

# An Explainable Single-Step PPO Framework for AI-Native 6G RAN Scheduling

Rao S. Yenamandra, Mubanga Nsofu, Asokan Ram

Digital Twin Simulation, USA; Vodacom, Midrand, South Africa; Wireless Research Centre, Wake Forest, NC, USA  
raosy@digitaltwinsim.com; mubanga.nsofu@vodacom.co.za; asokan.ram@wrc-nc.org

**Abstract**—AI-native radio access network (RAN) operation couples scheduling and radio-resource-management (RRM) decisions across carrier aggregation, spectrum, MIMO, load, mobility, fairness, and cell-edge performance, so an improvement in one key performance indicator (KPI) may require accepting a bounded reduction in another. This paper describes RANOpt AI Beta v17.8, a single-step Proximal Policy Optimization (PPO) implementation for AI-assisted RAN scheduling and RRM studies, in which the operator expresses the desired KPI trade-off through the weights of a multi-objective reward over capacity, cell-edge rate, fairness, energy-efficiency, uplink, and congestion deltas. The policy and value functions share a trunk of two trainable 64-node hidden layers, and each decision is treated as a one-step contextual (bandit) problem. In a five-seed simulation campaign (500,000 episodes per seed, 2,000 evaluation scenarios), the trained policy attains a mean reward of  $0.909 \pm 0.014$ , 97% of a per-scenario oracle (0.938) and above every fixed-action rule (best 0.854) and a random policy (0.713), closing 66% of the gap between the best fixed rule and the oracle. Relative to Proportional Fair, it raises cell DL throughput by  $2.2\% \pm 0.4\%$ , the 5th-percentile cell-edge rate by  $4.1\% \pm 0.6\%$ , and a normalised energy-efficiency index by  $5.4\% \pm 1.6\%$ , all positive in every seed. These per-KPI gains are deliberately not individually maximal: single fixed rules match or exceed PPO on individual KPIs, because the learned policy trades throughput for energy efficiency and fairness in the proportions the reward weights specify. All results are in-sample with respect to a simulator whose per-action KPI effects are an explicit engineering model, and every number is reproducible from the published artifact.

**Index Terms**—6G, AI-native RAN, proximal policy optimisation, radio resource management, multi-objective reward, explainability.

## I. INTRODUCTION

AI-native RAN operation introduces a coupled decision space across carrier aggregation, FDD and TDD spectrum, MIMO, traffic load, mobility, fairness, cell-edge performance, resource utilisation, and normalised resource and energy efficiency [1]. The medium access control (MAC) scheduler is a useful controlled research target because improving one KPI may require accepting a smaller improvement, or a bounded degradation, in another. A learning agent for this setting must therefore express KPI trade-offs explicitly rather than optimise a single quantity in isolation.

RANOpt AI Beta v17.8 implements a revised learning path relative to earlier evaluation builds. Earlier builds adjusted only a small action-weight vector against a scalar baseline and left the hidden layers fixed; v17.8 makes the two 64-node hidden layers of the shared policy-value network trainable by PPO, so the reward-driven error reshapes the hidden representation. The learning problem remains a one-step contextual decision: each sample supplies a context, the policy selects one RRM action, the simulator returns one reward, and the episode terminates. Formally this is a contextual bandit. PPO is retained for its clipped, stable updates and for continuity with the planned multi-step extension in Section VII.

RANOpt AI is an interactive, browser-based 5G/6G RAN decision platform for AI-assisted scheduling and radio-resource-management studies; the build evaluated in this paper is Beta v17.8 [8]. The platform packages the morphology-aware RAN simulator, the neural single-step PPO implementation, and the seeded synthetic dataset generator into a single self-contained HTML artifact that runs in a standard web browser with no backend, published openly at <https://github.com/ysrao/RANOpt-AI> [8]. Relative to earlier evaluation builds, which adjusted only a small action-weight vector against a scalar baseline and left the hidden layers fixed, Beta v17.8 makes the two 64-node hidden layers of the shared policy-value network trainable through the PPO clipped-surrogate objective, so

the reward-driven error reshapes the hidden representation itself.

