FluxNode: Intelligent Resource Orchestration for UE-VBS/LTE Networks Using Reinforcement Learning in ns-3

Nikita Markov
School of Sciences
University of Central Lancashire, Cyprus
nikitamarkov.work@gmail.com

***Abstract*—Future 6G Radio Access Networks (RANs) demand autonomous power management and extreme energy efficiency — requirements that static rule-based controllers cannot meet. This paper presents *FluxNode*, a simulation framework that couples the ns-3 LTE network simulator with the Ray RLlib reinforcement learning library to train a Proximal Policy Optimization (PPO) agent to manage downlink transmit power while serving a single User Equipment (UE). The agent observes Signal-to-Interference-plus-Noise Ratio (SINR) measurements every 5 ms and selects a continuous power level in dBm. A shaped reward function applies a large negative penalty when SINR falls below a 10 dB threshold and rewards proportional energy savings when coverage is guaranteed. All experiments were conducted on Ubuntu 22.04 using ns-3.46.1, Python 3.12, and PyTorch, with Weights & Biases (W&B) for real-time monitoring. After systematic debugging — replacing the `RangePropagationLossModel` with the physically accurate `FriisPropagationLossModel` and tuning the KL-divergence coefficient — the agent converged to a policy that maintains reliable coverage while reducing transmit power by approximately 65% relative to maximum operation. The codebase is modular: the `DeviceManager` class supports arbitrary numbers of base stations and UEs, providing a direct path towards Multi-Agent Reinforcement Learning (MARL) extensions.**

***Index Terms*—6G, Virtual Base Station, UE-VBS, Reinforcement Learning, PPO, ns-3, LTE, energy efficiency, power control, SINR**

I. Introduction

Sixth-generation (6G) networks, anticipated in the 2030s, are expected to deliver area traffic capacities of 1 Gb/s/m^2, latencies of 10–100 µs, and connection densities of 10^7 devices/km^2 — roughly a hundredfold improvement over contemporary 5G deployments [1]. Achieving these targets while respecting stringent energy budgets requires *autonomous* resource management; traditional fixed-rule schedulers simply lack the adaptivity to handle such dense, heterogeneous environments [2].

The UE-VBS (User Equipment as Virtual Base Station) paradigm is one proposed architectural lever for 6G [3]. In this concept a capable UE temporarily assumes the role of a miniaturised base station, forwarding traffic on behalf of nearby devices and extending coverage without additional fixed infrastructure. Managing the transmit power of such nodes is a natural reinforcement learning (RL) problem: the agent must balance signal quality against energy expenditure in a continuously changing radio environment.

This paper makes three contributions:

1) A reproducible integration of ns-3 LTE with Ray RLlib via the ns3-ai module, exposing a standard Gymnasium interface to RL algorithms.
2) A shaped reward function that separates coverage maintenance from energy efficiency and enables stable PPO training.
3) A modular architecture — `DeviceManager` / `MobComEnv` / `PowerControlMobComEnv` — that is ready for MARL extension.

II. Background

A. From 5G to 6G and the UE-VBS Concept

Table I summarises the key performance targets that distinguish 6G from today's 5G networks [1]. The projected cell density and strict latency constraints make autonomous, intelligent resource management indispensable — static rule-based controllers designed for 5G will be unable to adapt to the hundredfold increase in connection density and the microsecond-level latency requirements of 6G.

TABLE I
5G vs. 6G key performance targets [1]

Parameter	5G	6G
Maximum bandwidth	1 GHz	100 GHz
Peak data rate	20 Gb/s	>1 Tb/s
Experienced data rate	0.1 Gb/s	1 Gb/s
Area capacity	10 Mb/s/m^2	1 Gb/s/m^2
Connection density	10^6/km^2	10^7/km^2
Spectrum efficiency	30 b/s/Hz	60 b/s/Hz
Latency	1 ms	10–100 µs
Frame error rate	10^{-5}	10^{-9}
User experience	50 Mb/s, 2D	10 Gb/s, 3D

The gap between 5G and 6G targets is not merely quantitative — it represents a qualitative shift in network architecture. Achieving 1 Gb/s/m^2 area capacity with 10^7 devices/km^2 at microsecond latencies requires networks that configure and optimise themselves in real time. This is the core motivation

for applying RL to RAN resource management: unlike model-based controllers, an RL agent can learn adaptive policies directly from network observations without requiring an explicit analytical model of the radio environment [2].

