

# Making Tunable Parameters State-Dependent in Weather and Climate Models with Reinforcement Learning

Pritthijit Nath<sup>1</sup>, Sebastian Schemm<sup>1</sup>, Henry Moss<sup>1,2</sup>, Peter Haynes<sup>1</sup>, Emily Shuckburgh<sup>3</sup> and Mark Webb<sup>4</sup>

<sup>1</sup>Department of Applied Mathematics and Theoretical Physics, University of Cambridge, UK.

<sup>2</sup>School of Mathematical Sciences, Lancaster University, UK.

<sup>3</sup>Department of Computer Science and Technology, University of Cambridge, UK.

<sup>4</sup>Met Office Hadley Centre, UK. E-mail: [pn341@cam.ac.uk](mailto:pn341@cam.ac.uk).

**Keywords:** reinforcement learning, model parametrisations, weather and climate models, machine learning, federated learning, online learning

## Abstract

Weather and climate models rely on parametrisations to represent unresolved sub-grid processes. Traditional schemes rely on fixed coefficients that are weakly constrained and tuned offline, contributing to persistent biases that limit their ability to adapt to the underlying physics. This study presents a framework that learns components of parametrisation schemes online as a function of the evolving model state using reinforcement learning (RL) and evaluates the resulting RL-driven parameter updates across a hierarchy of idealised testbeds spanning a simple climate bias correction (SCBC), a radiative–convective equilibrium (RCE), and a zonal mean energy balance model (EBM) with both single-agent and federated multi-agent settings. Across nine RL algorithms, Truncated Quantile Critics (TQC), Deep Deterministic Policy Gradient (DDPG), and Twin Delayed DDPG (TD3) achieved the highest skill and the most stable convergence across configurations, with performance assessed against a static baseline using area-weighted RMSE, temperature profile and pressure-level diagnostics. For the EBM, single-agent RL outperformed static parameter tuning with the strongest gains in tropical and mid-latitude bands, while federated RL on multi-agent setups enabled geographically specialised control and faster convergence, with a six-agent DDPG configuration using frequent aggregation yielding the lowest area-weighted RMSE across the tropics and mid-latitudes. The learnt corrections were also physically meaningful as agents modulated EBM radiative parameters to reduce meridional biases, adjusted RCE lapse rates to match vertical temperature errors, and stabilised SCBC heating increments to limit drift. Overall, results highlight RL to deliver skilful state-dependent, and regime-aware parametrisations, offering a scalable pathway for online learning within numerical models.

## Plain Language Summary

Weather and climate models cannot fully resolve small-scale processes such as cloud formation, radiation, and turbulence, so these effects are represented using simplified rules known as parametrisations. These rules are usually tweaked during model development and then held fixed during simulations, which can contribute to persistent biases when models are compared with observations. In this study, we explore whether reinforcement learning (RL), used as an online trial-and-error training method, can be used to learn machine-learning (ML) components that set existing tunable parameters as a function of the model state, while reducing biases against temporally sparse observations. We test this approach across a hierarchy of simplified climate models, ranging from bias correction to single-column convection and zonal energy balance models. Results show that RL can reduce errors relative to traditional methods while adjusting parameters in physically meaningful ways. When multiple RL agents share information, learning is faster and more responsive to local conditions. Overall, these findings suggest that RL could support the development of state-dependent parametrisation components that are trained online, then frozen for operational use, with modest computational cost compared to large contemporary ML models.

## 1. Introduction

Weather and climate models solve the governing equations of fluid dynamics, thermodynamics, continuity, and moisture transport on discretised grids, but processes occurring at scales smaller than the grid resolution cannot be explicitly represented (Kalnay 2002; Hartmann 2016). To account for these unresolved processes, such as moist convection, cloud microphysics, boundary-layer turbulence, and gravity wave drag, models rely on parametrisations that approximate their aggregate influence on the resolved flow (Randall et al. 2007; Stensrud 2007). Although indispensable for long-term simulations, parametrisations are a major source of structural error since they depend on empirical closures and limited observations (Randall et al. 2007; Vial et al. 2013; Webb et al. 2017). This reliance on simplified formulations often leads to persistent biases in climate sensitivity, precipitation, and circulation patterns (Soden and Held 2006; Flato et al. 2014). For instance, in weather forecasts, errors in parametrising mesoscale convection over the continental United States can increase ensemble spread downstream over the North Atlantic (Baumgart et al. 2018; Lojko et al. 2022), while in climate simulations, parametrisation errors can result in an uncertain response to warming. Additionally, many schemes are resolution-dependent and degrade when grid spacing approaches the scales of the represented processes, constraining their applicability in kilometre-scale models (Hallberg 2013; Stevens et al. 2019; Schneider et al. 2024). Despite decades of refinement, parametrisations remain the dominant source of uncertainty in general circulation model (GCM) projections (Ceppi et al. 2017), motivating the development of scale-adaptive machine learning (ML)-based alternatives which, although still in their early stages, may help reduce biases, improve robustness, and improve confidence in future weather and climate projections (Morcrette et al. 2025).

Historically, much tuning of numerical models has relied on manual, expert-driven adjustments targeting a small set of global or regional diagnostics such as the top-of-atmosphere (TOA) energy balance or global precipitation patterns (Hourdin et al. 2017). These manual procedures are time-consuming, subjective, and often can yield parameter choices that overcompensate for structural errors elsewhere in the model rather than reducing process-level error (Mauritsen et al. 2012). Automated tuning workflows help mitigate some of the associated cost and reproducibility issues, yet they remain constrained by the choice of summary metrics and emulator accuracy, which can bias the explored parameter subspace (Lguensat et al. 2023; Bonnet et al. 2025). Offline ML approaches, trained to map resolved states to subgrid tendencies using high-resolution simulations or observations, show promise but frequently fail to transfer reliably when coupled into a full GCM, since offline loss functions do not guarantee online stability or realistic long-term climatology (Brenowitz et al. 2020; Chen et al. 2025). This offline–online gap arises because ML parametrisations often learn spurious correlations and omit coupled-system feedbacks, allowing small errors to accumulate into large biases and producing instabilities or degraded circulation and precipitation statistics when integrated forward (Bertoli et al. 2025; Lin et al. 2025). Furthermore, both traditional tuning and many offline ML studies under-quantify uncertainty, requiring large ensembles or Bayesian calibration to assess robustness, which adds significant computational cost and complicates operational adoption (Dunbar et al. 2021; Howland et al. 2022). Finally, when both approaches are hardware-accelerated, offline-trained ML parametrisations are not always necessarily faster than well-tuned conventional parametrisations, reducing the case for ML unless it demonstrably improves predictive skill (Bertoli et al. 2025).

Reinforcement learning (RL) (Sutton and Barto 1998; Silver and Sutton 2025), one of the key driving forces behind the reasoning and alignment of large language models (LLMs) (Ouyang et al. 2022; Guo et al. 2025), gained prominence from mid-2010s when it was shown to demonstrate superhuman performance in games such as Atari (Mnih et al. 2013), Go (Silver, A Huang et al. 2016) and general gameplay (Schrittwieser et al. 2020), as well as recently in physical domains such as robot table tennis (Su et al. 2025). In the weather and climate context, RL offers a state-aware alternative to static, hand-tuned parametrisations, where policies (action selection rules) set parameters as a function of the model state at each timestep and the learning task is framed as a sequence of decisions optimised for long-horizon performance in the coupled model during training. Unlike offline surrogates trained on short-term simulations or reduced diagnostics, RL learns in a closed loop with the numerical model (without requiring it to be end-to-end differentiable), potentially allowing rewards to directly enforce physical constraints and numerical stability while balancing short-term forecast skill against long-term climatology. Continuous-control algorithms such as deep deterministic policy gradients (DDPG) (Silver, Lever et al. 2014; Lillicrap et al. 2019) and truncated quantile critics (TQC) (Kuznetsov et al. 2020) provide practical mechanisms for stable updates in high-dimensional action spaces, which map naturally onto the multivariate parameter sets of climate models. Evidence from other domains of complex physics control demonstrates that deep RL can maintain stability under strict constraints, for instance in magnetic shape control of tokamaks (Degrave et al. 2022), closed-loop separation control in turbulent

flows (Bae and Koumoutsakos 2022; Font et al. 2025), and optimisation of wind farm power generation (Mole et al. 2025). Moreover, approaches such as federated and distributed RL approaches align naturally with the domain decomposition strategies of general circulation models, where local agents can be trained on subdomains and periodically aggregated into a coherent global policy (Jin et al. 2022). Collectively, these features highlight RL as a promising pathway towards state-dependent parametrisations that adapt with evolving climate states. Complementary approaches include online ensemble Kalman inversion (EKI) (Kovachki and Stuart 2019; Christopoulos et al. 2024), which estimates parameters in-situ by iteratively updating closures to match target diagnostics, producing static calibrated parameters and showing strong results in single-column settings.

Building on the promise of RL, this study aims to design, implement, and critically evaluate RL-assisted parametrisation schemes that learn from observations while preserving the governing physical principles encoded in the host model, including conservation laws. Rather than replacing existing physics, the RL components learn state-dependent adjustments to tunable parameters in a way that remains consistent with the model's energy conservation and dynamics. Since running directly on full GCMs is computationally expensive and often difficult to interpret, this study instead focuses on experiments across a hierarchy of idealised climate testbeds allowing rapid prototyping and efficient screening of algorithms before moving to full GCM experiments in the future. The progression begins with single-agent formulations: first, a simple temperature bias correction environment is formulated where an RL agent learns state-dependent corrections to a heating rate parameter for bias-correcting the temperature value. Next, a radiative–convective equilibrium (RCE) setup is designed where the agent adjusts two parameters, evaluated in terms of long-horizon stability and temperature profile bias against reanalysis mean. Finally, building on these results, progressively more complex experiments are carried out with a multi-parameter Budyko–Sellers zonal mean energy balance model (EBM) (Budyko 1969; Sellers 1969; North 1975), together with federated coupling strategies that extend from robust single-agent continuous-control setups to multi-agent federated designs that mirror the spatial decomposition of GCMs. A central aim of this transition is to quantify how learning dynamics change under spatial decomposition and whether agents achieve improved skill through parallel learning in local regions. Together, these idealised experiments provide practical prototypes for integrating RL into operational parametrisation development, along with code for diagnostics and reproducible training recipes.

The remainder of this paper is structured as follows. Section 2 introduces the basics of RL, outlines the federated learning setup and coupling strategies used to integrate RL into climate simulations, and then describes the hierarchy of climateRL testbeds along with the training workflow used. Section 3 reports results from single- and multi-agent experiments, with discussions on stability, skill, physical meaningfulness and broader implications on future parametrisation development. Finally, Section 4 summarises the key findings, highlights avenues for future research, and points to openly available code and experiment data for transparency and reproducibility.

## 2. Methods

### 2.1. Reinforcement Learning (RL)

RL addresses sequential decision-making problems under uncertainty by allowing an agent to interact with an environment and optimise its behaviour through cumulative rewards (Silver, A Huang et al. 2016; Kiran et al. 2021; Font et al. 2025). Unlike supervised learning which depends on labelled input–output pairs, RL relies on trial-and-error interaction with the environment where policies improve as the agent receives feedback from its own actions. In contrast to other online learning strategies such as bandits, which optimise only immediate rewards without state information (Lattimore and Szepesvári 2020), RL utilises the formalism of a Markov Decision Process (MDP) to represent state information and assign credit to both immediate and delayed rewards.

#### 2.1.1. Key Components

At the core of RL, is an “agent” that interacts with an “environment” over discrete timesteps by observing a state  $s_t$ , selecting an action  $a_t$  according to its policy  $\pi(a_t | s_t)$  (which may be stochastic or deterministic), receiving a reward  $r_t$ , and transitioning to a new state  $s_{t+1}$  under some unknown dynamics  $P(s_{t+1} | s_t, a_t)$ . The fundamental ingredients are: (i) the state space  $\mathcal{S}$ , (ii) the action space  $\mathcal{A}$ , (iii) the reward function  $r(s, a)$ , (iv) the transition dynamics  $P$ , (v) the action policy  $\pi$ , and (vi) the cumulative discounted reward (known as return)  $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ , with discount factor  $\gamma \in [0, 1)$  over an episode (a finite sequence of timesteps from an initial state until a specified terminal time or stopping criterion). Together, these elements define a MDP, where the aim is to discover a policy  $\pi(a_t | s_t)$  that maximises expected return  $R_t$  while balancing exploration and exploitation (selectively preferring actions that appear to maximise  $R_t$ ). RL algorithms are commonly divided into two classes: model-free approaches, which learn directly from sampled system trajectories (e.g., Q-learning (Watkins and Dayan 1992), REINFORCE (Williams 1992)), and model-based approaches, which additionally learns a surrogate model  $P_\theta$  to support planning and improve data efficiency. In this study, the focus is on model-free methods, where RL agents interact directly with the numerical model  $P$  at each timestep by selecting an action  $a_t$  that sets the tunable parameters, after which the model integrates the state from  $s_t$  to  $s_{t+1}$ , eliminating the need for a separate surrogate dynamics model.

#### 2.1.2. TD Update

The state-value function  $V^\pi(s)$  and the action-value function  $Q^\pi(s, a)$  quantify the expected return from a state  $s$ , and from a state–action pair  $(s, a)$  respectively, when following a policy  $\pi$ . Both functions satisfy the “Bellman equations”, which express each value as the immediate reward plus the discounted estimate of the subsequent state (or state–action pair). In realistic environments, solving these equations exactly is infeasible, and approximation methods are required. Temporal-difference (TD) learning provides one such approach by updating estimates directly from already sampled transitions and “bootstrapping” from existing predictions:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)], \quad (2.1)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \quad (2.2)$$

where the bracketed terms denote the TD errors for  $V$  and  $Q$ ,  $\alpha$  represents the learning rate and  $\gamma$  denotes the discount factor as described before. TD methods underpin many value-based RL algorithms such as Q-learning. By updating incrementally at each timestep without requiring full episodic returns, TD methods provide the foundation of many scalable model-free RL algorithms.

#### 2.1.3. On-Policy vs Off-Policy

In on-policy methods, the agent improves the same policy  $\pi$  that is used while generating its training data (in our case, produced by the numerical model  $P$ ). Examples include REINFORCE and Proximal Policy Optimisation (PPO) (Schulman, Wolski et al. 2017), which require fresh trajectories for each update. Although this increases sample complexity, such approaches often can produce faster learning. Off-policy methods, by contrast, allow the learning of a target policy  $\pi_{\text{target}}$  from experience collected under a similar (but different) policy (e.g., a time-lagged version of the current policy). This distinction enables reuse of past data and greatly improves sample efficiency. Algorithms such as Deep Q-learning (Mnih et al. 2013) and Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2019) achieve stability through several mechanisms: bootstrapping, where value estimates are updated using other learnt

estimates; target networks, where slowly updated copies of networks reduce instability arising from feedback loops; and replay buffers, which break correlations in trajectories by storing and resampling transitions. These techniques underpin the stability of many modern off-policy algorithms such as Twin Delayed DDPG (TD3) (Fujimoto et al. 2018), Soft Actor–Critic (SAC) (Haarnoja et al. 2018), and Truncated Quantile Critics (TQC) (Kuznetsov et al. 2020).

#### 2.1.4. Policy Gradient Theorem

In policy-based RL, a parametrised stochastic policy  $\pi_\theta(a | s)$  is optimised to maximise the expected return under a set of state-action trajectories. Given a trajectory  $\tau = (s_0, a_0, s_1, a_1, \dots)$  generated by  $\pi_\theta$  with discount factor  $\gamma \in [0, 1]$ , the performance objective can be written as:

$$J(\theta) = \int_{\tau} \pi_\theta(\tau) R(\tau) d\tau = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \quad (2.3)$$

To compute the gradient of this objective, the **log-derivative trick** is applied and expanded across timesteps:

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \int_{\tau} \pi_\theta(\tau) R(\tau) d\tau = \int_{\tau} \nabla_{\theta} \pi_\theta(\tau) R(\tau) d\tau = \int_{\tau} \pi_\theta(\tau) \nabla_{\theta} \log \pi_\theta(\tau) R(\tau) d\tau \quad (2.4)$$

$$= \mathbb{E}_{\tau \sim \pi_\theta} [\nabla_{\theta} \log \pi_\theta(\tau) R(\tau)] = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{\infty} \nabla_{\theta} \log \pi_\theta(a_t | s_t) R_t \right], \quad (2.5)$$

where  $R_t = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k)$  is the cumulative discounted reward from time  $t$ . This formulation yields an unbiased gradient estimator and forms the basis of the REINFORCE algorithm upon which numerous modern RL algorithms are based, although in practice, variance-reduction techniques such as baselines or advantage functions are often introduced in addition to improve stability and efficiency of learning.

#### 2.1.5. Actor–Critic Methods

Actor–critic algorithms combine the strengths of value-based learning (via TD update) and policy-based learning (via policy gradients) by maintaining two sets of learnable function approximators. The actor, parametrised by  $\theta$ , defines the policy  $\pi_\theta(a | s)$ , while the critic, parametrised by  $w$ , estimates returns through a value function  $V_w(s)$  or an action-value function  $Q_w(s, a)$ . The actor is updated via gradient ascent on the policy objective:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_{\theta} \log \pi_\theta(a_t | s_t) A^\pi(s_t, a_t)],$$

where the advantage function  $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$  reduces variance by comparing action values against a state-dependent baseline.

For deterministic policies  $\mu_\theta(s)$ , as used in algorithms such as DPG, DDPG and TD3, the update, after mathematical simplifications, takes the form

$$\nabla_{\theta} J(\theta) = \mathbb{E} \left[ \nabla_a Q_w(s, a) \Big|_{a=\mu_\theta(s)} \nabla_{\theta} \mu_\theta(s) \right].$$

Meanwhile, the critic updates its parameters  $w$  by minimising the temporal-difference (TD) error,

$$\delta = r + \gamma Q_w(s_{t+1}, \pi_\theta(s_{t+1})) - Q_w(s, a), \quad \mathcal{L}(w) = \frac{1}{2} \delta^2,$$

using gradient descent. This squared-error loss  $\mathcal{L}(w)$  encourages the critic’s estimates to align with the one-step bootstrapped target  $r + \gamma Q_w(s_{t+1}, \pi_\theta(s_{t+1}))$ , ensuring consistency with the Bellman equation (Sutton and Barto 1998).

Through this interplay, the actor refines its policy based on the critic’s feedback, while the critic improves its value estimates from trajectories sampled through interactions with the environment. Collectively, this dual update mechanism yields a stable and sample-efficient framework that underpins various current continuous-control RL algorithms used in practice.

### 2.1.6. Challenges of Deep RL and Mitigation Strategies

The use of deep neural networks as function approximators (for actor and critic components) makes RL powerful but also prone to instability. Several sources of difficulty arise in practice:

- **Non-stationarity:** As the policy improves, the distribution of observed states changes, creating a moving target for policy learning. Replay buffers mitigate this by storing past transitions and sampling them as approximately independent and identically distributed (i.i.d.) batches, which helps to stabilise updates.
- **Bootstrapping Error:** Temporal-difference targets depend on their own estimates, which can amplify errors and cause divergence (known as “overestimation bias”). Stability is improved by using slowly updated target networks, where parameters are updated according to  $\theta^{\text{target}} \leftarrow \tau\theta + (1 - \tau)\theta^{\text{target}}$  with  $\tau \ll 1$ .
- **Correlated Samples:** Consecutive transitions are highly correlated, violating the i.i.d. assumption required for stochastic gradient descent (SGD). Replay buffers reduce this problem by randomising the sampling of experience.
- **High Gradient Variance:** Policy gradients can exhibit large variance, slowing convergence. Techniques such as baseline subtraction (e.g., using  $V^\pi(s)$ , as discussed in Section 2.1.5) and entropy regularisation (used in methods such as SAC) help reduce variance and promote more consistent exploration.

Together, these strategies: replay buffers, target networks, variance-reducing baselines, and entropy regularisation, form the basis of stable and scalable deep RL methods, enabling their application to high-dimensional continuous-control problems.