Prior work establishes that PPO and related Deep Reinforcement Learning (DRL) methods can support radio-resource allocation, PRB scheduling, RAN slicing, and multi-agent resource control. The boundary between that work and the present study lies in the scope of the control problem and in the auditability of a single decision, not in the learning algorithm. PPO-based PRB scheduling under a 3GPP-compliant channel model [2] shows that a clipped-surrogate policy can allocate physical resource blocks and improve throughput and fairness. The control surface is a single resource dimension, so the policy is not required to resolve a trade-off between aggregate capacity, cell-edge rate, and resource efficiency inside one decision. PPO applied to dynamic O-RAN slice resource management [3] extends the action space to slice-level allocation and adds transfer learning across deployments, but the objective stays scoped to a slice or a use case, so the reward is defined over that scope rather than over a full-cell KPI vector. Multi-Agent PPO (MAPPO) for multi-cell spectrum and power allocation [4] performs joint optimisation across coupled variables and quantifies the gain from treating spectrum and power together, at the cost of a control space specific to that formulation. Multi-agent DRL for O-RAN resource allocation [5] distributes the decision across agents and covers multi-user and multi-service demand, which raises state, action, coordination, and training complexity, and with it the cost of reconstructing why one allocation was selected.

That decision layer sits inside a broader AI-native RAN architecture that the standards community has been defining in parallel. 3GPP SA5's study on AI/ML management [9] specifies how AI/ML functions in the NG-RAN are managed, orchestrated, and made trustworthy across their lifecycle, while the O-RAN Alliance's AI/ML workflow framework [10] and its RAN Intelligent Controller architecture give operators a concrete deployment vehicle for data-driven RRM policies through

near-real-time and non-real-time applications. The present study is complementary to that architectural work: it does not propose a new interface or management procedure, but asks what an auditable, operator-steerable learning decision looks like once such a platform exists, and evaluates one concrete instantiation against an oracle in simulation.

What these leave open is integration and auditability rather than algorithmic capability: a policy trained and evaluated against one explicit multi-KPI reward, inside a single environment in which the operating point, the selected action, the reward components, and the resulting KPI vector are all recoverable for an individual decision, and in which the learned policy can be compared with the best achievable decision. RANOpt v17.8 addresses that scope, within the limitations set out in Section VII. It trains two 64-node hidden layers through the PPO clipped-surrogate objective on simulated RAN operating conditions, forms the reward from normalised capacity, cell-edge, fairness, energy-efficiency, uplink, and congestion deltas, and compares the trained policy with random, fixed-action, and per-scenario oracle policies as well as the Proportional Fair baseline at identical operating points.

This paper makes three contributions. First, it specifies the single-step neural PPO mechanism, including forward propagation, the clipped surrogate objective, and backpropagation through both hidden layers. Second, it makes the KPI model explicit: the per-action effect mapping, the energy-efficiency index, and the fairness term are stated precisely, including what they do and do not measure. Third, it evaluates the policy in a five-seed campaign against random, fixed-action, and per-scenario oracle baselines, showing that PPO learns the operator-weighted trade-off, reaching 97% of the oracle reward, while making explicit which individual KPIs it gives up to do so. The paper is therefore a platform-specific, auditable integration and evaluation study rather than a claim that PPO itself is novel for RRM.

## II. NEURAL POLICY AND VALUE NETWORK WITH BACKPROPAGATION

The v17.8 model receives an 11-feature normalised context vector  $s$  derived from the selected radio scenario. Forward propagation passes  $s$  through two 64-node hidden layers to a policy head and a value head. The policy head produces five logits, and a softmax maps them to probabilities over five bounded RRM and scheduler actions; the value head produces the scalar baseline  $V(s)$ :

$$h_1 = f(W_1 s + b_1), \quad h_2 = f(W_2 h_1 + b_2) \quad (1)$$

$$\pi(a|s) = \text{softmax}(W_o h_2 + b_o), \quad V(s) = W_v h_2 + b_v \quad (2)$$