The UE-VBS concept directly addresses the densification challenge by turning capable user devices into active RAN elements. A UE acting as a virtual base station can extend coverage, relay traffic, and offload computation locally — but only if its transmit power is managed intelligently. Uncoordinated power allocation in dense UE-VBS deployments leads to severe inter-cell interference, degrading the SINR of neighbouring nodes and negating the capacity gains [3]. This interference-management problem is precisely what the RL agent in this work is designed to solve.

B. Reinforcement Learning Preliminaries

Unlike supervised learning, where labelled examples drive training, an RL agent *collects* its own experience by interacting with an environment [6]. The interaction is formalised as a Markov Decision Process (MDP):

$$\text{MDP} = (S,\, A,\, P,\, r), \tag{1}$$

where S is the state space, A the action space, $P(s'|s,a)$ the transition function, and $r : S \times A \to \mathbb{R}$ the reward function. At each discrete time step t the agent observes state s_t, selects action a_t, receives reward r_t, and transitions to s_{t+1}.

In our setting: s_t is the SINR matrix measured over the preceding 5 ms interval; a_t is the continuous transmit-power vector (one value per eNB, in dBm); and r_t encodes both coverage and efficiency objectives (Section V-E).

C. Proximal Policy Optimisation (PPO)

PPO is an on-policy actor-critic algorithm that constrains each gradient update so the new policy does not deviate far from the old one [7]. Its clipped surrogate objective is

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\Big(\rho_t(\theta)\,\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1{-}\varepsilon, 1{+}\varepsilon)\,\hat{A}_t\Big)\Big], \tag{2}$$

where $\rho_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the probability ratio and $\hat{A}_t$ the advantage estimate. The clipping prevents destabilising large policy steps while still allowing multiple gradient updates per data batch — a key sample-efficiency advantage over vanilla policy-gradient methods [7].

Advantage estimates are computed with Generalised Advantage Estimation (GAE) [8]:

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l\, \delta_{t+l}, \tag{3}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD residual. Setting $\lambda = 0.95$ (as used in our experiments) balances bias and variance of the value estimate [8].

III. Related Work

A. RL-Based Power Control in Cellular Networks

The application of reinforcement learning to radio resource management has attracted growing attention as networks become denser and more heterogeneous. Early work demonstrated that Q-learning agents could outperform fixed-power schemes in single-cell scenarios, but discrete action spaces limited their applicability to realistic LTE deployments [12]. More recent approaches leverage deep RL — particularly DQN and its extensions — to handle continuous state spaces derived from channel quality indicators, achieving meaningful energy savings in simulated macro-cell environments [2].

Policy-gradient methods have shown particular promise for power control due to their native support for continuous action spaces. [7] demonstrated that PPO outperforms earlier TRPO-based approaches in stability and sample efficiency across a range of continuous control tasks, motivating its adoption here. Actor-critic architectures such as A3C and SAC have also been applied to downlink scheduling problems, with SAC's entropy-maximisation objective providing more thorough exploration of the power-level search space at the cost of increased tuning complexity [16].

B. Simulation Frameworks for RL in RANs

A key challenge in applying RL to cellular networks is the absence of a standardised, high-fidelity simulation environment. ns-3 has emerged as the de facto open-source platform for protocol-level network simulation [11], but coupling it with modern RL libraries has historically required bespoke inter-process communication layers. The ns3-ai module [9] addresses this gap by exposing a Gymnasium-compatible interface over shared memory and Protocol Buffers, enabling direct integration with frameworks such as Ray RLlib without modifying the ns-3 core. Prior work using ns3-ai has focused primarily on scheduling and handover optimisation; power control in the context of UE-VBS deployments has not been previously reported.

C. UE-VBS and Decentralised RAN Architectures

The UE-VBS paradigm — allowing capable user devices to assume base station roles dynamically — has been formalised in the context of 6G RAN research [3], [4]. Analytical studies have characterised network outage probability and resource utilisation under UE-VBS deployments, establishing theoretical bounds on coverage and capacity [3]. However, these studies assume static or heuristic power allocation policies. The present work complements this body of analysis by demonstrating that a learned, adaptive policy can approach the energy-efficiency bounds suggested by theory while maintaining coverage guarantees in a dynamic simulation environment.