## 2.2. Federated Learning

### 2.2.1. FedRL Horizontal Domain Decomposition

Operational GCMs, such as the Met Office Unified Model (UM) (Brown et al. 2012), use domain decomposition in which the global domain  $\Omega$  is partitioned into subdomains  $\{\Omega_i\}_{i=1}^N$ , each mapped to a Message Passing Interface (MPI) process (known as rank)  $P_i$ . This architecture lends itself naturally to decentralised RL: a local agent  $\mathcal{A}_i$  is assigned to each rank  $P_i$ , where it observes a state  $s_i^t \in \mathbb{R}^d$  from prognostic (state variables) and selects actions  $a_i^t$  (e.g., adjusting physical parameters such as the convective entrainment rate) according to a policy distribution  $\pi_{\theta_i}(a | s_i^t)$ . In this formulation, each subdomain can develop regime-specific control strategies (policies), reflecting the fact that different parts of the domain experience distinct climates. At every timestep, the  $i$ -th agent receives a local reward,  $r_i^t = -\|\phi_i^{\text{model}}(t) - \phi_i^{\text{target}}(t)\|^2$ , where  $\phi_i$  is a diagnostic (observed or reference quantity) of interest (e.g., temperature profile or radiative flux), thus encouraging improved accuracy relative to observations or high-resolution reference simulations at the regional scale. This decentralised design ensures scalability via parallel learning while requiring only minimal modifications to the structure of the physics in the dynamical core.

To stabilise training and enforce global consistency, this decentralised setup is extended using a federated reinforcement learning (FedRL) framework (Jin et al. 2022). In this approach, each agent  $\mathcal{A}_i$  on rank  $P_i$  interacts with its local environment  $\mathcal{E}_i$ , collects trajectories, and updates its parameters  $\theta_i$  using policy gradients. After  $K$  episodes, the agents synchronise through a global averaging step,  $\theta^{\text{global}} = \frac{1}{N} \sum_{i=1}^N \theta_i$ , and the aggregated parameters are broadcast back so that  $\theta_i \leftarrow \theta^{\text{global}}$  for all  $i$ . This procedure, shortened as the FLAG cycle (short for Fine-tune Local, Aggregate Global), repeats every  $K$  episodes: agents perform local fine-tuning updates for  $K - 1$  episodes, followed by global synchronisation on the  $K^{\text{th}}$  episode. In this way, the framework balances the benefits of regional specialisation and global coherence. For climate modelling, regime-aware policies that remain globally coherent are generally preferable, since GCMs apply the same physics everywhere and must generalise under climate change (such as differing responses in warm and cool regimes). For global NWP, however, varying degrees of regional specialisation can be useful, for e.g., differentiating between tropics and high latitudes where grid-cell sizes and dominant processes differ.

This federated aggregation step accelerates convergence by allowing agents to benefit indirectly from information learnt in other regions, while still preserving data locality since each agent updates solely from its own trajectories without requiring a full-state exchange. This design closely parallels the structure of coupled GCMs, in which the dynamical core advances the state evolution while the physics package, representing sub-grid processes, interacts with it non-linearly (Gross et al. 2016). By maintaining this separation of concerns, the FLAG cycle improves sample efficiency and preserves global consistency, making the approach scalable and well-suited for HPC environments,

thus potentially enabling practical integration of RL-assisted parametrisations into production-grade numerical weather prediction (NWP) and climate systems.

### 2.2.2. Coupling Climate Simulations with AI

Full-fledged climate models are typically implemented in highly optimised Fortran to support efficient numerical integration, whereas most modern AI and RL workflows are developed in Python, leveraging its extensive ML ecosystem (Atkinson et al. 2025). Connecting these two ecosystems without introducing performance bottlenecks or creating significant development overhead requires specialised infrastructure. In this work, SmartSim (Partee et al. 2022) and its in-memory communication layer SmartRedis are used to bridge this gap, enabling direct tensor exchange between Fortran/Python-wrapped solvers and Python-based RL agents. This approach removes the overhead associated with file-based input–output (I/O) and complex foreign-function interfaces, while remaining fully compatible with HPC-scale execution.

SmartSim provides an orchestration layer for hybrid HPC–ML workloads by running simulation workflows alongside in-memory Redis databases (Sanfilippo and Noordhuis 2009), while SmartRedis offers efficient tensor exchange through native Fortran and Python APIs. In the ML context, a tensor is a multi-dimensional numerical array (often with gradient-tracking support for backpropagation) that serves as the fundamental unit of computation. By enabling direct in-memory communication of such tensors, SmartSim and SmartRedis allow numerical solvers and learning algorithms to interact seamlessly, ensuring low-latency coupling and supporting scalable in-situ training of AI models within large-scale climate simulations.

Coupling between the climate model and RL agents is achieved by encoding state and numerical model parameter tensors as named keys within the Redis database. Each MPI rank in the climate model periodically writes its state variables into Redis using the SmartRedis Fortran client (or Python client, if the model is Python-based). These tensors are then retrieved by Python-based RL agents via the SmartRedis Python API, where policy updates are computed and the modified parameters are written back under predefined keys. During the subsequent timestep, the climate model reads these updated parameters and incorporates them into the next-step integration, completing a continuous, closed in-memory loop of data exchange between the numerical model and the RL agents.

Coordination of RL agents across different spatial subdomains is achieved using `f1wr` (Flower) Beutel et al. 2022, a widely adopted (mostly Python based) federated learning framework. In this configuration, each MPI rank hosts a local RL agent that interacts with its subdomain, updates its parameters, and periodically synchronises with a central aggregator through a strategy such as FedAvg (McMahan et al. 2023). After every  $K$  episodes, all agents transmit their policy weights for aggregation, receive the globally averaged weights (in case of FedAvg), and resume local fine-tuning for the next  $K - 1$  episodes. SmartRedis manages the in-memory exchange of local climate states and model parameters between the distributed agents and the central `f1wr` process, preserving data locality while minimising communication latency.

## 2.3. climateRL Environments

### 2.3.1. Simple Climate Bias Correction Model (SCBC)

The Simple Climate Bias Correction (SCBC) environment defines a scalar temperature adjustment model in which the agent applies a heating control term  $u(t)$  to steer the model temperature  $T(t) \in \mathbb{R}$  towards a prescribed observational reference  $T_{\text{observed}} = 321.75$  K. The system evolves over discrete timesteps  $t \in \{0, 1, \dots, T_{\text{final}}\}$ , with the temperature updated in three steps.

In the first step, the current temperature is combined with a relaxation term that pulls the state toward a background physical temperature  $T_{\text{physics}}$ :

$$T^*(t+1) = T(t) + \varepsilon_1 \cdot \left( \frac{T_{\text{physics}} - T_{\text{current}}(t)}{T_{\text{physics}} - T_{\text{observed}}} \right), \quad (2.6)$$

where  $\varepsilon_1 = 0.2$  regulates the strength of this physical relaxation.

A second step introduces a bias-correction relaxation toward the target observation, yielding:

$$T^{**}(t+1) = T^*(t+1) + \varepsilon_2 \cdot \left( \frac{T_{\text{observed}} - T^*(t+1)}{T_{\text{physics}} - T_{\text{observed}}} \right), \quad (2.7)$$

with  $\varepsilon_2 = 0.1$ . This term acts as a nudging mechanism that applies a correction proportional to the residual discrepancy between the intermediate and observed states.

A third step adds a control mechanism using the heating control term  $u(t)$  for the RL agent to perform corrections towards the observed model temperature  $T_{\text{observed}}$  and create the final update:

$$T(t+1) = T_{\text{new}}(t) = T^{**}(t+1) + u(t), \quad (2.8)$$

Both the first and the second steps, utilise the normalisation factor  $(T_{\text{physics}} - T_{\text{observed}})^{-1}$ , ensuring that the updates scales consistently with the gap between the physical model and observations.

To improve numerical stability and prevent large temperature magnitudes from dominating the learning signal, all temperature values are normalised by first subtracting the freezing point of water (273.15 K) and then scaling by a factor of  $\frac{1}{100}$ :

$$T_{\text{observed}} \leftarrow \frac{T_{\text{observed}} - 273.15}{100}, \quad T_{\text{physics}} \leftarrow \frac{T_{\text{physics}} - 273.15}{100}. \quad (2.9)$$

Since all temperature variables are now scaled by a factor of 100, the agent's control action  $u$  (representing an additive heating term) is restricted to the interval  $[-1, 1]$ . This constraint ensures that perturbations remain physically realistic relative to the rescaled temperature state.

### Single-agent Environments

The SCBC (scbc) environments (simulation interfaces) are implemented as a Gymnasium (Gym) (Brockman et al. 2016; Towers et al. 2024) environment, providing a standardised RL interface for testing an agent's ability to correct systematic bias in a simplified programmable setup. The model simulates the evolution of temperature over 200 timesteps, with updates governed by the three-step formulation described above. Three variants are designed to progressively challenge the agent under different reward formulations, as described below:

**scbc-v0:** In this baseline variant, the reward at each timestep is defined in terms of the squared value of the bias-correction introduced:

$$r(t) = - \left[ \left( \frac{T_{\text{observed}} - T_{\text{new}}(t)}{T_{\text{physics}} - T_{\text{observed}}} \right) \cdot \varepsilon_2 \right]^2. \quad (2.10)$$

This reward explicitly reflects the model's nudging step and incentivises the agent to minimise the bias-correction required to track the observation.

**scbc-v1:** Here, the bias-correction step in Eq. (2.7) is disabled ( $\varepsilon_2 = 0$ ), and the reward is defined simply as the negative squared error between the current state and the observational target:

$$r(t) = -(T_{\text{observed}} - T_{\text{current}}(t))^2. \quad (2.11)$$

This design removes dependence on the model's internal bias term, yielding a more intuitive and stable training signal while requiring the agent to implicitly discover the bias-correction step.

**scbc-v2:** The third variant extends v1 by introducing temporal sparsity. The reward signal is provided only every five timesteps, with a constant penalty assigned otherwise:

$$r(t) = \begin{cases} -(T_{\text{observed}} - T_{\text{current}}(t))^2, & \text{if } t \bmod 5 = 0, \\ -1, & \text{otherwise.} \end{cases} \quad (2.12)$$

This sparse feedback setup tests the agent's ability to handle delayed rewards and encourages the agent to learn strategies that remain robust over extended horizons.

### 2.3.2. Radiative-Convective Equilibrium (RCE)

Radiative-Convective Equilibrium (RCE)(such as the one shown in (Manabe and Wetherald 1967)) represents the balance between radiative heating and convective transport in a vertical atmospheric column under horizontally homogeneous conditions. The prognostic variable is the temperature profile  $T(z, t)$  over height  $z \in [0, H]$ , governed

by the thermodynamic energy equation:

$$C_p \rho(z) \frac{\partial T}{\partial t}(z, t) = -\frac{\partial F_{\text{rad}}}{\partial z}(z, t) + Q_{\text{conv}}(z, t), \quad (2.13)$$

where  $C_p$  is the specific heat capacity at constant pressure,  $\rho(z)$  the density profile,  $F_{\text{rad}}$  the net vertical radiative flux, and  $Q_{\text{conv}}$  the convective heating rate.

At equilibrium ( $\frac{\partial T}{\partial t}(z, t) = 0$ ), radiative flux divergence is balanced by convective adjustment, producing a stable vertical temperature profile. The lower troposphere is maintained near a moist-adiabatic lapse rate through convective adjustment, while the stratosphere remains radiatively controlled. This simplified framework provides an ideal testbed for evaluating parametrisations of vertical energy transport.

### Single-agent Environments

The RCE environment (rce) is implemented using `climlab` (Rose 2018) on Gym to model a global-mean vertical temperature profile  $T(z)$ . Radiative transfer is computed with the Rapid Radiative Transfer Model for GCMs (RRTMG) scheme (Clough et al. 2005; Iacono et al. 2008), while convective adjustment (implemented as a separate module in `climlab`) enforces neutral stability by constraining the lapse rate to remain below a critical moist-adiabatic threshold. Within this setup, we introduce an RL agent which modifies a small set of physically interpretable parameters, such as the effective surface emissivity (inside RRTMG) and the critical lapse rate used in the convective adjustment process. The resulting profile is compared against reanalysis climatology, and the agent is rewarded for reducing errors. The reward at time  $t$  is defined as the negative mean-squared error between the simulated temperature profile and the NCEP/NCAR reanalysis long-term monthly mean profile,  $r(t) = -\frac{1}{N} \sum_{i=1}^N [T_{\text{simulated}}(t, z_i) - T_{\text{reanalysis}}(z_i)]^2$ , where  $N = 17$  denotes the number of vertical levels. This formulation provides a continuous and physically interpretable signal that encourages the agent to produce profiles consistent with observed climatology.

Three variants are tested in the RCE environment, each increasing in complexity by expanding the set of parameters controlled by the agent, as detailed below:

**rce-v0:** The baseline environment, where the agent learns two global scalar parameters: the effective longwave surface emissivity and the critical lapse rate for convective adjustment. This setup tests whether RL can learn bulk thermodynamic controls on the vertical profile.

**rce17-v0:** An extension of the baseline in which the agent learns a 17-dimensional vector of critical lapse rate values, one for each model layer. This formulation introduces additional vertical structure into the convective adjustment scheme, enabling the agent to represent non-uniform heating profiles.

**rce17-v1:** A further extension where the agent jointly learns the vertical profile of specific humidity along with the critical lapse rate values. This coupling links radiative transfer to water-vapour feedback, creating a more challenging problem that requires the agent to balance moist thermodynamics with radiative equilibrium.

To ensure physical realism and maintain numerical stability, the parameters controlled by the agent are restricted to bounded ranges. The critical lapse rate for convective adjustment is confined to  $\Gamma_{\text{crit}} \in [5.5, 9.8] \text{ } ^\circ\text{C km}^{-1}$ , spanning typical moist to dry adiabatic conditions. The effective surface emissivity is limited to  $\epsilon \in [0, 1]$ , representing the range from a perfect reflector to a black body. In the most complex variant, the specific humidity profile is bounded within  $q \in [0, 0.005] \text{ kg kg}^{-1}$ , ensuring realistic vertical moisture content. These constraints preserve thermodynamic consistency while preventing the RL agents from exploring non-physical regions of parameter space.

### 2.3.3. Budyko Energy Balance Model (EBM)

The Budyko–Sellers EBM (Budyko 1969; Sellers 1969; North 1975) (ebm) is an idealised, latitude-resolved model designed to represent the zonal-mean surface temperature  $T_s(\phi)$  as a function of latitude  $\phi$ . The model captures the balance between absorbed shortwave radiation, outgoing longwave radiation (OLR), and meridional heat transport (described using a downgradient diffusion assumption) governed by a zonal-mean energy balance equation:

$$C(\phi) \frac{\partial T_s}{\partial t} = \underbrace{(1 - \alpha(\phi))Q(\phi)}_{\text{absorbed shortwave}} - \underbrace{(A + BT_s)}_{\text{longwave cooling}} + \underbrace{\frac{D}{\cos \phi} \frac{\partial}{\partial \phi} \left( \cos \phi \frac{\partial T_s}{\partial \phi} \right)}_{\text{diffusive transport}}. \quad (2.14)$$

Here,  $C(\phi)$  denotes the effective heat capacity,  $\alpha(\phi)$  the albedo,  $Q(\phi)$  the insolation,  $A$  and  $B$  the OLR coefficients, and  $D$  the meridional heat transport parameter. These parameters are usually prescribed as constants tuned to match observations. The EBM is discretised into  $N = 96$  latitude bands for numerical integration of the temperature field. Within the FedRL framework,  $A$  and  $B$  are treated as adaptive control variables and are learnt as policy outputs of the RL agents. The reward is defined as the negative mean squared error (MSE):

$$r(t) = -\frac{1}{N} \sum_{i=1}^N [T_{\text{simulated}}(t, \phi_i) - T_{\text{observed}}(\phi_i)]^2,$$

between simulated and observed climatology to encourage learning control strategies that reduce regional biases.

### Single-agent and multi-agent climateRL Environments

Four environments (with progressive complexities) are developed from the Budyko–Sellers EBM (schematics in Figures 1–3, constraints in Table 1). Each environment introduces additional spatial decomposition or coupling complexity, providing a hierarchy of testbeds that gradually resemble the the distributed nature of full GCMs, described below:

**ebm-v0:** Implemented using `climlab` and Gym, this baseline configuration uses a single agent that observes the full temperature profile and learns five global scalar parameters:  $A$  and  $B$  (OLR coefficients),  $\alpha_0$  and  $\alpha_2$  (albedo terms), and  $D$  (diffusive heat transport) applied uniformly across all latitudes.

**ebm-v1:** This variant extends `ebm-v0` by allowing  $A$  and  $B$  to vary with latitude. The agent now learns a 96-dimensional vector for each parameter, observing the full temperature profile, enabling spatially localised adjustments to radiative properties.

**ebm-v2:** In this FedRL configuration, the latitudinal domain is divided into non-overlapping regions, each governed by a distinct RL agent operating on its own `climlab` EBM instance. Each agent observes the full temperature profile but learns region-specific values of  $A$  and  $B$ . Local rewards are computed via MSE, and agents synchronise after every  $K$  episodes using FedRL (implemented in `f1wr` with orchestration through SmartSim). This setup enables learning specific to conditions in local regimes while maintaining coordinated updates across regions.

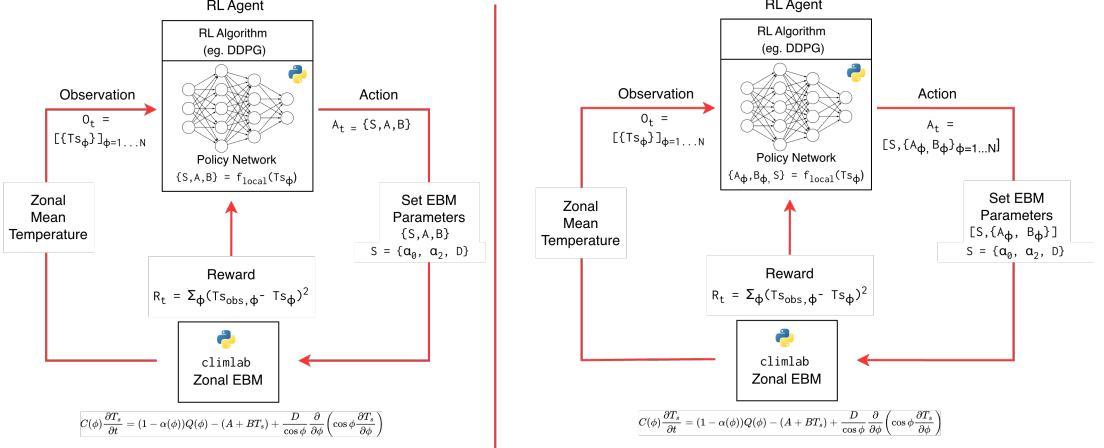


Figure 1: **ebm-v0** (left) and **ebm-v1** (right) single-agent setup. In **ebm-v1**, the global agent observes the full zonal-mean temperature profile and outputs latitude-dependent OLR parameters  $\{A_\phi, B_\phi\}$  as well as independent ones  $\{\alpha_0, \alpha_2, D\}$ . Loss from observations are computed over all 96 latitudes.

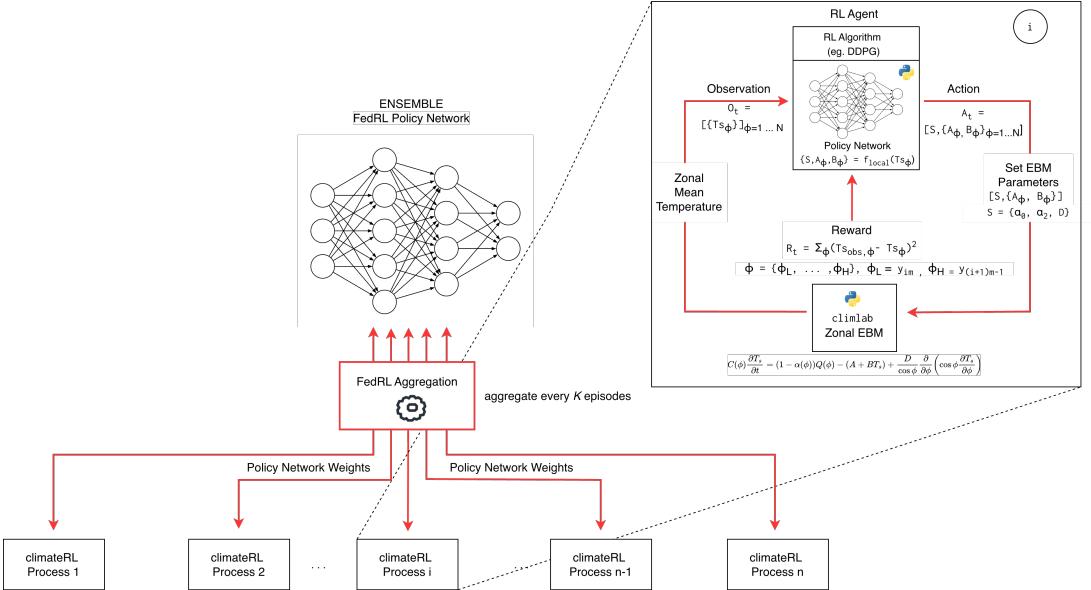


Figure 2: **ebm-v2** multi-agent ensemble with FedRL agents operate on assigned regions with local rewards while receiving the global profile as input. Periodic aggregation every  $K$  episodes synchronises policy weights across  $n$  agents.