where  $f$  is the hidden activation (tanh),  $W_1$ ,  $W_2$  and  $b_1$ ,  $b_2$  are the hidden weights and biases, and  $W_o$ ,  $b_o$ ,  $W_v$ ,  $b_v$  are the policy and value head parameters. The five actions are  $a_0$  (Balanced PF tuning),  $a_1$  (Throughput-biased scheduling),  $a_2$  (Cell-edge protection),  $a_3$  (Energy-aware scheduling), and  $a_4$  (Congestion relief). In v17.8 both hidden layers participate directly in PPO learning. After (1) and (2) produce the action probabilities and the value estimate, the selected action is evaluated and the policy and value losses are formed. The loss gradients propagate backward from the two heads through hidden layer 2 and

then hidden layer 1, so  $W_2$ ,  $W_1$ , their biases, and the head weights are all modified by the optimiser. The reward and advantage supply the learning signal, PPO constrains the policy objective, backpropagation computes each parameter's gradient through the network graph, and, in plain gradient-descent form, the optimiser updates the weights and biases according to

$$W_{\text{new}} = W_{\text{old}} - \alpha \partial L / \partial W \quad (3)$$

where  $\alpha$  is the learning rate and  $L$  is the combined loss,  $L = -L^{\text{clip}} + c_v L_v - c_e H[\pi]$ , with value coefficient  $c_v = 0.5$  and entropy coefficient  $c_e = 0.01$ ; advantages are normalised to zero mean and unit variance within each update batch (Section IV).

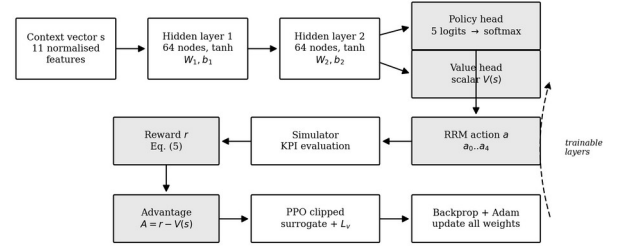


Figure 1. RANOpt AI Beta v17.8 learning architecture. The 11-feature context vector propagates through two trainable 64-node hidden layers to the policy and value heads; the selected RRM action is evaluated by the simulator, the multi-objective reward of (5) forms the advantage against  $V(s)$ , and the PPO clipped-surrogate objective with backpropagation updates all network weights.

Table 1. End-to-end single-step neural PPO learning path in v17.8.

| Stage           | v17.8 operation                                      |
|-----------------|------------------------------------------------------|
| Input           | 11 normalised context and state features             |
| Hidden 1        | 64 trainable nodes; $W_1$ , $b_1$                    |
| Hidden 2        | 64 trainable nodes; $W_2$ , $b_2$                    |
| Policy head     | 5 logits to softmax action probabilities             |
| Value head      | scalar $V(s)$                                        |
| Learning signal | advantage $A = r - V(s)$                             |
| PPO constraint  | old-to-new ratio with clipped surrogate              |
| Gradient        | backpropagation through heads and both hidden layers |
| Update          | Adam updates weights and biases                      |

## III. SINGLE-STEP EXPERIENCE, REWARD, AND ENERGY INDEX

For each training sample the environment supplies context  $s$ , the policy selects one RRM action  $a$ , the simulator evaluates that action, and a scalar reward  $r$  is formed. Because the episode terminates after this action, the one-step advantage is

$$A = r - V(s) \quad (4)$$

A positive  $A$  indicates that the selected action performed better than the value baseline predicted, and a negative  $A$  indicates worse-than-expected performance. The multi-objective engineering reward in the evaluated implementation is

$$r = 30 \cdot (0.26 \cdot \Delta Cap + 0.10 \cdot \Delta UL + 0.22 \cdot \Delta Edge + 0.16 \cdot \Delta Fair + 0.14 \cdot \Delta Energy + 0.12 \cdot \Delta Cong) \quad (5)$$