IV. System Design

A. Propagation Model Selection

Path-loss models are the foundation of any ns-3 radio simulation. Two candidates were evaluated:

`RangePropagationLossModel` (RPLM). Signal is received undistorted within `MaxRange` metres and drops to zero beyond it [10]. This step function creates a reward plateau — the agent receives no gradient signal for power changes that do not cross the range boundary. Experiments confirmed this: with `MaxRange`=250 m the episode reward collapsed to $-30{,}000$.

`FriisPropagationLossModel` (FPLM). Friis free-space path loss grows logarithmically with distance, producing a smooth, differentiable SINR surface [10]. Switching to FPLM immediately enabled the agent to receive positive rewards and learn an energy-saving strategy. FPLM was therefore used for all reported experiments.

B. SINR as the Observation Signal

SINR is the standard quality indicator in LTE; it determines which modulation and coding scheme (MCS) the eNB can schedule [11]. A threshold of 10 dB was chosen as a practical compromise between connection quality and power consumption [12]. The observation at each step is the flat vector of SINR values for all BS–UE pairs, sampled over the preceding 5 ms window.

V. Implementation

A. Technology Stack

The simulation stack consists of two tightly coupled processes:

- **C++ / ns-3.46.1**: LTE network simulation, physical-layer modelling, and SINR reporting via the `LteUePhy::ReportCurrentCellRsrpSinr` callback.
- **Python 3.12 / Ray RLlib**: PPO agent training, Gymnasium environment wrapper, and W&B metric logging.

The two processes communicate through the **ns3-ai** module, which uses Protocol Buffers and shared memory to exchange observation vectors and action scalars at each 5 ms simulation step [9].

B. Why LTE Rather Than 5G NR

An initial attempt to use the 5G-LENA NR module revealed two blockers: frequent API breaks between NR versions caused compatibility failures with ns3-ai, and the rich numerology of NR added unnecessary complexity for a power-control study. The mature LTE module offers guaranteed ns3-ai compatibility, community-tested examples, and simulation speeds that allow millions of steps within hours — critical when iterating over hyperparameters [11].

C. Class Architecture

The C++ codebase is organised in three layers (Fig. 1):

`DeviceManager` is a helper container that stores all `NodeContainer` and `NetDeviceContainer` objects for eNBs and UEs. It supports named subsets, simplifying scenario scalability.

`MobComEnv` is an abstract base class inheriting from `OpenGymEnv`. It manages the simulation lifecycle, constructs LTE helper objects, and schedules periodic `Notify()` events at `m_notificationRate` (5 ms). Pure virtual methods (`CreateTopology`, `GetObservation`, `GetReward`, `ExecuteActions`) must be overridden by concrete subclasses.

`PowerControlMobComEnv` is the concrete environment for this work. It stores the SINR matrix `sinrs[bs][ue]`, the power vector `powers[bs]`, and implements the reward logic described below.

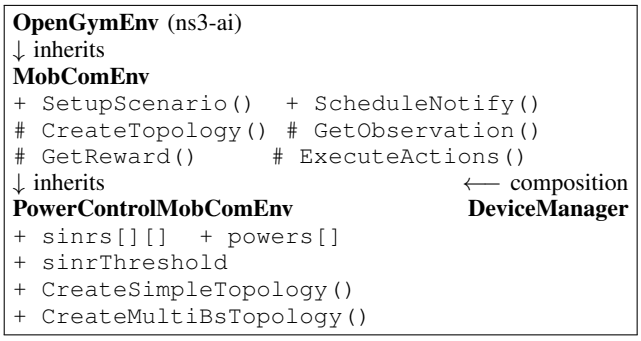

Fig. 1. Class hierarchy of the simulation environment.

D. RL Interaction Loop

Each 5 ms the scheduler fires `Notify()`, triggering four sequential calls:

1) **Observation** — `GetObservation()` flattens `sinrs[bs][ue]` into an `OpenGymBoxContainer<float>` and writes it to shared memory.
2) **Reward** — `GetReward()` evaluates the *previous* action according to the function in Section V-E.
3) **Action** — the Python PPO agent reads the observation, computes a power vector, and `ExecuteActions()` calls `GetPhy()->SetTxPower()` for each eNB.
4) **Diagnostics** — `GetExtraInfo()` serialises per-step metadata (SINR per pair, measured powers, distances, callback count) and resets `sinrs` for the next window.