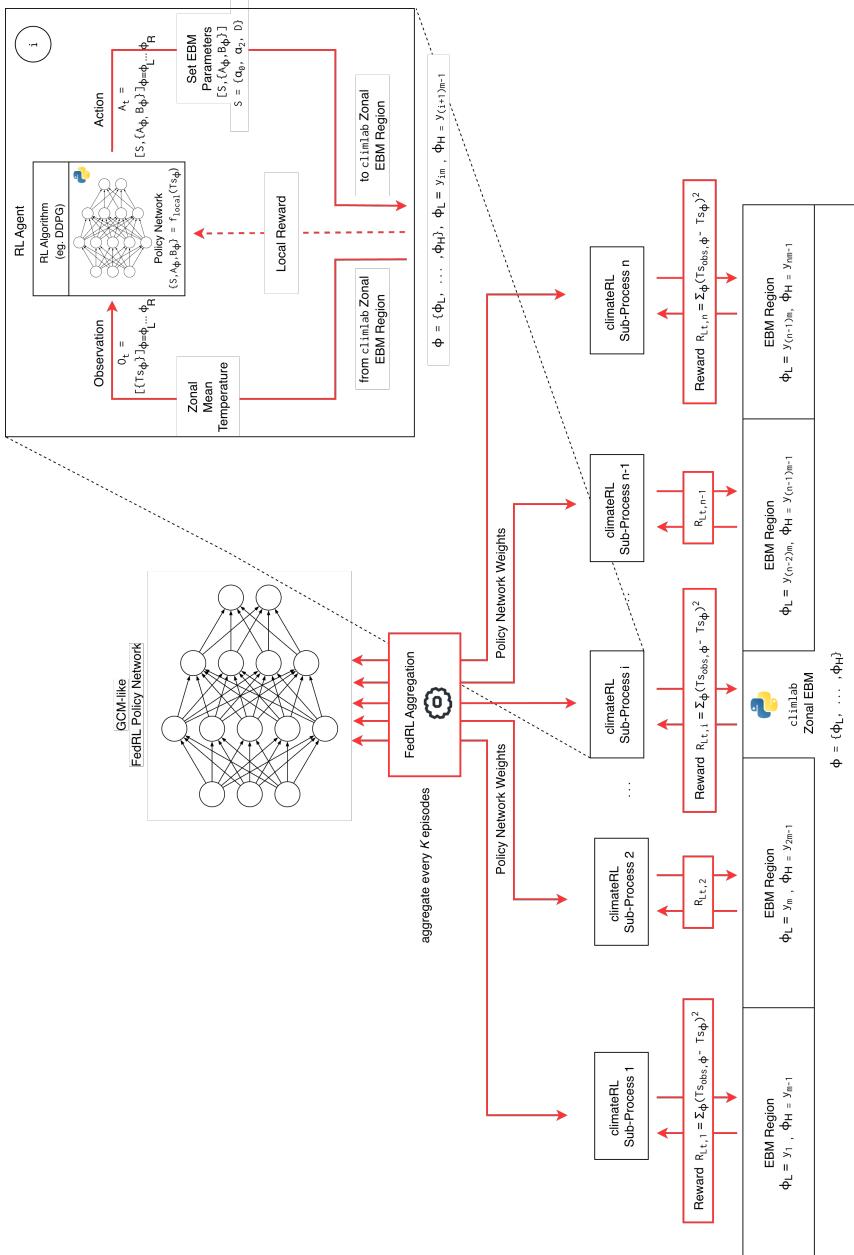


Figure 3: Schematics for GCM-like ebm-v3. FedRL agents observe local temperature slices only (unlike ebm-v2) and optimise local rewards. Local rewards are computed over regions for each agent. Policy weights are aggregated every  $K$  episodes via Flower for FedRL and also contribute towards a global non-local policy.  $\{\alpha_0, \alpha_2, D\}$  are averaged over all  $n$  sub-processes.

Parameter	Description	Range	Canonical Value
$A$	OLR intercept	$[140, 420] \text{ W m}^{-2}$	$210 \text{ W m}^{-2}$
$B$	OLR slope	$[1.95, 2.05] \text{ W m}^{-2} \text{ }^{\circ}\text{C}^{-1}$	$2 \text{ W m}^{-2} \text{ }^{\circ}\text{C}^{-1}$
$\alpha_0$	Albedo baseline	$[0.3, 0.4]$	0.354
$\alpha_2$	Albedo amplitude	$[0.2, 0.3]$	0.25
$D$	Diffusive transport	$[0.55, 0.65]$	0.6

Table 1: Parameter ranges used in the `ebm-v0/1/2/3` experiments

`ebm-v3`: Designed to more closely reflect operational GCMs, this variant introduces a central `climlab` parent process that integrates the global EBM state. Subdomain RL agents act on latitudinal subregions and transmit their updated  $A$  and  $B$  OLR parameters to the parent, which advances the full simulation and broadcasts the updated temperature profile back to each agent. Communication is handled by SmartSim with SmartRedis, while synchronisation of the policy network weights is performed via FedAvg after every  $K$  episodes. Together with a decentralised architecture, this setup simulates RL–numerical model coupling with realistic spatial hierarchy and efficient inter-process communication.

## 2.4. Experimental Setup

### 2.4.1. RL Algorithms

Nine RL algorithms (shown in Figure 4, summaries and pseudocodes in Appendices A.1 and A.2) are evaluated across the single-agent environments (`scbc-v0/1/2`, `rce-v0`, `rce-17-v0/v1`, and `ebm-v0/1`). These include five on-policy methods: REINFORCE (Williams 1992), TRPO (Schulman, Levine et al. 2015), PPO (Schulman, Wolski et al. 2017), DPG (Silver, Lever et al. 2014), and AVG (Vasan et al. 2024), which update policies using freshly sampled trajectories, and five off-policy methods: DDPG (Lillicrap et al. 2019), TD3 (Fujimoto et al. 2018), SAC (Haarnoja et al. 2018), and TQC (Kuznetsov et al. 2020), which utilise replay buffers and target networks to improve sample efficiency. Except for REINFORCE, a pure policy-gradient algorithm, all methods follow an actor–critic structure. Table A.1 summarises their architectures and main features. For the multi-agent environments (`ebm-v2/3`), only the top three performers from the single-agent experiments are carried forward for evaluation.

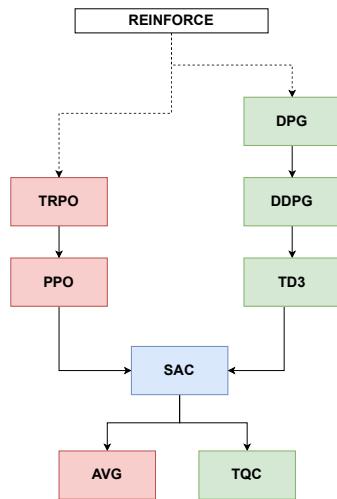


Figure 4: Developmental progression of RL algorithms, evaluated in this study. The single agent climateRL experiments span classical policy-gradient methods such as REINFORCE, on-policy algorithms — DPG, TRPO, PPO, AVG, and advanced off-policy actor–critic approaches — DDPG, TD3, SAC, TQC. Adapted from [https://master-dac.isir.upmc.fr/rl/12\\_sac.pdf](https://master-dac.isir.upmc.fr/rl/12_sac.pdf).

## 2.4.2. Hyperparameter Tuning

Hyperparameter optimisation is performed using Ray (Moritz et al. 2018) (a distributed computing framework) on a SLURM-managed cluster, with one head node and three workers (each equipped with 8× AMD EPYC 74F3 cores). For algorithms such as TQC, which require additional hardware acceleration due to multiple critic networks, GPUs are partitioned into four logical devices using Ray to enable efficient parallel trials. Advanced sampling during the search is handled using Optuna (Akiba et al. 2019) (a hyperparameter optimisation framework) within Ray. For each RL algorithm, 32 parallel experiments are launched on JASMIN (Lawrence et al. 2012), enabling a full sweep of the nine-algorithm suite within a three-hour walltime. For multi-agent environments (ebm-v2/3), the best hyperparameters identified for the single-agent baseline (ebm-v1) are reused.

## 2.4.3. Evaluation Strategies

### Single-agent RL

Single-agent environments (scbc-v0/1/2, rce-v0, rce-17-v0/v1, and ebm-v0/1) are evaluated under two architectural setups. In the first setup, optim-L, actor-critic network sizes are tuned individually to capture environment-specific complexity. In the second, homo-64L, all networks are fixed to 64 hidden units per layer to enforce uniformity. Each setup is run under two tuning regimes: a constrained regime that mirrors a scenario where full convergence run may not be practical due to compute limitations, and an extended regime with longer tuning horizons (Table 2). Together, these design choices yield 32 distinct experiment configurations, summarised in Appendix A.2.

Environment ID	Tuning ID	Tuning Timesteps	Episode Length
scbc-v0/1/2	homo-64L	2000	200
	optim-L		
rce-v0 rce17-v0/1	homo-64L-60k	60000 - TOTAL	500
	optim-L-60k		
ebm-v0/1	homo-64L	5000	200
	optim-L		
	homo-64L-10k	10000 - TOTAL	
	optim-L-10k		
	homo-64L	10000	
	optim-L		
	homo-64L-20k	20000 - TOTAL	
	optim-L-20k		

Table 2: Optimisation timesteps for hyperparameter tuning in each environment. Tuning is performed on seed 1 across specified timesteps/episodes. Episode lengths are in timesteps.

Since RL lacks a universal train/test split procedure unlike supervised learning (Patterson et al. 2024), evaluation is based on three complementary metrics:

- Sample efficiency:** Measured by the number of numerical models steps  $N_{\text{to\_threshold}}$  required to cross an empirically defined return threshold (Table 3). These thresholds roughly correspond to ankle points in training curves and quantify how quickly an algorithm acquires a useful policy.
- Policy stability:** Defined as the variance  $\sigma_{\text{after\_threshold}}^2$  of episodic returns once the threshold has been crossed, indicating the consistency and reliability of performance over extended training.
- Asymptotic performance:** Measured as the difference  $\Delta_{\text{from\_10k/20k/60k}}$  between the final return at the end of the experiment and the return threshold, reflecting the long-horizon convergence behaviour of the algorithm.

Environment	Threshold	Error (in K) per Episodic Step
SimpleClimateBiasCorrection-v0	-0.25	$\pm 0.035$
SimpleClimateBiasCorrection-v1	-2.718	$\pm 0.116$
SimpleClimateBiasCorrection-v2	$-1 \times (160 + 2.718)$	$\pm 0.116$
RadiativeConvectiveModel-v0	-43900	$\pm 9.370 (0.551)$
RadiativeConvectiveModel17-v0	-43700	$\pm 9.348 (0.550)$
RadiativeConvectiveModel17-v1	-43650	$\pm 9.343 (0.549)$
EnergyBalanceModel-v0	-10000	$\pm 7.071 (0.074)$
EnergyBalanceModel-v1	-30000	$\pm 12.247 (0.127)$

Table 3: Empirically determined episodic return thresholds for each RL environment. Error is computed using the formula  $\sqrt{\frac{\text{Threshold}}{\#\text{Timesteps per episode}}}$ . Sparse rewards in SimpleClimateBiasCorrection are implemented in the Gym environment by assigning a constant upper-bound normalised temperature error of 1 at every timestep, except every 5th step (when the reward is activated), yielding a minimum episodic return of  $200 - \frac{200}{5} = 160$ . Errors shown in brackets for RadiativeConvectiveModelEnv and EnergyBalanceModelEnv are averaged across 17 pressure levels and 96 latitudes respectively.

This multi-metric evaluation, averaged across 10 random seeds, balances sample efficiency (metrics (a) and (c)) with robustness (metric (b)), providing a holistic view of learning behaviour. Benchmark return thresholds, listed in Table 3, are empirically derived from observed learning curves and represent episodic return values that indicate meaningful learning, serving as a consistent reference point for analysis.

**1. Rank-based Composite Scoring:** To compare algorithms across metrics, we adopt a rank-based composite evaluation. Each algorithm is ranked in ascending order for metrics (a)–(c), and an additional penalty is imposed if policy variance exceeds environment-specific thresholds (e.g.,  $3 \times 10^{-3}$  for SCBC and  $3 \times 10^5$  for RCE and EBM environments). Final scores are obtained by summing ranks across all metrics and sorting in ascending order. This procedure, commonly used in benchmarking studies (Ikhtiarudin et al. 2025), enables fair comparison across heterogeneous regimes while maintaining interpretability.

**2. Post-Ranking Diagnostics:** To better understand top-performing algorithms, additional diagnostics are run in inference mode with fixed policy weights. In the RCE setup, mean absolute error (MAE) in temperature is computed against reanalysis data at 100, 200, and 1000 hPa, capturing both upper-tropospheric and near-surface behaviour. For the EBM, the 96 latitudes are aggregated into six  $30^\circ$  zones, where area-weighted RMSE (areaWRMSE), and model bias are evaluated against the reference climatology for both RL-assisted and vanilla climlab run which uses a static fitted solution obtained by calibrating the EBM parameters against the target climatology via linear regression. These diagnostics add to the physical meaning that extend beyond episodic return, quantifying fidelity to observations and highlighting model realism.

**3. Temporal Evolution:** In addition to reward- and error-based evaluations, the SCBC, RCE, and EBM environments are examined by visualising the temporal evolution of the state dependent values (i.e., the agent’s actions). Tracking these trajectories shows how policies learn with the evolving physics-based state, providing insight into agent stability and convergence. Such visual diagnostics reveal whether policies converge during learning, oscillate, or follow trends consistent with physical processes, helping to distinguish meaningful learning from artefacts of exploration or instability.

#### Multi-agent RL

To assess scalability and coordination in distributed settings, the top three algorithms from the single-agent EBM experiments are re-run in the multi-agent FedRL environments. For consistency, ebm-v2 and ebm-v3 reuse the best hyperparameters obtained for the single-agent baseline (ebm-v1). Three aggregation regimes are considered: fed05, where parameters are synchronised every five episodes, fed10, where synchronisation occurs every ten episodes, and nofed, where agents train independently for the full 20k timesteps. Each regime is tested under two spatial decompositions: a 2-zone split (northern vs. southern hemisphere) and a 6-zone split ( $30^\circ$  latitude bands spanning polar, mid-latitude, and tropical regions). Together, these choices yield 12 experimental configurations, detailed in Appendix A.3.

Training curves from these multi-agent setups are benchmarked against the single-agent `ebm-v1` baseline to evaluate convergence dynamics and assess whether spatial decomposition accelerates learning. The `areaWRMSE` post-ranking diagnostic introduced earlier is applied separately to both `ebm-v2` and `ebm-v3` for locally fine-tuned and globally averaged policies. These analyses provide a consistent measure of performance across federated configurations and highlight potential benefits of decentralised, regime-aware learning.

### 3. Results and Discussion

#### 3.1. Experimental Outline

A structured four-step protocol is followed to ensure robust and reproducible evaluation of RL-assisted parametrisation schemes. The process begins with hyperparameter tuning, followed by multi-seed training, post-training skill assessment, and finally FedRL evaluation. An overview of this workflow is provided in Figures 5 and 6.

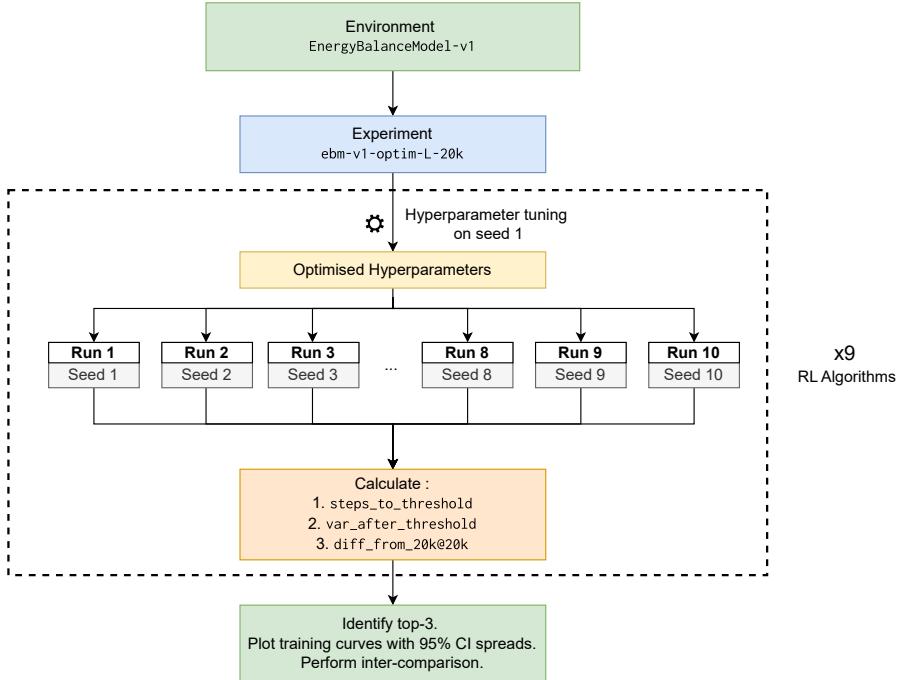


Figure 5: Schematic of the experimental workflow for the single agent climateRL experiments (e.g., ebm-v0/1). The process begins with hyperparameter tuning on seed 1, followed by evaluation across 10 random seeds and evaluation metric computation (steps-to-threshold, variance-after-threshold, and final return difference) for all nine algorithms. Top-3 algorithms are then selected and training curves analysed with 95% confidence intervals.

**1. Hyperparameter Optimisation:** Hyperparameters for each RL algorithm are tuned using Optuna with 100 trials on seed 1. The default TPESampler, which implements the Tree-structured Parzen Estimator (TPE) algorithm (Watanabe 2023), is used. TPE uses kernel density estimation (KDE) to model the distributions of promising and less promising configurations separately, directing the hyperparameter search towards regions likely to yield better performance. Each trial is run for a fixed number of timesteps (Table 2), with the objective pre-defined as the maximum episodic return in the final episode. The best configuration is selected and reused for all subsequent experiments in that environment.

**2. Multi-seed Evaluation:** To capture robustness and variability, the tuned hyperparameters are then tested across 10 random seeds (1–10). For each seed, the RL agent is retrained from scratch and executed for the full experiment duration, with results recorded independently. Aggregated training curves from these runs are used to rank algorithms, with the evaluation strategies described earlier identifying the top three performers.