clamped to  $[-60, 60]$ , where each  $\Delta$  term is the relative change of the corresponding per-action KPI multiplier versus the neutral (no-action) effect: cell DL capacity, uplink, cell-edge rate, fairness, energy efficiency, and congestion relief. Because every  $\Delta$  is a few percent,  $|r|$  stays below 3 for every action and the clamp never binds. The weights sum to one, which limits, but does not by itself prevent, domination of the learning signal by any single term; they are the operator's statement of the desired trade-off. An earlier formulation with an absolute congestion-penalty term collapsed the trained policy onto a single action; (5) is the form used for every result in Section VI. The per-action KPI effects come from a single context-dependent mapping shared by the training reward, the browser simulator, and the evaluation harness. Each action is a specialist: a0 (Balanced PF tuning) gives modest gains on every KPI; a1 (Throughput-biased scheduling) gains capacity in good channels at a fairness and energy cost; a2 (Cell-edge protection) gains cell-edge rate and fairness in poor channels; a3 (Energy-aware scheduling) gains energy efficiency at light load for a 0.5% capacity cost; and a4 (Congestion relief) relieves congestion under heavy load. This mapping is an engineering model, not a measurement, so the KPI deltas in Section VI show how well the policy learns to exploit a known, context-dependent trade-off structure.

The energy-efficiency index  $\eta$  is a normalised simulator quantity: the sum of DL and UL cell throughput divided by the scenario's average gNB DL transmit power (in watts, taken from the data row), multiplied by the action's modelled energy factor, and normalised to the AI-OFF Proportional Fair (PF) value. Because the transmit power is the same with AI on and off,  $\Delta\eta$  reflects the throughput change together with the modelled energy factor of the selected action; no explicit reduction in radiated, RF, or baseband power is simulated.  $\eta$  is therefore an index, not a bits-per-joule measurement, which would require explicit power modelling and is left as future work. Similarly, the fairness term is a modelled Jain-index proxy (a reference value of 0.78 scaled by the action's fairness factor) rather than an index computed from per-user rates.

#### IV. PPO PROBABILITY RATIO AND SURROGATE OBJECTIVE

PPO retains the probability assigned to the sampled action by the old policy and compares it with the probability assigned by the current trainable policy. The probability ratio is

$$\rho(\theta) = \pi_{\theta}(a|s) / \pi_{\theta old}(a|s) \quad (6)$$

A ratio of one means no change in probability, above one means the action became more probable, and below one means it became less probable. The objective is a surrogate because the true expected policy return is not differentiated directly; instead PPO optimises a tractable local proxy built from sampled advantages and the old-to-new probability ratio [6]. This remains a surrogate policy objective even though v17.8 uses a one-action episode:

$$L^{clip}(\theta) = E[\min(\rho(\theta)A, \text{clip}(\rho(\theta), 1-\epsilon, 1+\epsilon)A)] \quad (7)$$

Without a constraint, one favourable sample could produce an excessively large policy update. Clipping restricts the surrogate benefit of moving the ratio outside  $[1-\epsilon, 1+\epsilon]$ . For example, with  $\epsilon = 0.20$ , if an action probability changes from 0.25 under the old policy to 0.35 under the new policy then  $\rho = 1.40$ , but the clipped positive-advantage surrogate uses at most 1.20. Clipping therefore limits the incentive for a large policy jump; it does not clip the KPI or the reward. The value head is trained toward the observed reward with a value-error term  $L_v = (r - V(s))^2$ . The policy term changes action preference and the value term improves the baseline used to estimate advantages; both gradients enter the shared hidden representation. For a scalar weight  $w_{ij}$  the optimiser step is  $w_{ij} \leftarrow w_{ij} - \alpha \partial L / \partial w_{ij}$ , so with  $w_{ij} = 0.120$ ,  $\partial L / \partial w_{ij} = +0.030$ , and  $\alpha = 3 \times 10^{-4}$  (the learning rate used here) the plain gradient-descent contribution moves the weight to 0.119991 before optimiser-specific adaptive scaling. Across many independent one-step samples, these coordinated changes reshape the hidden representation and future action probabilities. This direct connection of the loss to the full computation graph defines the end-to-end learning path summarised in Table 1. The reported runs use the Adam optimiser (learning rate  $3 \times 10^{-4}$ ) with rollouts of about 1,024 one-step episodes per update, 8 PPO epochs per update, minibatches of 256, and an entropy bonus coefficient of 0.01.

