt.py`) formats the cue with emoji and streams it to the console or any mobile UI, achieving sub-250 ms end-to-end latency on campus Wi-Fi. The same hook can be integrated in wearable applications via BLE-to-MQTT bridges.

5 Results

5.1 Training Performance

Across 24 athlete–workout combinations the cumulative reward converged within $\approx 1\,200$ episodes; elite profiles required fewer iterations owing to laxer fatigue penalties. Averaged over the last 100 episodes, the per-minute reward improved from -0.8 ± 0.4 (random policy) to $+2.1 \pm 0.3$. Reward curves (Figure **??**) show smooth monotonic growth, confirming stable hyper-parameter choices.

5.2 Behavioural Analysis

Qualitative inspection of *endurance* runs on the `acquedotti` circuit highlights:

- Early push phases where the agent accelerates only until the athlete enters Z_3 HR, then locks pace despite positive slope changes.
- Prompt slow-down commands once cumulative fatigue crosses the medium threshold, preventing long exposures to Z_4.
- Recovery segments where the agent holds *keep going* on flat sections yet recommends *slow down* on downhills, indicating slope-aware energy saving.

5.3 Online Deployment

Three volunteer runners completed 5 km tests wearing a heart-rate strap and a foot-pod power meter while receiving live cues on a smartphone. All reported the suggestions to be "intuitive" and "well-timed", with HR trace analysis showing +18% time inside the prescribed zone compared with self-pacing.

6 Identified Challenges and Future Developments

1. **Sensor Noise & Delay**: wrist-based HR sensors add 3–5 s latency; future work will incorporate a Kalman predictor to compensate.
2. **State-Space Explosion**: tabular Q-learning limits resolution. We plan to migrate to a DQN with embeddings for continuous HR and power.
3. **Generalisation to New Tracks**: current policies are learnt on three circuits; Domain-Randomised training on synthetic elevation profiles could improve robustness.
4. **Physiological Fidelity**: integrating VO_2 and glycogen models would allow carbohydrate-aware pacing and fuelling advice.
5. **User Experience**: a web dashboard with real-time charts and Strava export is under development, alongside a Flutter watch app for offline guidance.