**3. Skill Assessment and Action Interpretability:** The top-3 algorithms are subsequently evaluated in inference mode, where the trained policy is executed for one episode without updating weights, mimicking deployment in an operational setting. Performance is measured using previously defined metrics, with results reported as mean and

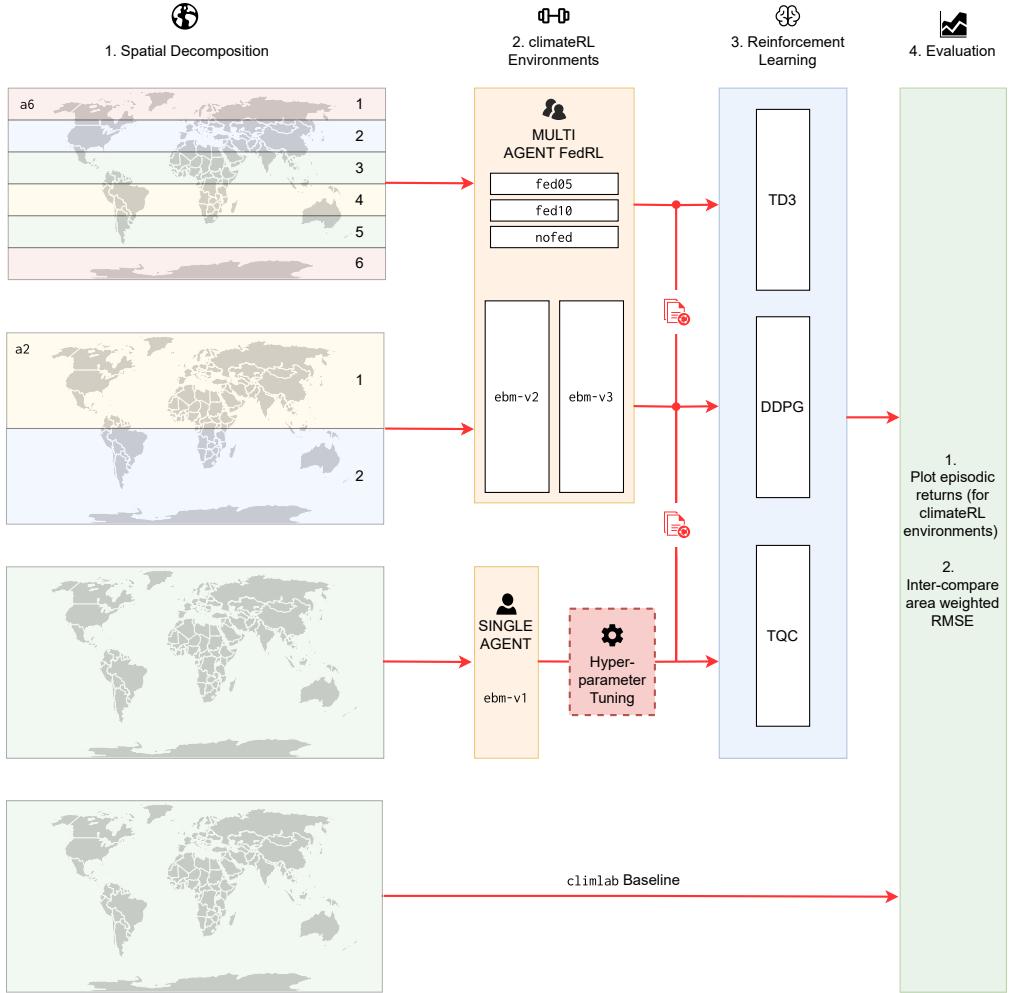


Figure 6: Pipeline for the ebm-v1/2/3 experiments. The process begins with configuring the Budyko–Sellers EBM in either single-agent (ebm-v1) or spatially decomposed multi-agent forms (ebm-v2, ebm-v3) using two (a2) or six (a6) regions. Agents are trained with one of three RL algorithms (DDPG, TD3, TQC) under FedRL coordination schemes fed05, fed10, or nofed. In multi-agent settings, policies are periodically aggregated via FedRL every  $K$  episodes. Hyperparameters tuned for ebm-v1 are transferred over to ebm-v2/v3. Finally trained models are assessed on their training curves and benchmarked against a static climlab baseline, using a skill measure such as areaWRMSE across  $30^\circ$  latitude groups.

standard deviation across 10 seeds for both RCE and EBM environments. In addition, inference trajectories of agent actions are analysed to provide insight into policy behaviour.

**4. Federated RL and Global Policy Evaluation:** For multi-agent experiments (ebm-v2 and ebm-v3), performance metrics such as areaWRMSE are applied not only to fine-tuned local policies but also to the aggregated global policy obtained at intermediate steps via FedRL. This enables direct comparison (even though having different reward structures), highlighting whether non-local policies yield improved or degraded skill relative to decentralised local policies.

### 3.2. Single-agent RL

#### 3.2.1. SCBC Environment

##### Training Dynamics

Figure 7 and Appendix B.1.1 presents the training dynamics for all SCBC variants (`scbc-v0/v1/v2`) under four configurations: `optim-L`, `optim-L-60k`, `homo-64L`, and `homo-64L-60k`. Rewards generally converge within 10k steps, and longer budgets (60k) do not consistently improve stability. In `scbc-v2-optim-(L/L-60k)`, TD3 is the only algorithm that reliably ranks among the top-3 across seeds, with other methods exhibiting greater variability. Among different tuning configurations, the `optim-L` and `homo-64L` setups converge more quickly and with lower variance.

Rank	Algorithm	Frequency
1	TD3	10
2	TQC	9
3	DDPG	9
4	DPG	6
5	SAC	2

Table 4: Top-3 appearance frequency for each RL algorithm across all SCBC runs (10 seeds)

Among the top-performing algorithms, TD3, DDPG, and TQC show rapid initial learning followed by stabilisation. TQC occasionally achieves slightly lower episodic returns in certain configurations, while DDPG and TD3 often track each other closely. By contrast, DPG exhibits large uncertainties at isolated timesteps (suggestive of catastrophic forgetting (Ven et al. 2025)) which undermines its overall reliability. Across the three environment variants, `scbc-v1` emerges as the most stable. In the `optim-L`, `homo-64L`, and `homo-64L-60k` setups, TD3 and DDPG yield nearly overlapping reward trajectories across all seeds. Overall, TD3 stands out as the most robust and consistently reliable algorithm under the evaluation framework of Section 2.4.3, appearing in the top-3 across every seed. This trend is also confirmed in Table 4, where TQC and DDPG also feature prominently in the top-3 rankings.

##### Temperature Corrections

Figure 8 shows the offline skill evaluation of the top-3 RL algorithms: TD3, TQC, and DDPG, trained under the `scbc-v0-optim-L-60k` configuration. The red dashed line marks the observed target temperature (321.75 K). The control SCBC model without RL and with relaxed physics consistently overshoots the observed target, stabilising at 380 K. Two key observations follow: (1) once the target temperature is reached, maintaining it requires a steady heating increment of roughly -0.2, as can be inferred mathematically from the SCBC dynamics in Section 2.3.1, and (2) only TQC and DDPG (not TD3) appear among the top-3 algorithms for this experiment.

DDPG learns a stable heating profile centred around -0.2, locking the model output precisely to the target with no variability across seeds. TD3 produces a similar mean profile but with larger, persistent variance in the heating increments. This broader uncertainty band indicates unstable learning, leading some seeds to overshoot the target, a likely reason why TD3 fails to appear in the top-3 in this experiment. TQC, in contrast, adapts its heating actions dynamically while maintaining the observed temperature throughout the episode. The RL-assisted SCBC model closely follows the target temperature with low variance across seeds, indicating that TQC learns a robust, state-aware control policy capable of correcting the bias in the baseline parametrisation.

These results show that DDPG and TQC demonstrate strong skill in bias correction and heating control. The contrasting behaviour of TD3, despite strong training performance under the `scbc-v0-optim-L` experiment, highlights the sensitivity of RL algorithms to hyperparameter choices and environment configuration.

#### 3.2.2. RCE Environment

##### Training Dynamics

Figure 9 and Appendix B.1.2 shows the training dynamics across the RCE environments: `rce-v0`, `rce17-v0`, and `rce17-v1`, under four configurations: `optim-L`, `optim-L-10k`, `homo-64L`, and `homo-64L-10k`. Across all cases, convergence is rapid, typically within 2k–4k timesteps (4-8 episodes). The `rce-v0` variant displays the most

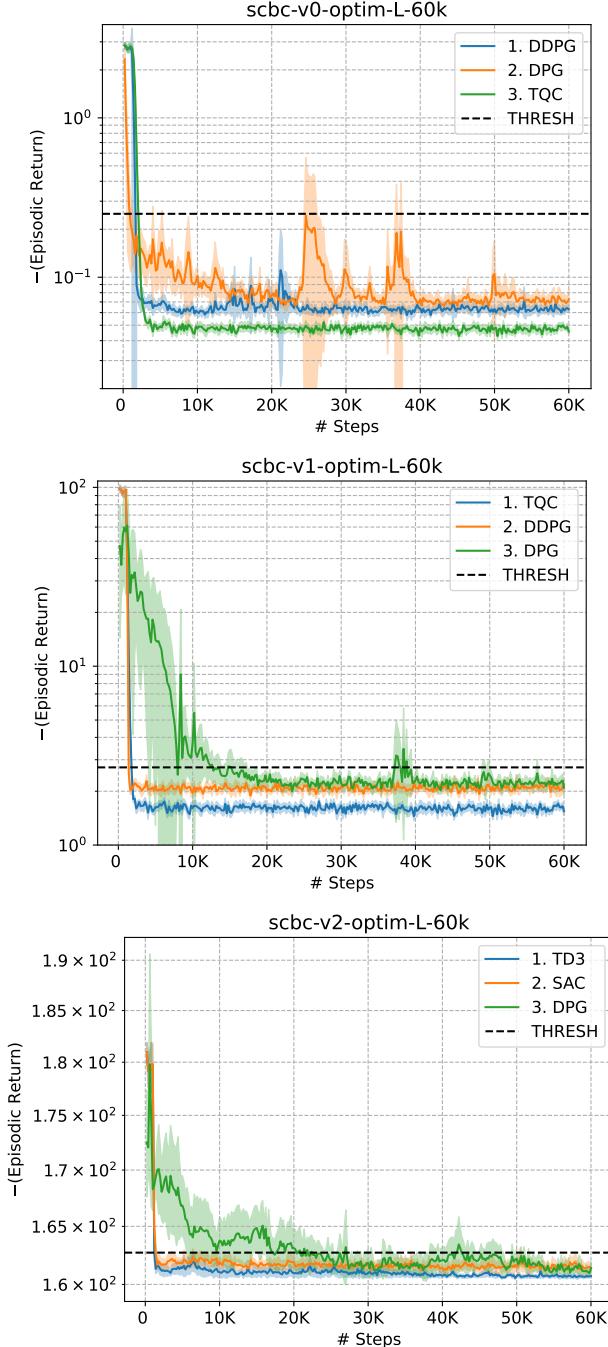


Figure 7: Training curves across 10 seeds for SCBC environments (v0 - with bias-correction, v1 - w/o bias-correction, v2 - with sparse rewards) under the **optim-L-60k** configuration. Episodic returns (each episode = 200 steps) are plotted on a log scale. Shaded regions denote  $\pm 1.96$  standard deviation (95% confidence intervals). The top-3 RL algorithms (DPG, DDPG, TD3, TQC and SAC) in scbc-v0/1/2-optim-L-60k environments are shown. Threshold values are mentioned in Table 3. Training curves for other configurations are in Appendix B.1.1.

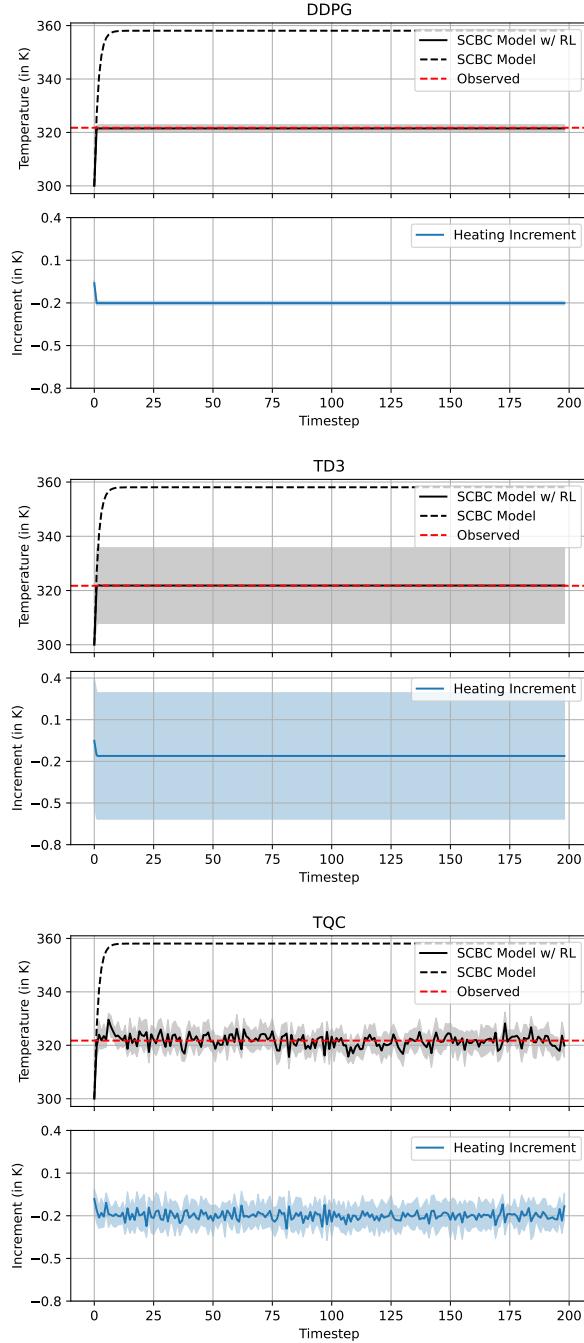


Figure 8: Heating increment dynamics for `scbc-v0-optim-L-60k`. Top panels: temperature evolution compared to the observed temperature (321.75 K). Bottom panels: normalised heating increments applied by the RL agent. Shaded regions denote  $\pm 1.96$  standard deviation (95%). Black dashed line indicates the SCBC model w/ bias-correction as reference. The reference SCBC model stabilised around 358 K driven by the cancellation between the relaxation term and the bias-correction components.

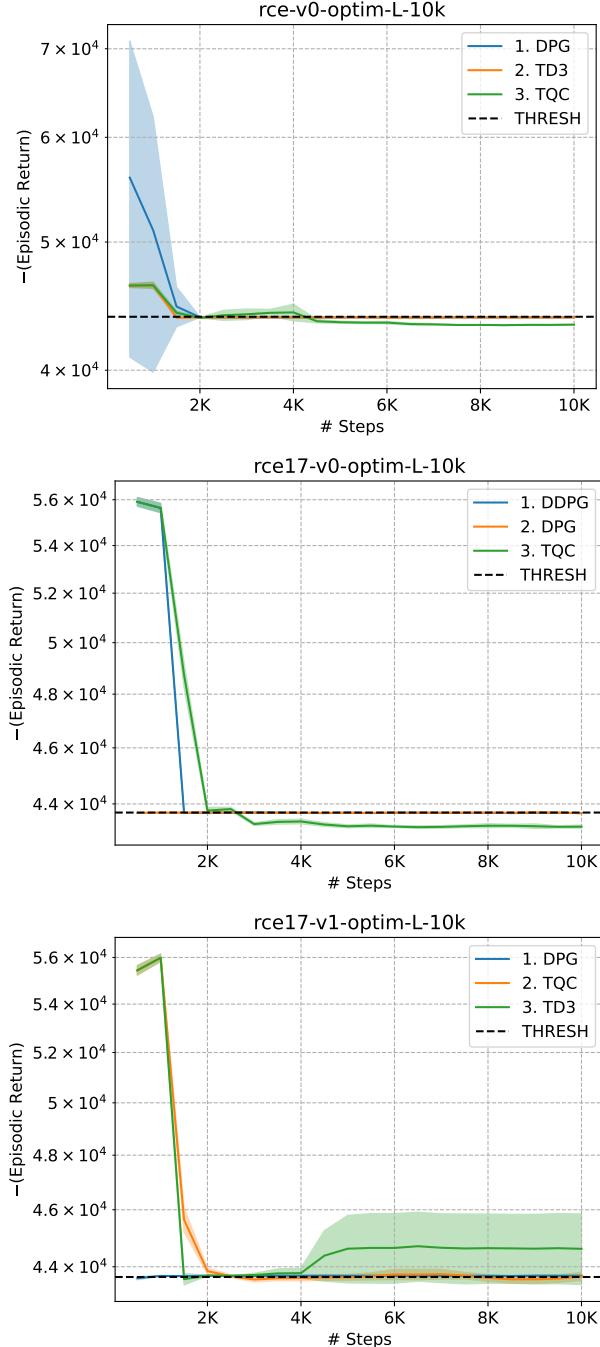


Figure 9: Training curves for RCE environments (`rce-v0`, `rce17-v0`, `rce17-v1`) under the `optim-L-10k` configuration. Episodic returns (each episode = 500 steps) are plotted on a log scale. Shaded regions denote  $\pm 1.96$  standard deviation (95% confidence intervals). Threshold values are mentioned in Table 3. DPG, TD3 (`rce-v0`) and DDPG, DPG (`rce17-v0`) converge close to the threshold value. Training curves for other configurations are in Appendix B.1.2.

stable post-convergence behaviour, with reduced spread and tight clustering across algorithms. While DPG shows notable early variance in the `optim-L` setup compared to PPO and TRPO, this instability diminishes with extended tuning. In general, convergence is sharp and variance small across all RCE setups, though some wider spreads occur for DPG in `rce-v0`, DDPG in `rce17-v0-optim-L`, and TD3 in `rce17-v1-optim-L-10k`.

According to the ranking framework in Section 2.4.3, DPG emerges as the most reliable algorithm, consistently appearing in the top-3 across all seeds and configurations (Table 5). TQC and DDPG also perform strongly, particularly in the `rce17` environments, with TQC occasionally edging ahead in episodic return, consistent with its stability and sample efficiency. TD3, which performed well in SCBC, is less reliable in RCE and did not feature in the top-3.

Rank	Algorithm	Frequency
1	DPG	10
2	TQC	8
3	DDPG	7
4	TD3	5
5	PPO	3
6	TRPO	2
7	SAC	1

Table 5: Top-3 appearance frequency for each RL algorithm across RCE runs (10 seeds)

Overall, the RCE environments present a well-conditioned optimisation landscape characterised by fast convergence and low reward oscillation. However, algorithm rankings are less consistent across configurations than in SCBC, as reflected in the wider spread of top-3 frequencies, suggesting greater sensitivity to hyperparameter variation and environmental noise in the RCE setting.

### Skill Evaluation

Figure 10 shows the mean absolute error at 100 hPa, 200 hPa, and 1000 hPa for the RCE `optim-L-10k` configuration. The baseline `climlab` RCE model exhibits substantial temperature biases, particularly at 200 hPa, where errors exceed 5–17.5°C. By contrast, RL-assisted (DPG, DDPG, and TQC) models consistently show reduced errors across 100 hPa and 200 hPa. At 1000 hPa in `rce-v0`, the baseline model performs well (relative to the RL-assisted models) with its fixed lapse rate of 6.5°C km<sup>-1</sup> unlike in the `rce17` environments where the moist adiabatic lapse rate (MALR) scheme is applied.

Skill remains robust across different RCE environments and training setups. DPG delivers the most stable performance, with narrow confidence intervals across levels and consistently lowest errors at 100 hPa, outperforming DDPG and TQC, especially where the baseline bias is the second largest. TQC achieves competitive results at 200 hPa, though with slightly wider uncertainty bands. DDPG shows moderate skill, outperforming the baseline in nearly all cases (except at 1000 hPa in `rce-v0` and `rce17-v0`), but exhibits greater variance, particularly in the `rce-v0` and `rce17-v0` setups.

All RL algorithms show great skill in reducing errors at 100 hPa and 200 hPa. These levels correspond to the tropopause and mid- to upper troposphere, regions that are highly sensitive to model parametrisations. Improvements at these atmospheric heights suggest that RL agents are capable of learning assistive parametrisation components that reduce biases in predicted variables..

### Vertical-Level Corrections

Figure 11 illustrates the temperature trajectories and the evolution of RL-controlled actions at 100 hPa, 200 hPa, and 1000 hPa for TQC in the `rce17-v0-optim-L-10k` setup. At 100 hPa, the RL-assisted model diverges (around timestep 100) from the vanilla RCE profile before stabilising near the observed target, accompanied by a steady upward adjustment of the critical lapse rate that reduces vertical mixing in the upper atmosphere. At 200 hPa, typically one of the levels with high bias in the baseline, the agent sustains a low critical lapse rate and maintains a moderate temperature gradient. In both layers, critical lapse rate adjustments mirror the shape of the temperature profile: a sharper curvature at 100 hPa near the tropopause, where the lapse rate values approach the canonical 6.5 °C km<sup>-1</sup> ( $\approx 0.23$  in the normalised scale), and a more gradual transition at 200 hPa. At 1000 hPa, however, the agent is less effective in reducing the near-surface bias. Despite jointly adjusting the critical lapse rate and surface emissivity, the

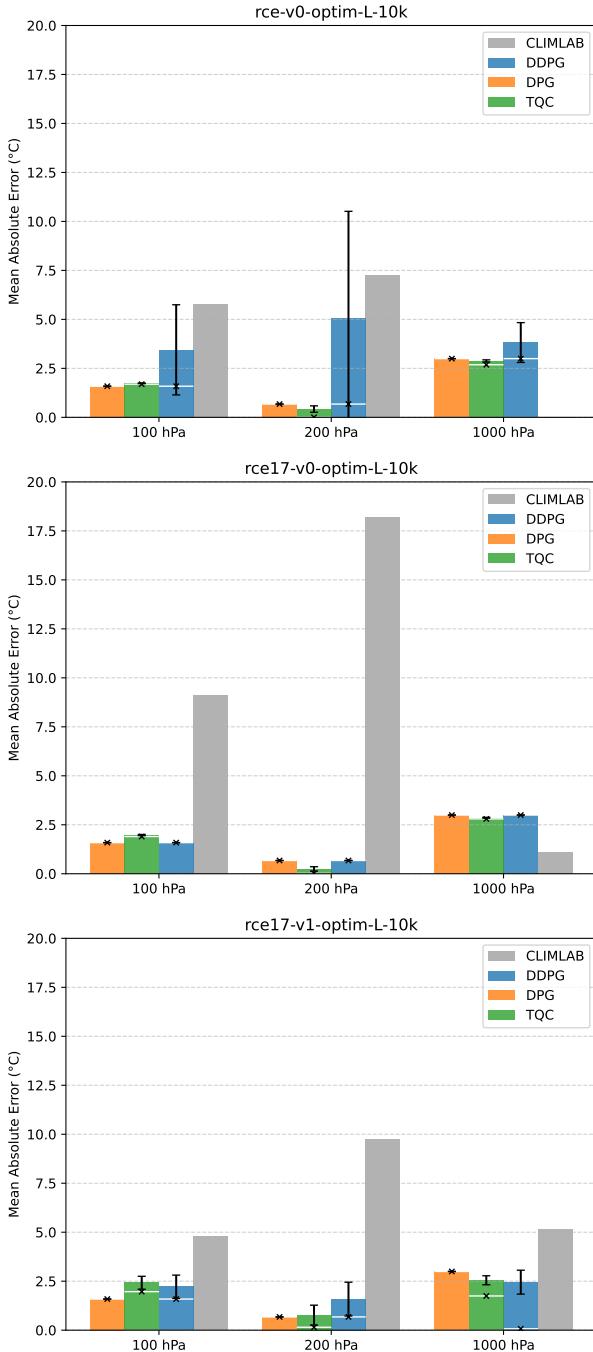


Figure 10: Mean absolute temperature error (°C) at 100 hPa, 200 hPa, and 1000 hPa for the optim-L-10k configuration. White horizontal bars with a cross indicate the best-performing seed for each algorithm. Error bars represent 95% confidence intervals over 10 seeds. The zero error for the vanilla climlab model at 1000 hPa in rce-v0 is with a constant lapse rate set at 6.5 (unlike MALR in rce17-v0 and MALR with water vapour coupling in rce17-v1).