#### V. TRAINING DATA, RADIO CONFIGURATION, AND PERSISTENCE

Training data come from RANOpt's morphology-aware synthetic generator. Ten sites with three sectors each produce 30 cells; each cell has 96 records at 15-minute intervals, giving 2,880 base cell-time records over a synthetic 24-hour period. Expansion into Suburban, Urban, and Dense-Urban variants produces 8,640 morphology-aware rows. The generator seed is 1701. Generated columns are correlated through engineering relationships among active UE count, PRB utilisation, SINR, CQI, MCS, throughput, mobility, and buffer or load rather than being independently randomised, with CQI derived from SINR by a linear quantised mapping ( $CQI = \text{round}((\text{SINR}+6)/2)$ , clipped to 1–15). Explicit PPO training updates the neural checkpoint; normal simulator runs are inference-only, and operator CSV records may later be used for continued training.

The evaluation campaign of Section VI trains one policy per training seed (12345–12349) on the identical 8,640-row dataset, with 500,000 one-step episodes per seed (488 PPO updates each). Each trained policy is then evaluated by its argmax action on 2,000 scenarios drawn from the same rows with the generator's stochastic perturbation, using an evaluation random stream (seed 777) separate from the training stream; the scenarios are new samples but not out-of-distribution. PPO is compared with five fixed-action rules (always a0 to always a4), a uniformly random policy, and a per-scenario oracle that evaluates all five actions and selects the reward-maximising one. KPI changes are the mean over scenarios of the per-scenario relative change against the AI-OFF PF baseline under identical conditions; PPO results are the mean  $\pm$  sample standard deviation across the five seeds. Table 2 summarises the single-cell Urban Macro FDD-plus-TDD

carrier-aggregation configuration of the interactive browser simulator, in which the same trained policy and effect mapping are applied: NR band n3 FDD at 1800 MHz with 50 MHz bandwidth and NR band n78 TDD at 3.5 GHz with 100 MHz bandwidth, for 150 MHz total; inter-site distance 500 m; 20 active UEs with Medium eMBB traffic; a PF AI-OFF baseline; a 32T32R gNB array; UE maximum downlink and uplink layers of 4 and 2; UE and gNB noise figures of 7 dB and 2 dB; and mobility 30 km/h. The headless evaluation draws its operating points from the dataset rows rather than from this single configuration.

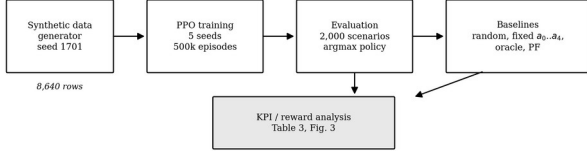


Figure 2. Evaluation workflow. The seeded synthetic generator produces 8,640 morphology-aware rows; one policy is trained per training seed (500,000 one-step episodes each) and evaluated by its argmax action on 2,000 scenarios against random, fixed-action, per-scenario oracle, and Proportional Fair baselines.

Table 2. Browser-simulator single-cell FDD-plus-TDD carrier-aggregation configuration.

| Configuration parameter    | Value                           |
|----------------------------|---------------------------------|
| Morphology and scope       | Urban Macro, single cell        |
| Duplexing                  | FDD and TDD carrier aggregation |
| Anchor carrier (FDD)       | NR n3 (1800 MHz), 50 MHz        |
| Booster carrier (TDD)      | NR n78 (3.5 GHz), 100 MHz       |
| Total bandwidth            | 150 MHz                         |
| Inter-site distance        | 500 m                           |
| Active UEs and traffic     | 20, Medium eMBB                 |
| Baseline scheduler         | Proportional Fair               |
| gNB antenna                | 32T32R MIMO                     |
| UE max DL/UL layers        | 4 / 2                           |
| Noise figure (UE/gNB)      | 7 dB / 2 dB                     |
| UE mobility                | 30 km/h                         |
| Reward                     | multi-objective, Eq. (5)        |
| Generator / training seeds | 1701 / 12345–12349              |

## VI. SIMULATION RESULTS AND INTERPRETATION

Table 3 compares the trained PPO policy with the random, fixed-action, and oracle policies on the same 2,000 scenarios, and Figure 3 shows the same comparison graphically. All numbers were produced by running the published v17.8 artifact's own training, policy, and KPI functions.