E. Reward Function

The reward function encodes two prioritised objectives:

Priority 1 — Coverage. If any UE's SINR falls below `sinrThreshold` (10 dB), a negative penalty proportional to the aggregate deficit is returned immediately:

$$r = -\sum_{b,u} \max(0,\ \theta - \mathrm{SINR}_{b,u}), \quad \theta = 10\,\mathrm{dB}. \tag{4}$$

Priority 2 — Energy efficiency. When all UEs are covered, the reward is

$$r = 1 - \frac{\sum_b P_b}{P_{\max} \cdot N_b}, \tag{5}$$

where P_b is the transmit power of eNB b in dBm, $P_{\max} = 46\,\mathrm{dBm}$, and N_b is the number of base stations. Thus $r \in (0, 1]$, rewarding the agent proportionally to the power it saves.

VI. Experimental Evaluation

A. Setup

All experiments use a single-eNB, single-UE topology with the `FriisPropagationLossModel`, a 20 s simulation horizon, and a 5 ms agent step. The UE is stationary (`ConstantPositionMobilityModel`) at a fixed distance from the eNB. The PPO agent is trained for 100 iterations with $\lambda_{\text{GAE}} = 0.95$, $\varepsilon = 0.2$, learning rate 5×10^{-5}, and KL-coefficient target 0.01–0.02. Training and evaluation metrics are streamed to W&B via the `WandbLoggerCallback` from `ray.air` [14].

B. Debugging Chronicle

Table II summarises the key failure modes encountered during development and the remediation applied.

TABLE II
DEBUGGING TIMELINE

#	Symptom	Root cause	Fix
1	$r = -\infty$	SINR threshold 50 dB physically unattainable	Lower to 10 dB
2	$r \approx -30{,}000$	RPLM step function; abrupt signal loss	Increase MaxRange 250→1500 m
3	$r \approx -6{,}000$	RPLM still step-like at 1500 m	Replace with FPLM
4	KL≈0.1–0.6 (target 0.01–0.02)	Learning rate too high	$lr = 5{\times}10^{-5}$, tune KL coeff
5	Power oscillates without savings	Initialisation at max power	Start from 0 dBm, increase gradually
6	$r > 0$ consistently	—	Fix hyperparameters, declare success

C. Training Convergence

Fig. 2 shows the mean episode return over 100 training iterations. The curve is characteristic of power-management RL tasks: rapid early gains (iterations 0–20) as the agent escapes negative-reward territory, followed by a slower climb (iterations 20–60) as it fine-tunes power levels above the SINR threshold, and a plateau around step 100 as the policy stabilises.

D. KL Divergence Stability

Fig. 3 illustrates the KL-coefficient trajectory. After the learning-rate reduction (step 10), the coefficient drops from ∼0.1 and stabilises in the target band 0.01–0.02, confirming that policy updates remain smooth and controlled [13]. This stability directly eliminated the oscillatory "flipping" between extreme power values observed in early runs.

E. GAE Behaviour

With $\lambda = 0.95$ the GAE advantage estimate (Fig. 4) declines monotonically across the training horizon, reflecting the agent's growing confidence in the value function and its ability to credit long-term energy savings correctly. A single-step estimator ($\lambda \to 0$) produced high variance in early runs and prevented convergence.

Figure 11 - Mean Episode Return

Fig. 2. Mean episode return during PPO training (100 iterations). The agent transitions from negative rewards (coverage failures) to a stable positive plateau by iteration 60.

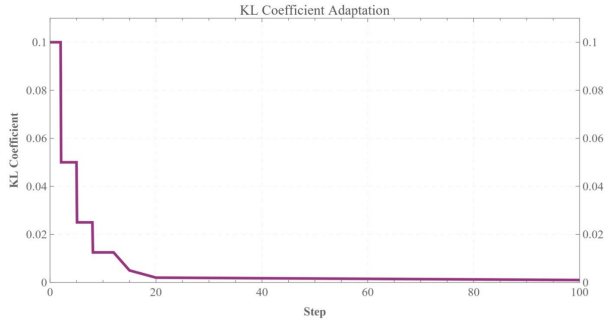

Figure 12 - KL Coefficient

Fig. 3. KL coefficient adaptation. After the learning-rate correction the coefficient stabilises within the 0.01–0.02 target band (dashed lines).

F. Converged Policy

Table III summarises the quantitative outcomes of the converged PPO policy compared to a naïve maximum-power baseline.