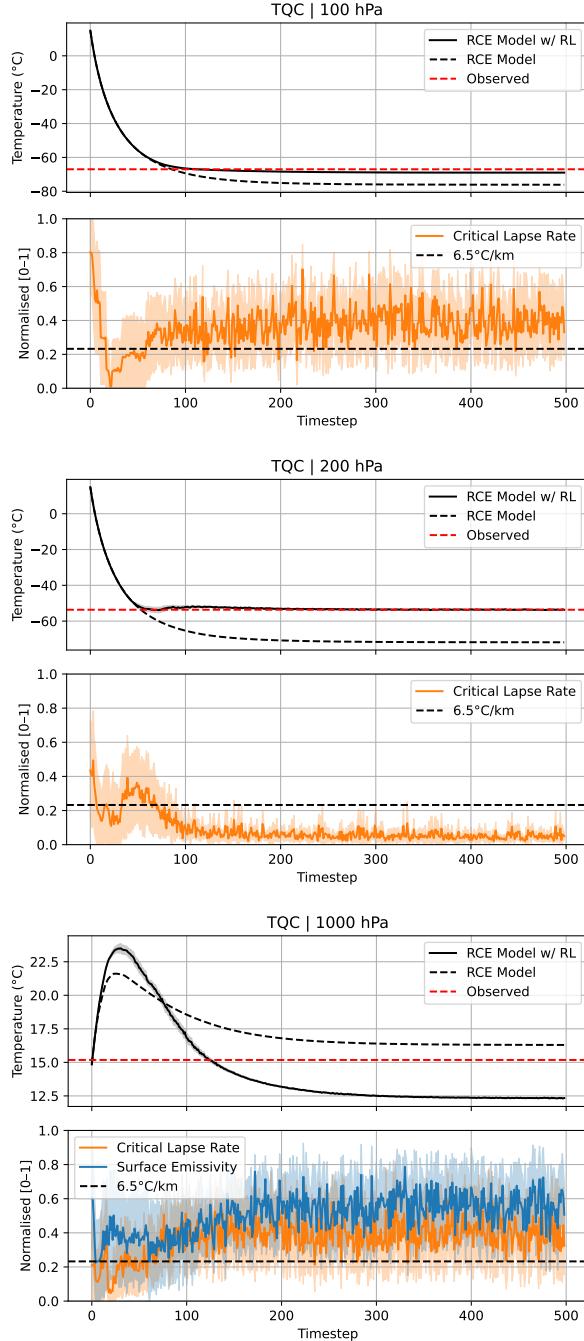


Figure 11: Top: Temperature trajectories at 100 hPa, 200 hPa, and 1000 hPa for TQC in `rce17-v0-optim-L-10k`. Bottom: Evolution of normalised critical lapse rate and surface emissivity. RL-assisted models closely track observations at 100 and 200 hPa, replacing fixed parameters by values that vary by level and as a function of model state. Shaded regions denote  $\pm 1.96$  standard deviation (95%). Canonical value of  $6.5^{\circ}\text{C km}^{-1} \approx 0.23$  in the normalised scale.

model remains offset from observations, with the high critical lapse rate suppressing vertical mixing and the increased surface emissivity (over the timesteps) likely enhancing OLR, altering the surface energy to drive surface cooling.

This dual-parameter modulation highlights the agent’s attempt to coordinate multiple physical controls, by responding to the changing state of the model rather than converging to a static solution. Yet the persistent surface bias indicates that more sophisticated corrections may be required. In particular, the structural simplicity of the convection scheme may limit its ability to simultaneously match temperatures across all vertical levels, even when parameters are allowed to vary with state.

### 3.2.3. Zonal EBM Environments

#### Training Dynamics

Figure 12 and Appendix B.1.3 presents training dynamics for the EBM environments `ebm-v0` and `ebm-v1` under four configurations: `optim-L`, `optim-L-20k`, `homo-64L`, and `homo-64L-20k`. In contrast to the SCBC and RCE experiments, the EBMs, particularly `ebm-v0`, display substantially higher variance and less stable convergence, due to their tightly coupled dynamics making small parameter perturbations produce large, temperature changes. Learning generally stabilises only after about 10k steps, with some algorithms (TRPO and SAC in `ebm-v0`) exhibiting catastrophic forgetting or oscillation during early training. TQC is the only method to maintain a consistently stable profile across all seeds. Increasing the tuning budget from `optim-L` to `optim-L-20k` (except in `ebm-v0`), as discussed in Section 2.4.3, improves both stability and smoothness of convergence, suggesting that the EBM reward landscape demands more extensive exploration than the simpler SCBC and RCE environments.

Rank	Algorithm	Frequency
1	TQC	8
2	TD3	6
3	DDPG	4
4	SAC	4
5	TRPO	1

Table 6: Top-3 appearance frequency for each RL algorithm across single-agent EBM runs (10 seeds)

Across all algorithms, TQC emerges as the most robust and consistently reliable, ranking highest in all eight configurations (Table 6). TD3 achieves moderate performance but exhibits higher variance, while DDPG and SAC struggle to stabilise, particularly under shorter tuning budgets. TRPO, the only on-policy method to feature within the top-3, makes only a marginal contribution, appearing once among the top performers. Performance and stability are strongly influenced by both RL algorithm choice and tuning duration, emphasising the importance of careful tuning for this class of geophysical experiments.

#### Skill Evaluation

Figures 13 and 14 jointly assess EBM skill using areaWRMSE and area weighted zonal mean temperature bias across six latitude bands. The baseline `climlab` model exhibits particularly large errors around Antarctica ( $90^{\circ}\text{S}$ – $60^{\circ}\text{S}$ ), consistent with its ocean only structure and lack of land representation, generating a biased pattern (due to structural limitations) that is warm over Antarctica and cool elsewhere, with pronounced Southern Ocean warming and widespread cooling biases. RL assisted runs using the best performing seeds reduce areaWRMSE across most latitudes and substantially shrink these biases, with the most consistent gains in `ebm-v1`, where learning  $A$  and  $B$  per latitude provides sufficient flexibility to partially compensate for the structural errors over Antarctica. However, the inter-seed spread remains large, emphasising the importance of algorithmic stability and careful hyperparameter tuning in these parameter sensitive settings. Among the top three methods, TQC is the most robust, showing narrow confidence intervals in areaWRMSE and the most uniform bias reductions across zones, whereas TD3 and DDPG typically outperform the baseline but with greater variance, including overshooting in `ebm-v0` and introducing spurious tropical warming, whilst also showing underestimation tendencies in `ebm-v1`. Taken together, the areaWRMSE and model bias diagnostics indicate that TQC not only lowers zonal errors but also yields more physically meaningful spatial patterns than either TD3 or DDPG.

#### State-Dependent Variations in $A$ and $B$ at Different Latitudes

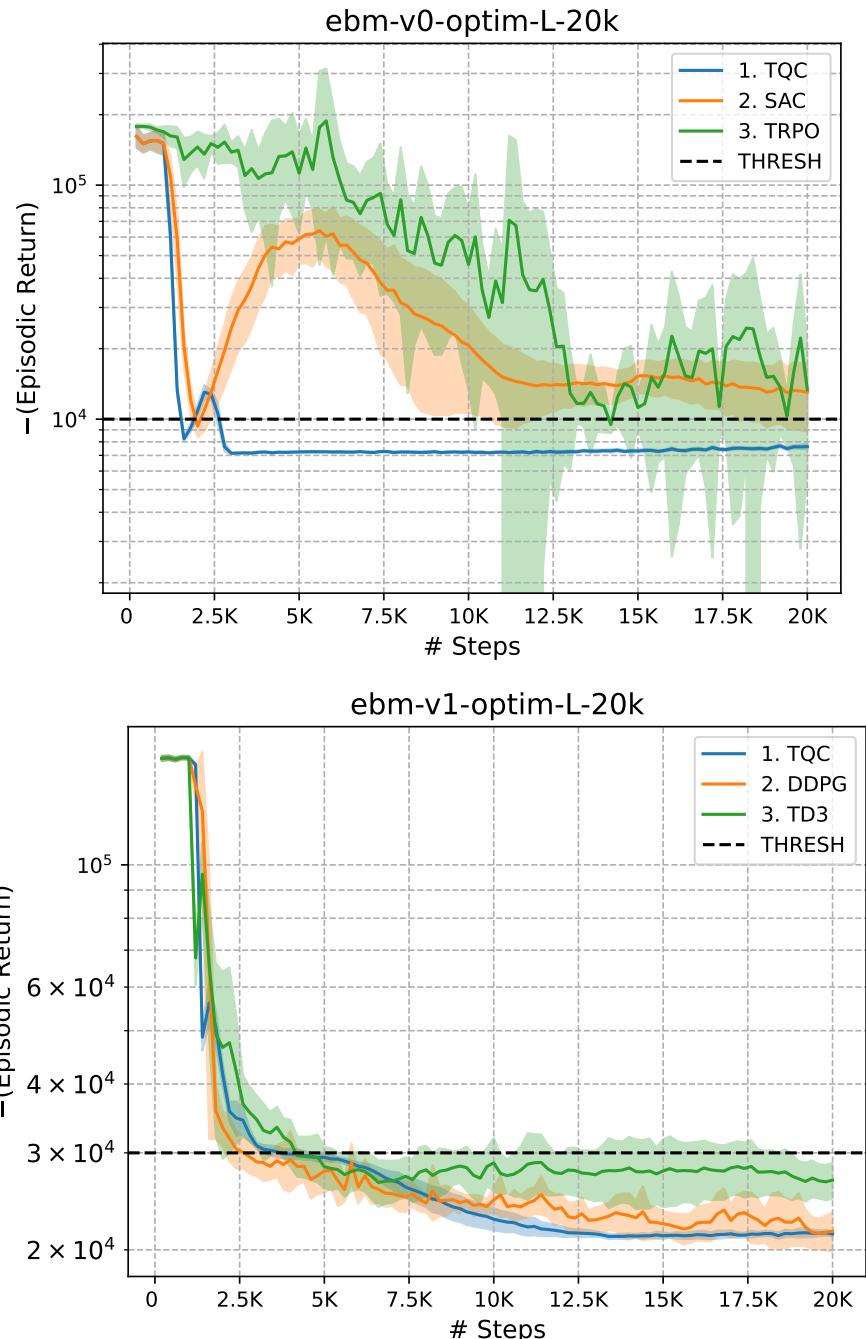


Figure 12: Training curves for single-agent EBM environments (ebm-v0, ebm-v1) under the optim-L-20k tuning regime. Episodic returns (each episode = 200 steps). Shaded regions denote  $\pm 1.96$  standard deviation (95% confidence intervals). Threshold values are mentioned in Table 3. Training curves for other configurations are in Appendix B.1.3.

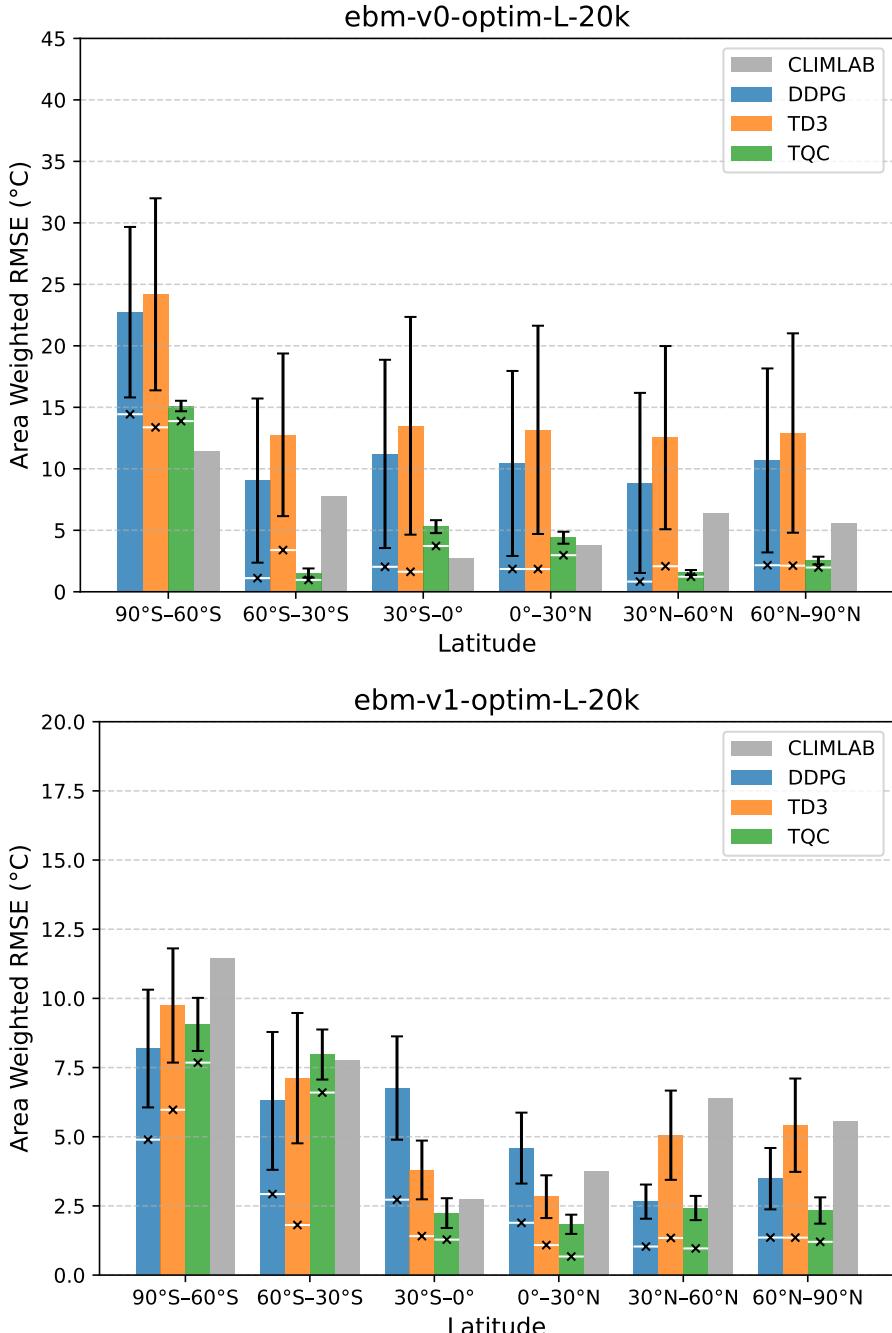


Figure 13: areaWRMSE of zonal mean temperatures across six latitude bands for three experiments under ebm-v0 and ebm-v1. Skill is evaluated using areaWRMSE between predicted and reference zonal temperature profiles, averaged with 95% confidence intervals over 10 seeds. White horizontal bars with a cross indicate the best-performing seed for each scheme.

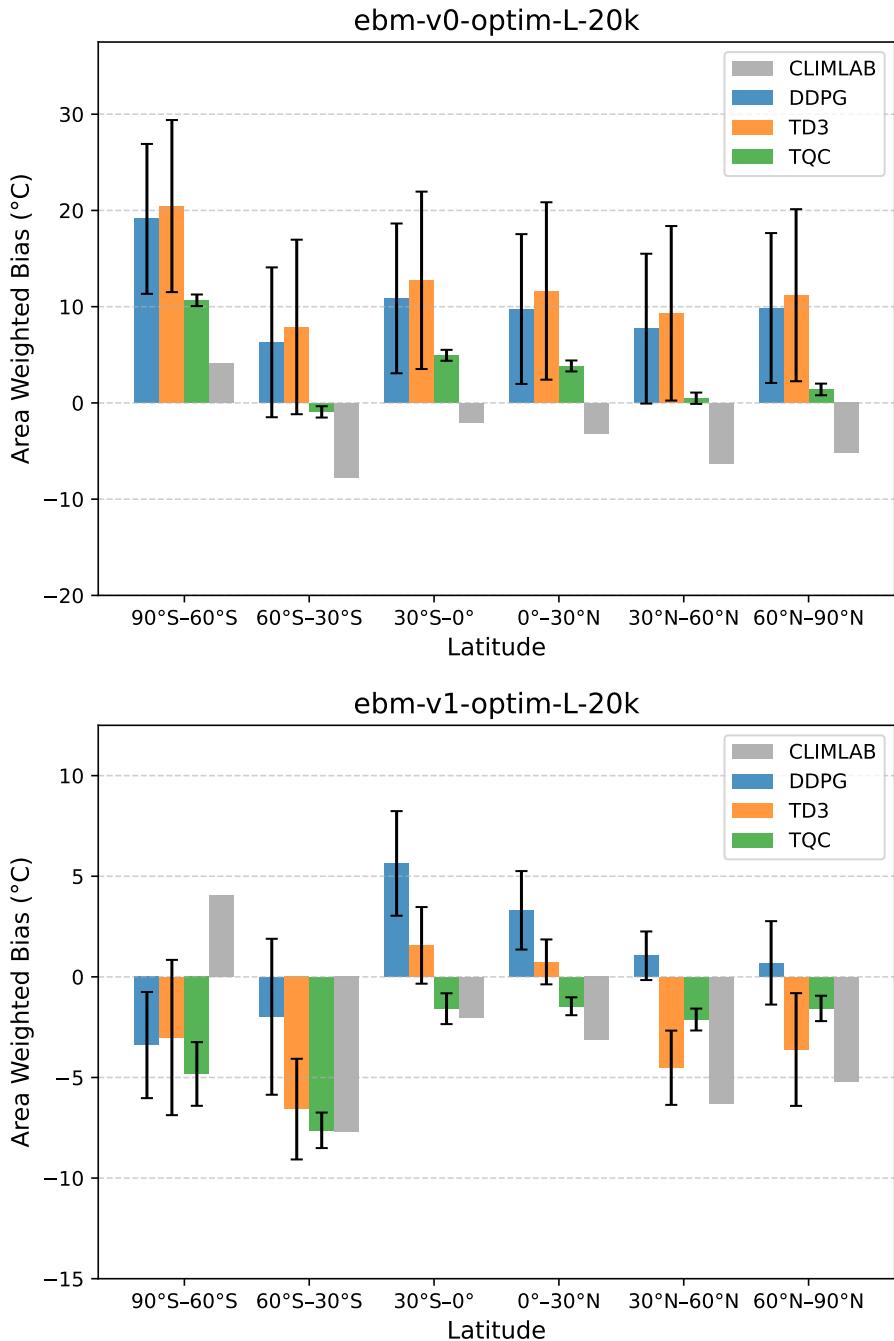


Figure 14: Area-weighted zonal mean temperature bias averaged with 95% spreads over 10 seeds across six latitude bands for ebm-v0 and ebm-v1 under top-performing RL algorithms. Negative values indicate underestimation and positive values indicate overestimation of temperature relative to observations.