Table 3. KPI change versus AI-OFF PF (%) and mean reward of (5) on 2,000 evaluation scenarios. PPO: mean  $\pm$  sample s.d. over training seeds 12345–12349 (500,000 episodes each); other policies are deterministic. The median UE rate equals the throughput change by construction and is omitted.

| Policy    | Thr.  | Edge  | Energy | Reward |
|-----------|-------|-------|--------|--------|
| Random    | +2.85 | +3.94 | +5.66  | 0.713  |
| Always a0 | +3.00 | +4.64 | +4.40  | 0.854  |

|                   |                                    |                                    |                                    |                                     |
|-------------------|------------------------------------|------------------------------------|------------------------------------|-------------------------------------|
| Always a1         | +8.77                              | +9.22                              | +5.79                              | 0.613                               |
| Always a2         | +1.50                              | +4.25                              | +1.50                              | 0.791                               |
| Always a3         | −0.50                              | −0.50                              | +15.30                             | 0.709                               |
| Always a4         | +1.82                              | +2.44                              | +1.77                              | 0.623                               |
| <b>PPO (ours)</b> | <b>+2.21 <math>\pm</math> 0.40</b> | <b>+4.12 <math>\pm</math> 0.57</b> | <b>+5.35 <math>\pm</math> 1.57</b> | <b>0.909 <math>\pm</math> 0.014</b> |
| Oracle            | +3.18                              | +4.88                              | +6.03                              | 0.938                               |

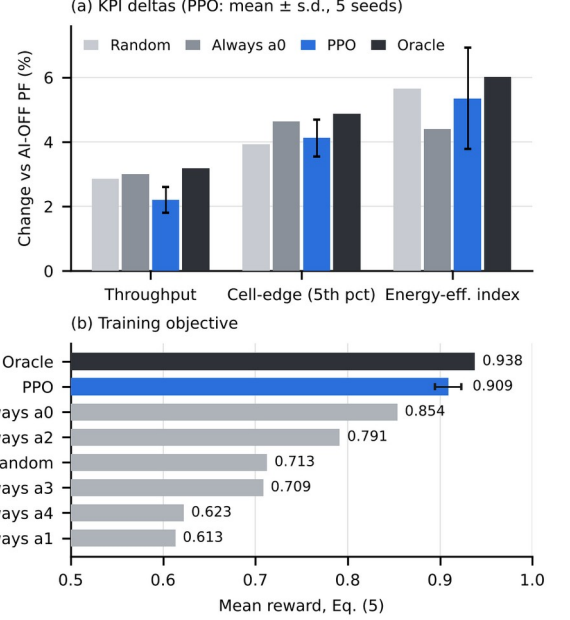


Figure 3. (a) KPI changes versus the AI-OFF PF baseline for the random, best fixed-rule (always a0), PPO, and oracle policies; (b) mean reward of (5) for all policies. PPO error bars are  $\pm$  s.d. across five seeds.