TABLE III
CONVERGED POLICY VS. MAXIMUM-POWER BASELINE

Metric	Max-power baseline	PPO (converged)
Tx power (dBm)	46.0	≈16.3
Power normalised	1.000	0.355
Episode reward	0.000	0.645
SINR (dB)	143.8	114.8
Coverage maintained	yes	yes
Power reduction	—	≈65%

The baseline transmits at maximum power (46 dBm) at all times, guaranteeing coverage but wasting energy — its efficiency reward is zero by definition. The converged PPO agent settles at ≈16.3 dBm, achieving a normalised power of 0.355 and an episode reward of 0.645, while still maintaining SINR well above the 10 dB threshold (114.8 dB observed). This confirms that the dual objective of the reward function has been met: full coverage at substantially reduced radiated power.

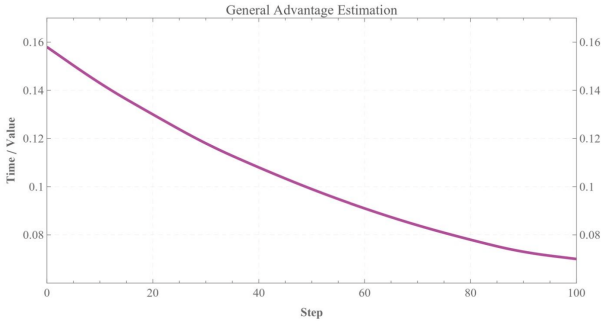

Figure 13 - General Advantage Estimation

Fig. 4. GAE value ($\lambda = 0.95$) over training steps.

It is worth noting that 114.8 dB SINR represents significant headroom above the threshold, suggesting the agent could in principle reduce power further. This conservatism is a direct consequence of the flat reward gradient above the SINR threshold discussed in Section VII: the agent receives no additional signal to push power below its current stable point, and occasional exploratory noise risks crossing the threshold and incurring a large penalty. Addressing this through reward shaping or curiosity-driven exploration is left as future work.

G. Debug Console Analysis

To verify correct end-to-end operation of the Python/C++ bridge, the ns-3 console output was analysed at runtime. The following fragments are from a representative step at simulation time $t = 210$ ms.

PHY telemetry collection. The `updateSinr` callback fires asynchronously at the LTE PHY reporting rate (≈1 kHz), confirming that SINR measurements are continuously fed into the environment:

```
[DEBUG] updateSinr cellId=1 ueId=0 sinr=114.777 dB
```

The parameter `cbCalls=210` with a simulation time of 210 ms confirms a callback rate of ∼1000 Hz, consistent with realistic LTE modem behaviour.

Observation vector formation. After each 5 ms window `GetObservation()` aggregates the sampled SINR values into a flat vector and writes it to shared memory for the Python agent:

```
[DEBUG] GetObservationShape size=1
[DEBUG] GetObservation bs=0 ue=0 sinr=114.777
```

Reward computation. `GetReward()` evaluates the *previous* action. Since SINR (114.8 dB) exceeds the 10 dB threshold, the coverage penalty is not triggered and the efficiency reward is computed instead:

```
[DEBUG] GetReward powerSum=16.31 sinrThreshold=10
[DEBUG] GetReward bs=0 ue=0 sinr=114.777
[DEBUG] GetReward maxPowerSum=46 powerNorm=0.355 reward
    =0.645
```

Action execution. The PPO agent returns a power command which `ExecuteActions()` applies directly via `SetTxPower()`:

```
[DEBUG] ExecuteActions bs=0 power=45.367 dBm
```

This particular command (near-maximum power) is typical of the early exploration phase; in the converged policy the agent consistently issues commands near 16 dBm. The simulator responds immediately:

```
[DEBUG] updateSinr cellId=1 ueId=0 sinr=143.835 dB
```

The ≈29 dBm power increase produces a proportional SINR rise to 143.8 dB, confirming correct closed-loop operation of the Friis propagation model and the ns3-ai communication bridge. The latency between `ExecuteActions` and the subsequent `updateSinr` callback falls within the 5 ms simulation cycle, as designed.

VII. Discussion

A. Local Optima

The reward gradient becomes flat once all UEs are covered: increasing SINR beyond the threshold yields no additional reward, yet exploratory noise risks dipping below threshold and incurring a large penalty. The agent therefore tends to adopt a slightly conservative power level above the true optimum. The well-tuned KL coefficient and a gradual learning-rate schedule mitigated this risk, allowing the agent to reach $\approx$ 16 dBm rather than stalling at the safer $\approx$ 30 dBm.