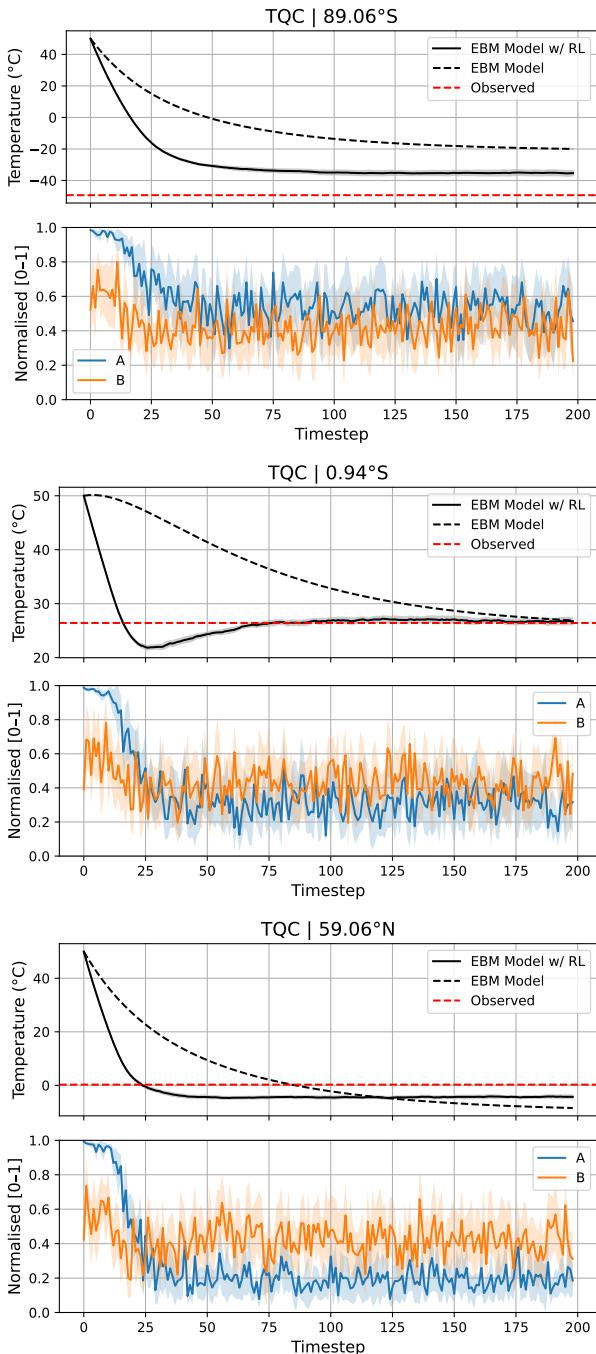


Figure 15: Temperature trajectories (top) and normalised actions (bottom) for parameters  $A$  and  $B$  under TQC at 89.06°S, 0.94°S, and 59.06°N in the `ebm-v1-optim-L-20k` experiment. Shaded regions denote  $\pm 1.96$  standard deviation (95% confidence intervals).

Figure 15 shows the temperature trajectories and the evolution of radiative parameters  $A$  and  $B$  at three representative latitudes: 89.06°S, 0.94°S, and 59.06°N, under the `ebm-v1-optim-L-20k` configuration with TQC. At the South Pole, the baseline `climlab` model produces strong warm biases due to structural limitations. The RL agent compensates by stabilising  $A$  and  $B$  between 0.4 and 0.6 (relative to the reference  $A = 0.25$ ), enhancing radiative cooling and reducing bias. At the tropics and mid-latitudes, final biases are smaller: the agent makes a sharp change to  $A$  (in response to the changing global temperature profile) around timestep 25, then gradually refines the parameters, stabilising both  $A$  and  $B$  between 0 and 0.4. Because the model is initialised from a warm isothermal state of 50°C, this early suppression of  $A$  reflects the agent’s attempt to rapidly dissipate excess heat, while  $B$  remains comparatively steady, suggesting that adjustments are driven primarily through  $A$  with minor adjustments of  $B$ . These latitude-specific corrections demonstrate the agent’s ability to learn geographically dependent, modulating radiative balance in accordance with regional climate conditions.

### 3.3. Multi-agent RL

#### Training Dynamics

Figure 16 compares training dynamics of multi-agent FedRL in `ebm-v2/3-optim-L-20k` under `fed05` with the single-agent baseline `ebm-v1`. Relative to the global single-agent setup, convergence in the multi-agent `a2` (hemispheric) configuration is both faster and more stable, with learning stabilising around 5k–7.5k steps compared to slower convergence beyond 10k steps in `ebm-v1`. In the `ebm-v2` hemispheric setup, convergence is nearly steady for DDPG, TD3, and TQC, with only minor fluctuations and low variance across seeds. In `ebm-v3`, DDPG remains the most stable with the least inter-seed variance, while TD3 exhibits moderate fluctuations but trends upward over time. TQC, despite strong performance in single-agent experiments, undergoes catastrophic forgetting mid-training, suggesting greater sensitivity to the increased complexity of multi-agent coordination.

#### Local Skill Evaluation

Figure 17 shows that across nearly all latitude bands, `fed05` outperforms both the static baseline and the non-federated (`nofed`) setups, with strongest gains in the tropics relative to other best performing alternatives. In both `ebm-v2` and `ebm-v3`, areaWRMSE is reduced by more than 50% in 30°S–0° and 0°–30°N relative to `ebm-v1`. Improvements are strongest in `ebm-v3`, with region-specific sliced inputs. By contrast, `fed10` still improves on `nofed` but mostly shows higher variance and less consistent benefits than `fed05` in `ebm-v2` and `ebm-v3`, underscoring the importance of frequent aggregation (`fed05`) for stable coordination. In polar regions, all federated schemes match or surpass `ebm-v1`, indicating that local specialisation helps to resolve regions like Antarctica better (despite structural limitations). Even under coarse decomposition (`a2`, Appendix B.2.1), DDPG in `ebm-v2/3` achieves monotonic convergence and low final errors, highlighting its robustness across spatial setups. Overall, these results confirm the benefits of regional specialisation through FedRL and demonstrate DDPG’s stability and efficiency under varying reward structures and input resolutions, making it well-suited to GCM-style architectures. Additional results for TD3 and TQC (Appendix B.2.2) show that while competitive at times, both suffer from higher variance and instability, particularly under frequent aggregation or in equatorial and polar regions.

#### Globally Uniform Policy Skill Evaluation

Figure 18 shows inference with globally aggregated non-local policies (`-GLOBAL`) degrades performance, and collapsing region-specific strategies into a single policy increases both areaWRMSE and inter-seed variance. While DDPG (in `fed05-GLOBAL`) preserves some regional fine-tuning benefits, these are diminished in the global aggregation, with `fed10-GLOBAL` performing worse due to infrequent synchronisation. In `ebm-v2` (see Appendix B.4), DDPG under the `a2` setup yields modest gains in the tropics and southern mid-latitudes, but in `ebm-v3` errors increase further, with robustness declining relative to the `climlab` baseline. Overall, while FedRL supports effective local specialisation, globally aggregated policy rollouts struggle to reconcile heterogeneous regimes, although more frequent aggregation (`fed05-GLOBAL`) offers slightly greater stability than `fed10-GLOBAL`.

### 3.4. Discussion

#### 3.4.1. Convergence and Stability of RL Algorithms

Across all experiments, the convergence behaviour of RL agents is strongly environment-dependent. The single-agent SCBC and RCE environments generally yield smoother and more stable training curves owing to their simpler

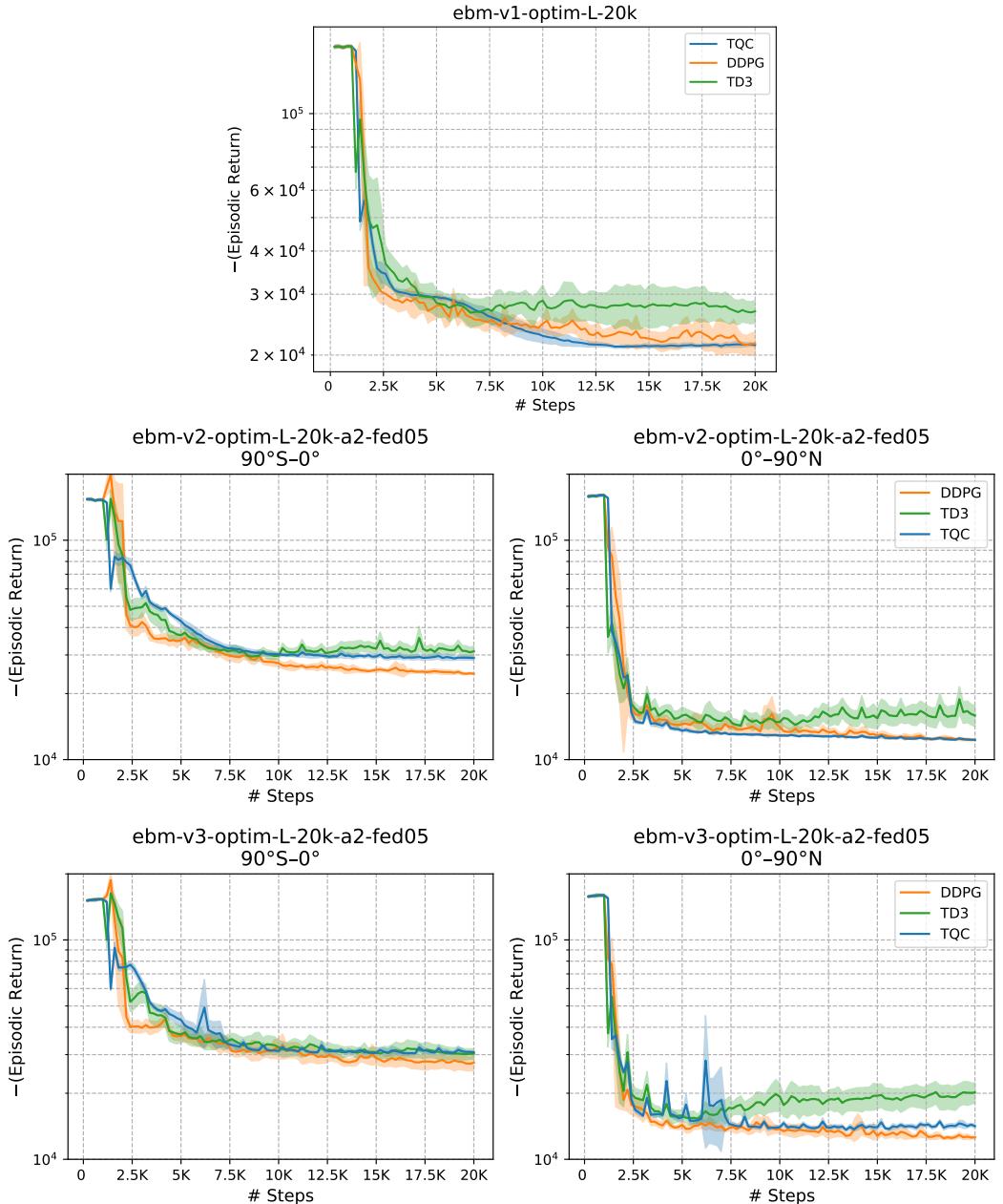
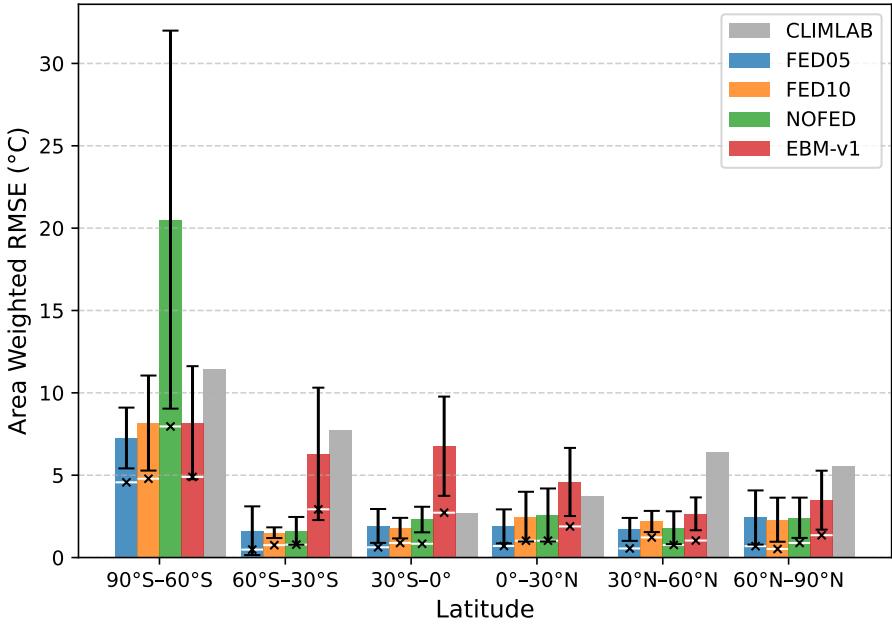


Figure 16: Episodic return curves (log-scaled) with 95% CI spreads over 10 seeds for three RL algorithms: TQC, DDPG, and TD, across three climateRL environments. Left: ebm-v1 (single-agent setup with global input, global reward, and latitude-specific parameters; reproduced from Figure 12 for comparison). Middle: ebm-v2 (multi-agent FedRL configuration with shared global profile input and local rewards). Right: ebm-v3 (multi-agent FedRL configuration with partitioned inputs and local rewards, closely resembling GCM-style spatial decomposition). ebm-v2/3 training curves are from a2 (hemispheric decomposition) with fed05 (aggregated every 5 episodes) setting. Threshold not shown for ebm-v1, as top-3 algorithms are already identified.

DDPG | ebm-v2-optim-L-20k-a6



DDPG | ebm-v3-optim-L-20k-a6

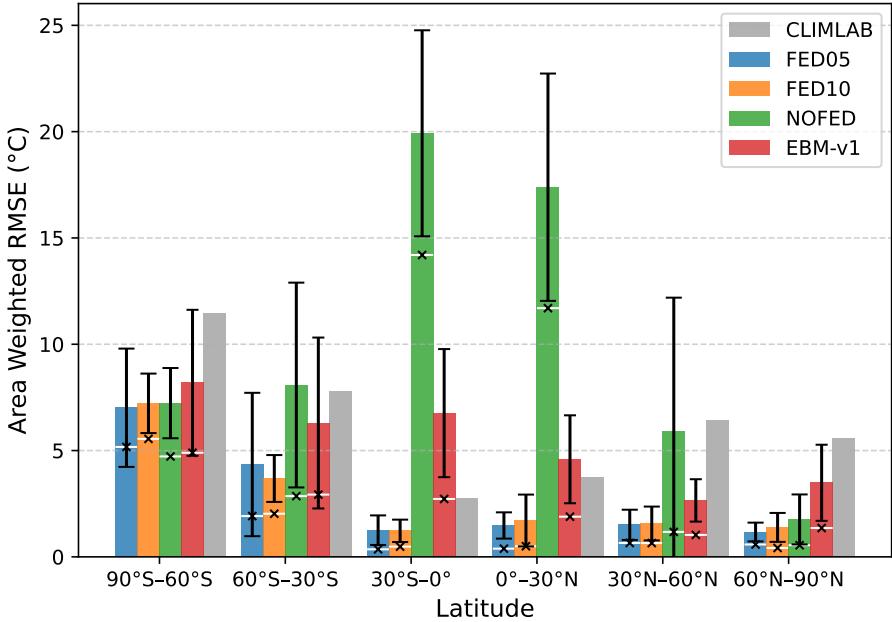
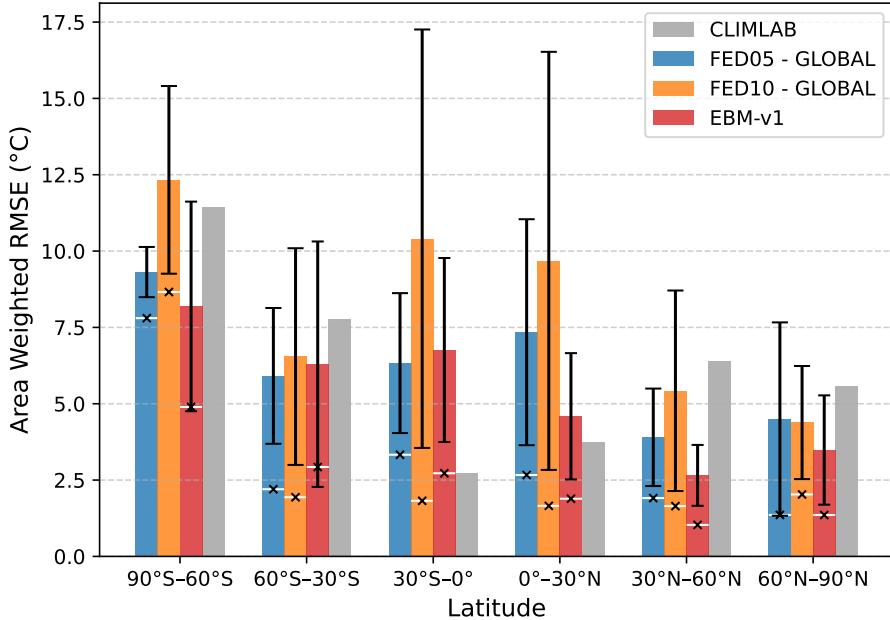


Figure 17: Comparison of zonal skill achieved by DDPG under FedRL coordination in ebm-v2 and ebm-v3, both using the 6-agent spatial decomposition (a6). Skill is evaluated using areaWRMSE between predicted and reference temperature profiles, averaged with 95% CI spreads over 10 seeds. Each subplot reports results for three FedRL schemes: fed05, fed10, nofed, along with single-agent ebm-v1 and the static climlab baseline. White horizontal bars with a cross indicate the best-performing seed for each scheme. Both setups adopt the same policy network design and hyperparameters as ebm-v1. Results for a2 and TD3/TQC in Appendix B.2.2.

DDPG |  $\text{ebm-v2-optim-L-20k-a6}$



DDPG |  $\text{ebm-v3-optim-L-20k-a6}$

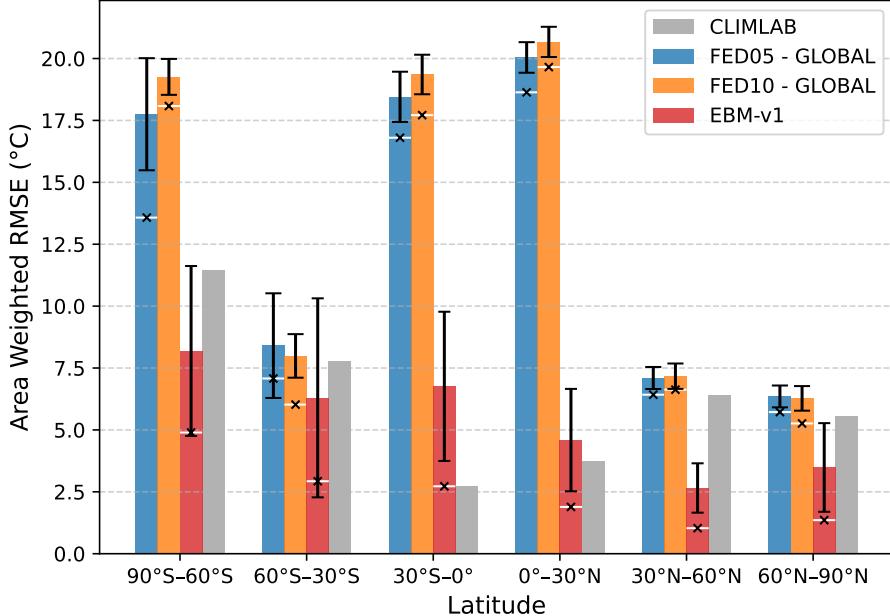


Figure 18: Comparison of global policy zonal skill achieved by DDPG under FedRL coordination in  $\text{ebm-v2}$  and  $\text{ebm-v3}$ , both using the 6-agent spatial decomposition (a6). Each subplot reports results for three FedRL schemes: `fed05-GLOBAL`, `fed10-GLOBAL`, along with single-agent  $\text{ebm-v1}$  and the static `climlab` baseline.

reward landscapes and lack of spatial coupling. By contrast, both `ebm-v0` and `ebm-v1` exhibit greater inter-seed variance and require longer training budgets for convergence. This reflects the intrinsic difficulty of EBMs, where each action affects global temperature gradients through diffusive interactions.