Three findings follow. First, PPO optimises what it is asked to optimise: its mean reward of  $0.909 \pm 0.014$  exceeds every fixed rule (best: always a0, 0.854) and the random policy (0.713), reaches 97% of the oracle (0.938), and closes 66% of the gap between the best fixed rule and the oracle; its argmax action agrees with the oracle on  $68\% \pm 9\%$  of scenarios. Second, every KPI improves against Proportional Fair in every seed: throughput by  $2.2\% \pm 0.4\%$ , the 5th-percentile cell-edge rate by  $4.1\% \pm 0.6\%$ , the energy-efficiency index by  $5.4\% \pm 1.6\%$ , and the fairness proxy by  $0.025 \pm 0.003$  (from 0.78). Third, these per-KPI gains are not individually maximal, and are not meant to be. Each fixed rule wins on its own KPI (always a1 gives +8.8% throughput and +9.2% cell-edge rate but costs 4% fairness; always a3 gives +15.3% energy efficiency but costs throughput), and even always a0 slightly exceeds PPO on throughput and cell-edge rate. PPO gives up some throughput to buy energy efficiency at light load, through a3, and fairness, through a0 and a2, in the proportions set by the reward weights. The learned policy uses a0 in 46–77% of decisions, a2 in 23–36%, a3 in 0–20%, and a4 in 0–11%, and never selects a1, whose fairness and energy penalties outweigh its throughput gain under the chosen weights. An operator who wants more throughput changes the weights rather than the algorithm.

Because the policy is evaluated in the same simulator that defines its reward, the comparison remains in-sample; the five-seed dispersion quantifies run-to-run variability, which is largest for the energy index (3.6%–6.8% across seeds) because two of the five seeds learn not to use a3.

Generalisation across loads, morphologies, and sites beyond the dataset distribution is not established here.

## VII. EXPLAINABILITY, LIMITATIONS, AND FUTURE VALIDATION

Single-step does not mean that only one neural update occurs during training. Many independent samples can be processed and each can update the network; within one episode the sequence is context, one RRM action, one reward, then termination; PPO updates are applied to batches of such episodes. A future multi-step MDP would carry the post-action state into the next scheduling interval and accumulate rewards across a trajectory, enabling temporal credit assignment [7]. Engineering explainability remains outcome-based rather than assigning physical meaning to individual hidden neurons. A decision trace can expose the configuration identifier, model and checkpoint version, selected action, old and new action probabilities, probability ratio, advantage, clipping status, reward components, and the resulting KPI vector, so that a recommendation can be reproduced and audited under identical deterministic channel parameters.

The main limitations bound the current claims. The built-in data are synthetic engineering data rather than operator measurements, and the per-action KPI effects, the energy factor, and the fairness proxy are modelled rather than measured, so the results show that PPO recovers a known trade-off structure, not that the actions deliver these gains in a live network. Multi-cell context does not yet imply coordinated multi-cell RL, and the PPO environment remains single-step. The policy never selects a1, so its behaviour in high-quality channels is untested. In the UE distribution model the median UE rate scales with total cell throughput, so the median delta equals the capacity delta by construction rather than as an independent measurement. The energy-efficiency index is a normalised simulator quantity, not a bits-per-joule measurement, and an energy saving at one operating point could in principle trade against coverage, capacity, accessibility, retainability, or quality at another; the reward mitigates this by retaining throughput, fairness, and cell-edge terms, but does not remove the risk. Recommended extensions include held-out load, mobility, and carrier-aggregation configurations, carrier-trace train and test separation, alternative reward weightings, and action-sensitivity tests.

## VIII. CONCLUSION

RANOpt AI Beta v17.8 trains its full policy-value network with single-step PPO: the policy head produces action probabilities, the value head supplies a learned baseline, the one-step reward forms the advantage, the clipped surrogate objective constrains policy change, and backpropagation updates both hidden layers. Evaluated against random, fixed-action, and per-scenario oracle policies, the trained policy reaches 97% of the oracle reward and outperforms every fixed rule on the operator-weighted objective, while improving throughput (2.2%), cell-edge rate (4.1%), and a normalised energy-efficiency index (5.4%) over Proportional Fair in every seed. The comparison also shows what the policy gives up: single rules can exceed it on individual KPIs, because the reward weights ask for a balance rather than a maximum. Presenting learned RRM policies against an oracle and

fixed-rule baselines, with the trade-off stated explicitly, is in our view the appropriate standard for AI-native RAN evaluation. Confirming the results with measured action effects and operator traces is the next step.