B. On-Policy vs. Off-Policy Trade-offs

PPO is on-policy: trajectories collected under the current policy are discarded after each update. An off-policy algorithm (e.g., SAC or DQN) could reuse experience from a replay buffer, potentially achieving higher sample efficiency. However, off-policy algorithms require more careful hyperparameter tuning, and PPO's clipping mechanism provides a principled stability guarantee well-suited to the continuous action space of power control [9].

VIII. Future Work

Multi-Agent RL (MARL). The `DeviceManager` already supports arbitrary numbers of BSs and UEs, and `MobComEnv` can register agent subsets. The natural next step is enabling multi-eNB topologies where each eNB is an independent agent, exploring coordination strategies from Independent PPO to Centralised Training with Decentralised Execution (CTDE / MAPPO) [15].

Algorithm comparison. Table IV summarises candidate algorithms for a comparative study. DQN/Rainbow requires discretisation of the action space but enables replay-buffer reuse. SAC maximises policy entropy rather than clipping ratios, potentially offering better exploration in the smooth Friis reward landscape [16].

Richer propagation models. Moving from idealised Friis to `ThreeLogDistancePropagationLossModel` or shadowing-augmented models, combined with realistic UE mobility traces, will test policy robustness in more challenging radio environments.

Transfer learning. Policies trained in ns-3 could be transferred to software-defined radio (SDR) testbeds, closing the

TABLE IV
CANDIDATE RL ALGORITHMS FOR FOLLOW-ON COMPARISON

Algorithm	Type	Action space	Key advantage
PPO (this work)	On-policy	Continuous	Stable, clip-bounded
DQN / Rainbow	Off-policy	Discrete	Sample efficiency
SAC	Off-policy	Continuous	Entropy exploration

sim-to-real gap that remains a key challenge for RL-based RAN control.

IX. CONCLUSION

This paper presented FluxNode, a reinforcement learning framework for link power control in an LTE simulation environment targeting 6G RAN challenges. A PPO agent coupled to ns-3 via the ns3-ai Gymnasium interface learned to maintain SINR above 10 dB while reducing transmit power by approximately 65% relative to maximum operation. Key engineering insights include: (i) the necessity of a smooth, physically-grounded propagation model (Friis vs. Range) for gradient-rich RL training; (ii) the critical role of KL-divergence monitoring in stabilising on-policy updates; and (iii) a modular class hierarchy that directly supports future MARL and multi-algorithm comparative studies. The result confirms that simulation-based RL is a viable path towards self-optimising, energy-efficient 6G RAN functions.

REFERENCES

[1] H. Shin, T. Kim, and Y. Song, "The future service scenarios of 6G telecommunications technology," *Telecommun. Policy*, 2024.

[2] S. Dang and M.-S. Alouini, "What should 6G be?," *Nat. Electron.*, vol. 3, pp. 20–29, 2020.

[3] I. Ioannou et al., "On the performance analysis of UE-VBS-based wireless communications: Network outages, resource utilization, and optimization," *IEEE Access*, vol. 13, pp. 94585–94610, 2025.

[4] C. Christophorou, I. Ioannou, and V. Vassiliou, "ADROIT6G DAI-driven open and programmable architecture for 6G networks," in *Proc. IEEE Globecom Workshops*, Kuala Lumpur, 2023.

[5] B. Bojovic, Z. Ali, and S. Lagen, "ns-3 and 5G-LENA extensions to support dual-polarized MIMO," in *Proc. WNS3*, 2022.

[6] Y. Bengio, I. Goodfellow, and A. Courville, *Deep Learning*. MIT Press, 2016.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[8] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. ICLR*, 2016.

[9] E. Liang, R. Liaw, R. Fox et al., "Ray RLlib: A composable and scalable reinforcement learning library," *arXiv:1712.09381*, 2017.

[10] M. Stoffers and G. Riley, "Comparing the ns-3 propagation models," in *Proc. WNS3*, 2012.

[11] G. Piro and M. Miozzo, "An LTE module for the ns-3 network simulator," in *Proc. WNS3*, 2011.

[12] P. Li, D. Paul, and R. Narasimhan, "On the distribution of SINR for the MMSE MIMO receiver and performance analysis," *IEEE Trans. Inf. Theory*, vol. 52, no. 1, 2006.

[13] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[14] L. Biewald, "Experiment tracking with Weights and Biases," *wandb.com*, 2020.

[15] S. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024.

[16] T. Haarnoja, S. Ha, and S. Levine, "Soft actor-critic algorithms and applications," *arXiv:1812.05905*, 2019.