Among the nine RL algorithms evaluated, TQC consistently delivers the most stable and robust performance across environments (with the exception of `a6` in FedRL settings). Its ensemble of quantile critics (Kuznetsov et al. 2020) provides fine-grained value estimation, enabling stable learning under high-variance returns. This strength is most evident in long-horizon tuning experiments (`optim-L-20k`) and in federated setups with frequent aggregation (`fed05` and `fed10`), where TQC achieves narrow confidence intervals across most latitude bands (with high latitudes being a notable exception). By maintaining a diverse critic ensemble, TQC is better equipped to mitigate noisy gradients and non-stationary policy updates, challenges that are inherent in climate environments with coupled feedbacks.

DDPG also emerges as a strong performer, frequently ranking in the top-3 across SCBC, RCE, and both single- and multi-agent EBM experiments. While it lacks the critic ensemble of TQC, its lightweight actor–critic architecture offers computational efficiency with a smaller memory and compute footprint. DDPG remains competitive under both short and long tuning budgets and shows robustness across spatial decompositions such as `a2` and `a6`. Its relative simplicity makes it particularly attractive for large-scale or resource-constrained deployments where training must be distributed across many agents. Unlike TQC, which depends on GPU acceleration to handle ensembles of critics, DDPG offers a more tractable solution for operational climate applications where computational cost and action interpretability are both critical, making it a practical baseline for geophysical RL, especially in federated contexts.

DPG and TD3 demonstrate promising but less reliable performance. DPG exhibits high inter-seed variance and catastrophic forgetting, especially in early training phases, as seen in the RCE experiments. This behaviour likely reflects its lack of target networks, replay buffers, and deeper critic architectures, which leaves its value estimates sensitive to Q-value errors. Despite these limitations, DPG occasionally achieves top-3 performance, indicating that its simplicity can still yield effective policies under favourable loss landscapes. TD3 improves upon DDPG by introducing twin critics to reduce overestimation bias and clipped noise for stabilising target actions, though at the cost of additional hyperparameters. Nevertheless, TD3 suffers from instability in certain FedRL EBM configurations, particularly in equatorial bands where sharp policy shifts and transient collapses are observed.

FedRL decomposition further improves stability by reducing the dimensionality and scope of each agent’s learning task. In the `ebm-v2` and `ebm-v3` experiments, spatially decomposed multi-agent setups, particularly under the `a6` configuration, converge much faster than single-agent setups, with most runs stabilising well before 10k steps. This acceleration arises from localised reward signals and simplified policy search spaces when agents operate over narrower latitude bands. Spatial decomposition thus enables more specialised learning through periodic policy aggregation. However, setups with more agents such as `a6` also introduce higher sensitivity to hyperparameters: both TD3 and TQC show catastrophic forgetting, suggesting that parameters tuned for single-agent global setups may not transfer directly to multi-agent regional FedRL environments with modified inputs and reward landscapes.

### 3.4.2. Skill Evaluation Across Latitudes

Single-agent RL models show heterogeneous skill across latitude bands, shaped by both climatic dynamics and reward structures. In the zonal EBM experiments, tropical and mid-latitude regions consistently exhibit lower bias and narrower confidence intervals, while polar zones (especially 90°S–60°S) show higher variance due to structure limitations in the EBM due to absence of land representations. Algorithms such as TQC maintain robust skill even under these high-variance conditions, whereas TD3 and DDPG display localised instabilities, including overshooting near the tropics and mid-latitudes.

FedRL enhances zonal skill by allowing agents to specialise in their own regions while synchronising periodically through global aggregation. In `ebm-v2` and `ebm-v3`, frequent aggregation (`fed05`) consistently improves accuracy in tropical and mid-latitude bands, with TQC (in `ebm-v2`) and DDPG often outperforming their single-agent baselines. Finer decomposition (`a6`) produces sharper corrections in warmer zones, while coarser setups (`a2`) yield more stable performance at the poles. In `ebm-v3`, where input states are restricted relative to `ebm-v2`, TD3 and TQC show a deterioration in zonal skill compared with `ebm-v1`, reinforcing the need for environment-specific tuning when reward formulations or inputs differ.

Deploying a globally aggregated policy during inference introduces a trade-off: while it reduces memory overheads by collapsing policies into a single set of weights, it weakens region-specific adaptations. DDPG emerges as the most stable across both local and global settings, particularly under `a6`, whereas TQC, though strong in

localised fine-tuning, suffers significant performance degradation when globally averaged. These results highlight the tradeoff between specialisation and generalisation in FedRL setups. Overall, federated coordination with well-chosen hyperparameters enables scalable and regime aware learning, outperforming both static baselines and globally trained single-agent RL.

### 3.4.3. Physical Interpretability and Alignment

A key advantage of the proposed RL framework is the physical interpretability of learnt policies, particularly in geophysical environments where actions correspond to parameters within the parametrisation schemes. In the single-agent `ebm-v1` experiments, agents demonstrated latitude-dependent corrections such as at the poles, where the RL agents increased radiative coefficients  $A$  enhancing OLR-driven cooling and alleviating persistent warm biases. In the tropics and mid-latitudes, parameter values were kept more moderate, consistent with flatter meridional temperature gradients, with warming in mid-latitudes compensating for baseline cooling biases.

Comparable interpretability is observed in the SCBC and RCE environments. In SCBC, high-performing algorithms such as DDPG and TQC converge to stable heating increments of  $-0.2$ , matching the theoretical requirement to maintain the climatological target of  $321.75$  K. In the RCE setup, agents modulate the critical lapse rate dynamically with altitude, with TQC showing the most consistent adjustments across pressure levels. These behaviours correspond to physically meaningful modifications of vertical mixing and radiative cooling, demonstrating that RL agents, when appropriately constrained, can internalise and act upon thermodynamic balances. These alignments with established physical principles enhances trust in the learnt parametrisations and supports potential integration into operational climate models.

### 3.4.4. Implications for Weather and Climate Model Parametrisations

The results presented here suggest that RL has significant potential to deliver promising improvements for weather and climate model parametrisations in the future, though confirming its practical value will require showing comparable gains in a full GCM setting (planned as the next phase of this work). Compared to traditional schemes which depend on static coefficients tuned offline through costly experiments, RL, particularly under federated and spatially decomposed regimes, offers a practical dynamic alternative in which agents adapt parameter values online as a function of the evolving model state. The emergence of geographically differentiated strategies, such as increasing the OLR parameters  $A$  and  $B$  in polar regions to counter persistent warm biases or gradually raising the critical lapse rate near the tropopause (high enough to prevent convective adjustment), demonstrates that physically interpretable parameter control can arise directly from reward-driven optimisation and federated learning which integrates naturally with MPI-based parallelisation used in many models operational at most modelling centres worldwide.

The findings also highlight the advantages of federated coordination, where frequent aggregations across distributed learning agents enable robust and scalable learning while retaining local specialisation. This is especially important for resolving spatial heterogeneity in processes such as meridional heat transport and regional radiative balances. Substantial reductions in aWRMSE and improved skill across latitude bands, most notably in the tropics and mid-latitudes, indicate that multi-agent RL can potentially not only match but surpass traditional approaches in accuracy.

Crucially, the RL-assisted parametrisations do not apply black-box corrections. Instead, they learn neural-network policy functions that set tunable parameters within existing physical parametrisations as a function of the model state, thus improving model performance while remaining embedded within the underlying physics. These policy networks are simple multilayer perceptrons (*leq* 200,000 parameters each), making them manageable for future examination with explainable AI techniques to understand how parameter choices vary across regimes. As GCMs advance toward higher resolutions and greater process complexity, the integration of interpretable, regime-aware, and online-learnt parametrisations may become a cornerstone of next-generation modelling strategies.

## 4. Conclusion

Climate models are indispensable for simulating the Earth system and making long-term climate projections. Their predictive accuracy, however, is constrained by the inability to resolve small-scale processes such as convection, radiation, and turbulent mixing which in turn contributes to the systematic biases commonly seen when climate model output is evaluated against observations. These processes are represented through parametrisations, simplified formulations with tunable parameters, which are traditionally tuned offline and remain static throughout simulations. This study explored reinforcement learning (RL) as a framework to learn functions (policies) which set these tunable parameters dynamically as a function of the model state whilst being aware of spatial regimes.

A hierarchy of idealised testbeds, spanning the heating increment simple climate bias correction model (SCBC), radiative convective equilibrium (RCE), and the zonal energy balance model (EBM), was developed to systematically evaluate RL algorithms across increasing levels of physical and spatial complexity. Experiments encompassed both single-agent setups (`scbc-v0/v1/v2`, `rce-v0/rce17-v0/v1`, `ebm-v0/v1`) and multi-agent federated setups (`ebm-v2/v3`), in which agents operated over regions and synchronised through global aggregation. Nine RL algorithms were benchmarked, with Truncated Quantile Critics (TQC) (Kuznetsov et al. 2020), Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2019), and Twin-Delayed DDPG (TD3) (Fujimoto et al. 2018) consistently achieving the highest skill and most stable convergence across configurations. Performance was assessed using area-weighted root mean squared error (areaWRMSE), model bias, temperature errors and pressure-level diagnostics (as appropriate), with results compared against a static `c1imlab` baseline with parameters fitted via linear regression.

In single-agent setups, RL agents demonstrated superior performance over static parametrisations, particularly in the tropics and mid-latitudes (for the EBM environments). TQC consistently achieved robust skill with narrow confidence intervals across SCBC, RCE, and EBM environments. In RCE, where vertical sensitivity is key, RL agents were able to vary critical lapse rates to reduce biases in temperature profile, while in SCBC, stable heating increments and temperature evolution indicated successful correction of structural drift. DDPG performed competitively across all environments, benefiting from computational simplicity and minimal tuning overhead. TD3, though effective in several experiments, showed local instability in high-variance zones such as equatorial EBM bands.

Federated reinforcement learning (FedRL) enabled regional regime-aware control by decomposing global space into zonal agents with periodic policy aggregation. In `ebm-v2/v3` with DDPG, the 6-agent configuration with frequent updates (`fed05`) achieved low zonal RMSE in tropical and mid-latitudes, surpassing both `ebm-v1` and `c1imlab`. However, in inference mode (for `ebm-v2/v3`), the globally aggregated set of weights (used in intermediate FedRL steps) introduced trade-offs, where being deployed as a single non-local policy, it diluted regime-awareness, particularly for TQC in high-gradient regions. DDPG emerged as the most robust algorithm balancing stability and generalisability. RL agents were also found to enact physically meaningful corrections, modulating radiative parameters  $A$  and  $B$  in the zonal EBM to correct for meridional biases, and correcting critical lapse rates in the RCE model to align with the climatological structure, demonstrating their capacity to learn appropriate control strategies.

The results presented here point towards an exciting future paradigm for weather and climate model parametrisations, one that is learnable and aligned with both observational and physical constraints. While this study focused on idealised heating-increment models, convection parametrised testbeds, and zonal energy balance models, the core methodology is readily extensible to more complex systems, such as the current Unified Model (UM) and the forthcoming Momentum system at the UK Met Office. Extending the approach to these more complex systems is a natural next step and is already under active development for a follow-up study. Looking ahead, integrating RL-assisted parametrisations into operational models, potentially via hybrid interfaces using SmartSim (Partee et al. 2022), FTorch (Atkinson et al. 2025) and TorchClim (Fuchs et al. 2024), offers a promising direction for future work.

In summary, this work shows that RL, when deployed in scalable and federated forms, enables the design of numerical model parametrisations that respond to local state and physical constraints, key capabilities for the next generation of skilful weather and climate models. As global modelling frameworks transition towards CMIP8, with greater emphasis on high-resolution dynamics and regional-global coupling, the demand for self-learning, flexible parametrisation schemes is set to grow. This study presents a promising prototype for such systems, where RL agents dynamically calibrate model behaviour while implicitly learning ML-based components of existing parametrisation schemes without compromising their physical integrity.

**Code and Data Availability.** The code for this project and its documentation are available in our GitHub repositories (<https://github.com/p3jnitnath/climate-rl> and <https://github.com/p3jnitnath/climate-rl-fedRL>). A lightweight API for future extensions of this work is developed and made available at <https://github.com/p3jnitnath/climate-rl-fedrlain-api>. Data for all experiment runs are available in our publicly available Zenodo repository (<https://doi.org/10.5281/zenodo.17116349>).

The software for this project was developed using Python (Van Rossum 2007) on VSCode (<https://code.visualstudio.com>) and Jupyter Notebooks (Kluyver et al. 2016) (<https://jupyter.org>). A number of Python packages have been used including:

- climlab (Rose 2018) (<https://climlab.readthedocs.io/en/latest>)
- gymnasium (<https://gymnasium.farama.org/index.html>)
- matplotlib (Hunter and Dale 2007) (<https://matplotlib.org>)
- numpy (Oliphant 2006) (<https://numpy.org>)
- optuna (Akiba et al. 2019) (<https://optuna.org>)
- pandas (Reback et al. 2020) (<https://pandas.pydata.org>)
- ray (Moritz et al. 2018) (<https://docs.ray.io/en/latest/index.html>)
- ray tune (Liaw et al. 2018) (<https://docs.ray.io/en/latest/tune/index.html>)
- stable\_baselines3 (Raffin et al. 2021) (<https://pypi.org/project/stable-baselines3>)
- tephilo (<https://tephilo.readthedocs.io/en/latest/index.html>)
- torch (Paszke et al. 2019) (<https://pytorch.org>)
- tyro (<https://brentyi.github.io/tyro>)
- xarray (Hoyer and Hamman 2017) (<https://docs.xarray.dev/en/stable>)
- smartsim (Partee et al. 2022) (<https://www.craylabs.org/docs/overview.html>)
- flower (Beutel et al. 2022) (<https://flower.ai>)

Code for RL algorithms (DDPG, TD3, PPO, SAC, TQC) were adapted from the cleanRL (S Huang et al. 2022) (<https://github.com/vwxyzjn/cleanrl/tree/master/cleanrl>) project repository. TRPO was adapted in the cleanRL style by Yuhua Jiang (<https://github.com/Jackory>). DPG, REINFORCE and AVG were adapted in the cleanRL style by Pritthijit Nath. This manuscript was prepared using L<sup>A</sup>T<sub>E</sub>X (<https://www.latex-project.org>) on Overleaf (<https://www.overleaf.com>).

**LLM Usage.** The authors acknowledge the use of AI language models, specifically ChatGPT (GPT-5.2 and GPT-4o <https://chatgpt.com>), during the preparation of this work. These tools were used to polish language usage and improve the overall clarity of the manuscript, as well as to assist with designing plotting code for the graphs. All AI-generated content was reviewed, verified, and edited by the authors to ensure accuracy and appropriateness.

**Acknowledgments.** P. Nath was supported by the UKRI Centre for Doctoral Training in Application of Artificial Intelligence to the study of Environmental Risks [EP/S022961/1] ([https://ai4er-cdt\\_esc.cam.ac.uk](https://ai4er-cdt_esc.cam.ac.uk)). Mark Webb was supported by the Met Office Hadley Centre Climate Programme funded by DSIT. This work used JASMIN, the UK's collaborative data analysis environment (<https://www.jasmin.ac.uk>) managed by UKRI NERC and STFC. We thank Andrew Shao and Alessandro Rigazzi (HPE) for their valuable assistance with setting up SmartSim on JASMIN. The authors declare that they have no conflict of interest.

## References

- Manabe S and Wetherald RT** (1967) Thermal Equilibrium of the Atmosphere with a Given Distribution of Relative Humidity. en, 27 May **24**(3), Section: Journal of the Atmospheric Sciences, 241–259. issn: 1520-0469. [https://journals.ametsoc.org/view/journals/atsc/24/3/1520-0469\\_1967\\_024\\_0241\\_teotaw\\_2\\_0\\_co\\_2.xml](https://journals.ametsoc.org/view/journals/atsc/24/3/1520-0469_1967_024_0241_teotaw_2_0_co_2.xml) (accessed 27 May 2025).
- Budyko MI** (1969) The effect of solar radiation variations on the climate of the Earth. en. *Tellus*, 6 April **21**(5), 611–619. issn: 2153-3490. doi: [10.1111/j.2153-3490.1969.tb00466.x](https://doi.org/10.1111/j.2153-3490.1969.tb00466.x). <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.2153-3490.1969.tb00466.x> (accessed 6 April 2025).
- Sellers WD** (1969) A Global Climatic Model Based on the Energy Balance of the Earth-Atmosphere System. en, 6 April **8**(3), Section: Journal of Applied Meteorology and Climatology, 392–400. issn: 1520-0450. [https://journals.ametsoc.org/view/journals/apme/8/3/1520-0450\\_1969\\_008\\_0392\\_agcmbo\\_2\\_0\\_co\\_2.xml](https://journals.ametsoc.org/view/journals/apme/8/3/1520-0450_1969_008_0392_agcmbo_2_0_co_2.xml) (accessed 6 April 2025).
- North GR** (1975) Theory of Energy-Balance Climate Models. en, 6 April **32**(11), Section: Journal of the Atmospheric Sciences, 2033–2043. issn: 1520-0469. [https://journals.ametsoc.org/view/journals/atsc/32/11/1520-0469\\_1975\\_032\\_2033\\_toebcm\\_2\\_0\\_co\\_2.xml](https://journals.ametsoc.org/view/journals/atsc/32/11/1520-0469_1975_032_2033_toebcm_2_0_co_2.xml) (accessed 6 April 2025).
- Watkins CJCH and Dayan P** (1992) Q-learning. en. *Machine Learning*, 21 June **8**(3), 279–292. issn: 1573-0565. doi: [10.1007/BF00992698](https://doi.org/10.1007/BF00992698). <https://doi.org/10.1007/BF00992698> (accessed 21 June 2025).
- Williams RJ** (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. en. *Machine Learning*, 27 June **8**(3), 229–256. issn: 1573-0565. doi: [10.1007/BF00992696](https://doi.org/10.1007/BF00992696). <https://doi.org/10.1007/BF00992696> (accessed 27 June 2025).
- Sutton RS and Barto AG** (1998) *Reinforcement learning: An introduction*. vol 1. 1. MIT Press Cambridge. Available at <http://incompleteideas.net/book/RLbook2020.pdf> (accessed 22 March 2025).