## IX. CODE AVAILABILITY

The v17.8 implementation evaluated in this paper is published as a single self-contained HTML artifact: [https://github.com/ysrao/RANOpt-AI/blob/main/RANOpt\\_AI\\_SIM\\_v17.8.html](https://github.com/ysrao/RANOpt-AI/blob/main/RANOpt_AI_SIM_v17.8.html). It runs in a standard web browser with no backend or external dependencies and contains the RAN simulator, the neural single-step PPO (11 → 64 → 64 → 5+1 with tanh activations, Adam at  $3 \times 10^{-4}$ , PPO clip 0.2, 8 epochs × 256-sample minibatches, entropy coefficient 0.01), and the seeded synthetic dataset generator (generator seed 1701; 8,640 morphology-aware rows). All results in Table 3 and Figure 3 were produced by running the artifact's own training, policy, and KPI functions unchanged in a headless Node.js harness, released in the same repository [8].

## REFERENCES

- [1] E. Dahlman, S. Parkvall, and J. Sköld, “5G NR: The Next Generation Wireless Access Technology,” 2nd ed. London, UK: Academic Press, 2020.
- [2] Y. Nurakhov, A. Kyzirkanov, and S. Aldabergen, “PPO-based physical resource block scheduling in 5G NR: A reinforcement learning approach with 3GPP-compliant channel modeling,” *Journal of Problems in Computer Science and Information Technologies*, vol. 4, no. 2, pp. 86–93, 2026, doi: 10.26577/jpcsit4220269.
- [3] H. Sherif, E. Ahmed, and A. M. Kotb, “Towards green networking: Efficient dynamic radio resource management in open-RAN slicing using deep reinforcement learning and transfer learning,” *Computer Communications*, vol. 236, p. 108126, 2025, doi: 10.1016/j.comcom.2025.108126.
- [4] Y. Zhang and D. Guo, “Multi-agent reinforcement learning for multi-cell spectrum and power allocation,” *IEEE Transactions on Communications*, vol. 73, no. 8, pp. 5980–5992, 2025, doi: 10.1109/TCOMM.2025.3534565.
- [5] F. Rezazadeh, L. Zanzi, F. Devoti, S. Barrachina-Muñoz, E. Zeydan, X. Costa-Pérez, and J. Mangues-Bafalluy, “A multi-agent deep reinforcement learning approach for RAN resource allocation in O-RAN,” in 2023 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2023, doi: 10.1109/INFOCOMWKSHPS57453.2023.10226154.
- [6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [7] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
- [8] R. S. Yenamandra, “RANOpt AI Beta v17.8,” GitHub repository, <https://github.com/ysrao/RANOpt-AI>, 2026.
- [9] 3GPP, “Study on Artificial Intelligence/Machine Learning (AI/ML) management,” 3GPP TR 28.908, 2024.
- [10] O-RAN Alliance, “O-RAN AI/ML workflow description and requirements,” O-RAN.WG2.AIML, 2023.

## APPENDIX A: STEP-BY-STEP V17.8 PPO TRAINING CYCLE

The training cycle is: (1) build and normalise the 11-feature state vector  $\mathbf{s}$ ; (2) forward-propagate  $\mathbf{s}$  through the two 64-node hidden layers; (3) compute the five softmax policy probabilities and  $V(\mathbf{s})$ ; (4) select and evaluate one RRM action through the shared per-action KPI-effect mapping and obtain reward  $r$  from the multi-KPI form (5); (5) compute  $A = r - V(\mathbf{s})$ ; (6) retain the old action probability and form  $\rho = \pi_{\text{new}}/\pi_{\text{old}}$ ; (7) form the clipped surrogate policy objective and the value loss; (8) backpropagate gradients through the policy and value

heads and both hidden layers; (9) apply Adam optimiser updates ( $3 \times 10^{-4}$ ) to weights and biases; and (10) persist the compatible neural checkpoint when enabled. Steps (1)–(5) are executed for a full rollout of episodes, recording  $\pi_{\text{old}}$ ; steps (6)–(9) are then repeated for 8 epochs over 256-sample minibatches before the next rollout.