- Kalnay E** (2002) *Atmospheric Modeling, Data Assimilation and Predictability*. en. ISBN: 9780511802270 Publisher: Cambridge University Press. doi: [10.1017/CBO9780511802270](https://doi.org/10.1017/CBO9780511802270). Available at <https://www.cambridge.org/highereducation/books/atmospheric-modeling-data-assimilation-and-predictability/C5FD207439132836E85027754CE9BC1A> (accessed 27 May 2025).
- Clough SA, Shephard MW, Mlawer EJ, Delamere JS, Iacono MJ, Cady-Pereira K, Boukabara S and Brown PD** (2005) Atmospheric radiative transfer modeling: a summary of the AER codes. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 27 June **91**(2), 233–244. ISSN: 0022-4073. doi: [10.1016/j.jqsrt.2004.05.058](https://doi.org/10.1016/j.jqsrt.2004.05.058). <https://www.sciencedirect.com/science/article/pii/S0022407304002158> (accessed 27 June 2025).
- Oliphant TE** (2006) *Guide to numpy*. vol 1. Trelgol Publishing USA. Available at <https://ecs.wgtn.ac.nz/foswiki/pub/Support/ManualPagesAndDocumentation/numpybook.pdf> (accessed 20 June 2024).
- Soden BJ and Held IM** (2006) An assessment of climate feedbacks in coupled ocean–atmosphere models. *Journal of climate*, 29 May **19**(14), 3354–3360. <https://doi.org/10.1175/JCLI3799.1>.xml (accessed 29 May 2025).
- Hunter J and Dale D** (2007) The matplotlib user's guide. *Matplotlib 0.90.0 user's guide*, 20 June. [https://www.jick.net/Manuals/Python/matplotlib-users\\_guide\\_0.90.0.pdf](https://www.jick.net/Manuals/Python/matplotlib-users_guide_0.90.0.pdf) (accessed 20 June 2024).
- Randall DA, Wood RA, Bony S, Fichefet T, Fyfe J, Kattsov V, Pitman A, Shukla J and Srinivasan J** (2007) Climate models and their evaluation. In *Climate change 2007: The physical science basis. Contribution of Working Group I to the Fourth Assessment Report of the IPCC (FAR)*. Cambridge University Press, 589–662. Available at [https://pure.mpg.de/rest/items/item\\_1765216/component/file\\_1765214/content](https://pure.mpg.de/rest/items/item_1765216/component/file_1765214/content) (accessed 21 March 2025).
- Stensrud DJ** (2007) *Parameterization Schemes: Keys to Understanding Numerical Weather Prediction Models*. Cambridge: Cambridge University Press. ISBN: 978-0-521-12676-2. doi: [10.1017/CBO9780511812590](https://doi.org/10.1017/CBO9780511812590). Available at <https://www.cambridge.org/core/books/parameterization-schemes/C7C8EC8901957314433BE7C8BC36F16D> (accessed 27 May 2025).
- Van Rossum G** (2007) Python programming language. In *USENIX annual technical conference*. vol 41. Issue: 1. Santa Clara, CA, 1–36. Available at [https://alalqab.com/en/Python\\_\(programming\\_language\)](https://alalqab.com/en/Python_(programming_language)) (accessed 25 June 2025).
- Iacono MJ, Delamere JS, Mlawer EJ, Shephard MW, Clough SA and Collins WD** (2008) Radiative forcing by long-lived greenhouse gases: Calculations with the AER radiative transfer models. en. *Journal of Geophysical Research: Atmospheres*, 27 June **113**(D13). ISSN: 2156-2202. doi: [10.1029/2008JD009944](https://doi.org/10.1029/2008JD009944). <https://onlinelibrary.wiley.com/doi/abs/10.1029/2008JD009944> (accessed 27 June 2025).
- Sanfilippo S and Noordhuis P** (2009) *Redis: In-memory Data Structure Server*. Available at <https://redis.%20io>.
- Brown A, Milton S, Cullen M, Golding B, Mitchell J and Shelly A** (2012) Unified Modeling and Prediction of Weather and Climate: A 25-Year Journey. en, 21 March **93**(12), Section: Bulletin of the American Meteorological Society, 1865–1877. doi: [10.1175/BAMS-D-12-00018.1](https://doi.org/10.1175/BAMS-D-12-00018.1). [https://journals.ametsoc.org/view/journals/bams/93/12/bams-d-12-00018.1.xml](https://doi.org/10.1175/BAMS-D-12-00018.1) (accessed 21 March 2025).
- Lawrence BN, Bennett V, Churchill J, Juckes M, Kershaw P, Oliver P, Pritchard M and Stephens A** (2012) *The JASMIN super-data-cluster*. arXiv:1204.3553 [physics]. doi: [10.48550/arXiv.1204.3553](https://doi.org/10.48550/arXiv.1204.3553). Available at [http://arxiv.org/abs/1204.3553](https://arxiv.org/abs/1204.3553) (accessed 12 June 2024).
- Mauritsen T, Stevens B, Roeckner E, Crueger T, Esch M, Giorgetta M, Haak H, Jungclaus J, Klocke D, Matei D, Mikolajewicz U, Notz D, Pincus R, Schmidt H and Tomassini L** (2012) Tuning the climate of a global model. en. *Journal of Advances in Modeling Earth Systems*, 26 August **4**(3). ISSN: 1942-2466. doi: [10.1029/2012MS000154](https://doi.org/10.1029/2012MS000154). <https://doi.org/10.1029/2012MS000154> (accessed 26 August 2025).
- Hallberg R** (2013) Using a resolution function to regulate parameterizations of oceanic mesoscale eddy effects. *Ocean Modelling*, 29 May **72**, Publisher: Elsevier, 92–103. [https://www.sciencedirect.com/science/article/pii/S1463500313001601?casa\\_token=O0JcLnENcokAAAAA:cfamxO61tFhbZ2MrPTGSy7lwWlfgKKv1L-a2lomgyGOINx7NIC3g3RvDI\\_xFsy4fRq5yCo](https://doi.org/10.1016/j.ocemod.2013.05.001) (accessed 29 May 2025).
- Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D and Riedmiller M** (2013) *Playing Atari with Deep Reinforcement Learning*. arXiv:1312.5602 [cs]. doi: [10.48550/arXiv.1312.5602](https://doi.org/10.48550/arXiv.1312.5602). Available at [http://arxiv.org/abs/1312.5602](https://arxiv.org/abs/1312.5602) (accessed 28 November 2025).
- Vial J, Dufresne JL and Bony S** (2013) On the interpretation of inter-model spread in CMIP5 climate sensitivity estimates. en. *Climate Dynamics*, 29 May **41**(11), 3339–3362. ISSN: 1432-0894. doi: [10.1007/s00382-013-1725-9](https://doi.org/10.1007/s00382-013-1725-9). <https://doi.org/10.1007/s00382-013-1725-9> (accessed 29 May 2025).
- Flato G, Marotzke J, Abiodun B, Braconnot P, Chou SC, Collins W, Cox P, Driouech F, Emori S and Eyring V** (2014) Evaluation of climate models. In *Climate change 2013: the physical science basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 741–866. Available at [https://pure.mpg.de/pubman/faces/ViewItemOverviewPage.jsp?itemId=item\\_1977534](https://pure.mpg.de/pubman/faces/ViewItemOverviewPage.jsp?itemId=item_1977534) (accessed 30 May 2025).
- Silver D, Lever G, Heess N, Degris T, Wierstra D and Riedmiller M** (2014) Deterministic Policy Gradient Algorithms. en. In *Proceedings of the 31st International Conference on Machine Learning*. ISSN: 1938-7228. PMLR, 387–395. Available at <https://proceedings.mlr.press/v32/silver14.html> (accessed 21 June 2025).
- Schulman J, Levine S, Abbeel P, Jordan M and Moritz P** (2015) Trust Region Policy Optimization. en. In *Proceedings of the 32nd International Conference on Machine Learning*. ISSN: 1938-7228. PMLR, 1889–1897. Available at <https://proceedings.mlr.press/v37/schulman15.html> (accessed 27 June 2025).

- Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J and Zaremba W** (2016) *OpenAI Gym*. arXiv:1606.01540 [cs]. doi: [10.48550/arXiv.1606.01540](https://doi.org/10.48550/arXiv.1606.01540). Available at <http://arxiv.org/abs/1606.01540> (accessed 27 June 2025).
- Gross M, Malardel S, Jablonowski C and Wood N** (2016) Bridging the (Knowledge) Gap between Physics and Dynamics. en, 25 June **97**(1), Section: Bulletin of the American Meteorological Society, 137–142. doi: [10.1175/BAMS-D-15-00103.1](https://doi.org/10.1175/BAMS-D-15-00103.1). <https://journals.ametsoc.org/view/journals/bams/97/1/bams-d-15-00103.1.xml> (accessed 25 June 2025).
- Hartmann DL** (2016) Climate Sensitivity and Feedback Mechanisms. In Hartmann DL (ed), *Global Physical Climatology (Second Edition)*. Boston: Elsevier, 293–323. ISBN: 978-0-12-328531-7. doi: [10.1016/B978-0-12-328531-7.00010-4](https://doi.org/10.1016/B978-0-12-328531-7.00010-4). Available at <https://www.sciencedirect.com/science/article/pii/B9780123285317000104> (accessed 27 May 2025).
- Kluver T, Ragan-Kelley B, Rez F, Granger B, Bussonnier M, Frederic J, Kelley K, Hamrick J, Grout J, Corlay S, Ivanov P, Avila D, n, Abdalla S, Willing C and Team JD** (2016) Jupyter Notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, 87–90. doi: [10.3233/978-1-61499-649-1-87](https://doi.org/10.3233/978-1-61499-649-1-87). Available at <https://ebooks.iospress.nl/doi/10.3233/978-1-61499-649-1-87> (accessed 20 June 2024).
- Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Driessche G van den, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T and Hassabis D** (2016) Mastering the game of Go with deep neural networks and tree search. en. *Nature*, 22 March **529**(7587), Publisher: Nature Publishing Group, 484–489. ISSN: 1476-4687. doi: [10.1038/nature16961](https://doi.org/10.1038/nature16961). <https://www.nature.com/articles/nature16961> (accessed 22 March 2025).
- Ceppi P, Brient F, Zelinka MD and Hartmann DL** (2017) Cloud feedback mechanisms and their representation in global climate models. en. *WIREs Climate Change*, 26 August **8**(4), e465. ISSN: 1757-7799. doi: [10.1002/wcc.465](https://doi.org/10.1002/wcc.465). <https://onlinelibrary.wiley.com/doi/abs/10.1002/wcc.465> (accessed 26 August 2025).
- Hourdin F, Mauritzen T, Gettelman A, Golaz JC, Balaji V, Duan Q, Folini D, Ji D, Klocke D, Qian Y, Rauser F, Rio C, Tomassini L, Watanabe M and Williamson D** (2017) The Art and Science of Climate Model Tuning. en, 10 August **98**(3), Section: Bulletin of the American Meteorological Society, 589–602. doi: [10.1175/BAMS-D-15-00135.1](https://doi.org/10.1175/BAMS-D-15-00135.1). <https://journals.ametsoc.org/view/journals/bams/98/3/bams-d-15-00135.1.xml> (accessed 10 August 2025).
- Hoyer S and Hamman J** (2017) xarray: ND labeled arrays and datasets in Python. *Journal of Open Research Software*, 20 June **5**(1), 10–10. <https://account.openresearchsoftware.metajnl.com/index.php/up-j-jors/article/view/jors.148> (accessed 20 June 2024).
- Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O** (2017) *Proximal Policy Optimization Algorithms*. arXiv:1707.06347 [cs]. doi: [10.48550/arXiv.1707.06347](https://doi.org/10.48550/arXiv.1707.06347). Available at [http://arxiv.org/abs/1707.06347](https://arxiv.org/abs/1707.06347) (accessed 21 June 2025).
- Webb MJ, Andrews T, Bodas-Salcedo A, Bony S, Bretherton CS, Chadwick R, Chepfer H, Douville H, Good P, Kay JE, Klein SA, Marchand R, Medeiros B, Siebesma AP, Skinner CB, Stevens B, Tselioudis G, Tushimura Y and Watanabe M** (2017) The Cloud Feedback Model Intercomparison Project (CFMIP) contribution to CMIP6. English. *Geoscientific Model Development*, 22 March **10**(1), Publisher: Copernicus GmbH, 359–384. ISSN: 1991-959X. doi: [10.5194/gmd-10-359-2017](https://doi.org/10.5194/gmd-10-359-2017). <https://gmd.copernicus.org/articles/10/359/2017/> (accessed 22 March 2025).
- Baumgart M, Riemer M, Wirth V, Teubler F and Lang STK** (2018) Potential Vorticity Dynamics of Forecast Errors: A Quantitative Case Study. en, 30 September **146**(5), Section: Monthly Weather Review, 1405–1425. doi: [10.1175/MWR-D-17-0196.1](https://doi.org/10.1175/MWR-D-17-0196.1). <https://journals.ametsoc.org/view/journals/mwre/146/5/mwr-d-17-0196.1.xml> (accessed 30 September 2025).
- Fujimoto S, Hoof Hv and Meger D** (2018) *Addressing Function Approximation Error in Actor-Critic Methods*. arXiv:1802.09477 [cs]. doi: [10.48550/arXiv.1802.09477](https://doi.org/10.48550/arXiv.1802.09477). Available at [http://arxiv.org/abs/1802.09477](https://arxiv.org/abs/1802.09477) (accessed 21 June 2025).
- Haarnoja T, Zhou A, Abbeel P and Levine S** (2018) *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. arXiv:1801.01290 [cs]. doi: [10.48550/arXiv.1801.01290](https://doi.org/10.48550/arXiv.1801.01290). Available at [http://arxiv.org/abs/1801.01290](https://arxiv.org/abs/1801.01290) (accessed 21 June 2025).
- Liaw R, Liang E, Nishihara R, Moritz P, Gonzalez JE and Stoica I** (2018) *Tune: A Research Platform for Distributed Model Selection and Training*. arXiv:1807.05118 [cs, stat]. doi: [10.48550/arXiv.1807.05118](https://doi.org/10.48550/arXiv.1807.05118). Available at [http://arxiv.org/abs/1807.05118](https://arxiv.org/abs/1807.05118) (accessed 26 June 2024).
- Moritz P, Nishihara R, Wang S, Tumanov A, Liaw R, Liang E, Elibol M, Yang Z, Paul W and Jordan MI** (2018) Ray: A distributed framework for emerging AI applications. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, 561–577. Available at <https://www.usenix.org/conference/osdi18/presentation/moritz> (accessed 12 June 2024).
- Rose BE** (2018) CLIMLAB: a Python toolkit for interactive, process-oriented climate modeling. *J. Open Source Softw.*, 8 June **3**(24), 659. <https://www.theoj.org/joss-papers/joss.00659/10.21105.joss.00659.pdf> (accessed 8 June 2024).
- Akiba T, Sano S, Yanase T, Ohta T and Koyama M** (2019) Optuna: A Next-generation Hyperparameter Optimization Framework. en. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA: ACM, 2623–2631. ISBN: 978-1-4503-6201-6. doi: [10.1145/3292500.3330701](https://doi.org/10.1145/3292500.3330701). Available at <https://dl.acm.org/doi/10.1145/3292500.3330701> (accessed 12 June 2024).
- Kovachki NB and Stuart AM** (2019) Ensemble Kalman inversion: a derivative-free technique for machine learning tasks. en. *Inverse Problems*, 20 December **35**(9), Publisher: IOP Publishing, 095005. ISSN: 0266-5611. doi: [10.1088/1361-6420/ab1c3a](https://doi.org/10.1088/1361-6420/ab1c3a). <https://doi.org/10.1088/1361-6420/ab1c3a> (accessed 20 December 2025).

- Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D and Wierstra D** (2019) *Continuous control with deep reinforcement learning*. arXiv:1509.02971 [cs]. doi: [10.48550/arXiv.1509.02971](https://doi.org/10.48550/arXiv.1509.02971). Available at <http://arxiv.org/abs/1509.02971> (accessed 22 March 2025).
- Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J and Chintala S** (2019) *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. arXiv:1912.01703 [cs]. doi: [10.48550/arXiv.1912.01703](https://doi.org/10.48550/arXiv.1912.01703). Available at <http://arxiv.org/abs/1912.01703> (accessed 25 June 2025).
- Stevens B, Satoh M, Auger L, Biercamp J, Bretherton CS, Chen X, Düben P, Judt F, Khairoutdinov M, Klocke D, Kodama C, Kornblueh L, Lin SJ, Neumann P, Putman WM, Röber N, Shibuya R, Vanniere B, Vidale PL, Wedi N and Zhou L** (2019) DYAMOND: the DYnamics of the Atmospheric general circulation Modeled On Non-hydrostatic Domains. en. *Progress in Earth and Planetary Science*, 30 May **6**(1), 61. issn: 2197-4284. doi: [10.1186/s40645-019-0304-z](https://doi.org/10.1186/s40645-019-0304-z) (accessed 30 May 2025).
- Brenowitz ND, Henn B, McGibbon J, Clark SK, Kwa A, Perkins WA, Watt-Meyer O and Bretherton CS** (2020) *Machine Learning Climate Model Dynamics: Offline versus Online Performance*. arXiv:2011.03081 [physics]. doi: [10.48550/arXiv.2011.03081](https://doi.org/10.48550/arXiv.2011.03081). Available at <http://arxiv.org/abs/2011.03081> (accessed 15 June 2025).
- Kuznetsov A, Shvechikov P, Grishin A and Vetrov D** (2020) Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics. en. In *Proceedings of the 37th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, 5556–5566. Available at <https://proceedings.mlr.press/v119/kuznetsov20a.html> (accessed 27 June 2025).
- Lattimore T and Szepesvári C** (2020) *Bandit Algorithms*. Cambridge: Cambridge University Press. ISBN: 978-1-108-48682-8. doi: [10.1017/9781108571401](https://doi.org/10.1017/9781108571401). Available at <https://www.cambridge.org/core/books/bandit-algorithms/8E39FD004E6CE036680F90DD0C6F09FC> (accessed 22 March 2025).
- Reback J, McKinney W, Van Den Bossche J, Augspurger T, Cloud P, Klein A, Hawkins S, Roeschke M, Tratner J and She C** (2020) pandas-dev/pandas: Pandas 1.0. 5. Zenodo, 20 June. <https://ui.adsabs.harvard.edu/abs/2020znndo...3898987R/abstract> (accessed 20 June 2024).
- Schriftwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T, Lillicrap T and Silver D** (2020) Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, 23 March **588**(7839), arXiv:1911.08265 [cs], 604–609. issn: 0028-0836, 1476-4687. doi: [10.1038/s41586-020-03051-4](https://doi.org/10.1038/s41586-020-03051-4). [http://arxiv.org/abs/1911.08265](https://arxiv.org/abs/1911.08265) (accessed 23 March 2025).
- Dunbar ORA, Garbuno-Íñigo A, Schneider T and Stuart AM** (2021) Calibration and Uncertainty Quantification of Convective Parameters in an Idealized GCM. en. *Journal of Advances in Modeling Earth Systems*, 22 June **13**(9), e2020MS002454. issn: 1942-2466. doi: [10.1029/2020MS002454](https://doi.org/10.1029/2020MS002454). <https://onlinelibrary.wiley.com/doi/abs/10.1029/2020MS002454> (accessed 22 June 2025).
- Kiran BR, Sobh I, Talpaert V, Mannion P, Sallab AAA, Yogamani S and Pérez P** (2021) *Deep Reinforcement Learning for Autonomous Driving: A Survey*. arXiv:2002.00444 [cs]. doi: [10.48550/arXiv.2002.00444](https://doi.org/10.48550/arXiv.2002.00444). Available at <http://arxiv.org/abs/2002.00444> (accessed 23 March 2025).
- Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M and Dormann N** (2021) Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 20 June **22**(268), 1–8. <https://www.jmlr.org/papers/v22/20-1364.html> (accessed 20 June 2024).
- Bae HJ and Koumoutsakos P** (2022) Scientific multi-agent reinforcement learning for wall-models of turbulent flows. en. *Nature Communications*, 25 December **13**(1), Publisher: Nature Publishing Group, 1443. issn: 2041-1723. doi: [10.1038/s41467-022-28957-7](https://doi.org/10.1038/s41467-022-28957-7). <https://www.nature.com/articles/s41467-022-28957-7> (accessed 25 December 2025).
- Beutel DJ, Topal T, Mathur A, Qiu X, Fernandez-Marques J, Gao Y, Sani L, Li KH, Parcollet T, Gusmão PPBd and Lane ND** (2022) Flower: A Friendly Federated Learning Research Framework. arXiv:2007.14390 [cs]. doi: [10.48550/arXiv.2007.14390](https://doi.org/10.48550/arXiv.2007.14390). Available at <http://arxiv.org/abs/2007.14390> (accessed 6 April 2025).
- Degrave J, Felici F, Buchli J, Neunert M, Tracey B, Carpanese F, Ewalds T, Hafner R, Abdolmaleki A, Casas D de las, Donner C, Fritz L, Galperti C, Huber A, Keeling J, Tsipoukelli M, Kay J, Merle A, Moret JM, Noury S, Pesamosca F, Pfau D, Sauter O, Sommariva C, Coda S, Duval B, Fasoli A, Kohli P, Kavukcuoglu K, Hassabis D and Riedmiller M** (2022) Magnetic control of tokamak plasmas through deep reinforcement learning. en. *Nature*, 17 August **602**(7897), Publisher: Nature Publishing Group, 414–419. issn: 1476-4687. doi: [10.1038/s41586-021-04301-9](https://doi.org/10.1038/s41586-021-04301-9). <https://www.nature.com/articles/s41586-021-04301-9> (accessed 17 August 2025).
- Howland MF, Dunbar ORA and Schneider T** (2022) Parameter Uncertainty Quantification in an Idealized GCM With a Seasonal Cycle. en. *Journal of Advances in Modeling Earth Systems*, 22 June **14**(3), e2021MS002735. issn: 1942-2466. doi: [10.1029/2021MS002735](https://doi.org/10.1029/2021MS002735). <https://onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002735> (accessed 22 June 2025).
- Huang S, Dossa RFJ, Ye C, Braga J, Chakraborty D, Mehta K and Araújo JG** (2022) Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 20 June **23**(274), 1–18. <https://www.jmlr.org/papers/v23/21-1342.html> (accessed 20 June 2024).
- Jin H, Peng Y, Yang W, Wang S and Zhang Z** (2022) Federated Reinforcement Learning with Environment Heterogeneity. en. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. ISSN: 2640-3498. PMLR, 18–37. Available at <https://proceedings.mlr.press/v151/jin22a.html> (accessed 18 August 2025).

- Lojko A, Payne A and Jablonowski C** (2022) The Remote Role of North-American Mesoscale Convective Systems on the Forecast of a Rossby Wave Packet: A Multi-Model Ensemble Case-Study. en, 30 September **127**(24). doi: [10.1029/2022JD037171](https://doi.org/10.1029/2022JD037171). <https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2022JD037171> (accessed 30 September 2025).
- Ouyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J, Hilton J, Kelton F, Miller L, Simens M, Askell A, Welinder P, Christiano P, Leike J and Lowe R** (2022) *Training language models to follow instructions with human feedback*. arXiv:2203.02155 [cs]. doi: [10.48550/arXiv.2203.02155](https://doi.org/10.48550/arXiv.2203.02155). Available at [http://arxiv.org/abs/2203.02155](https://arxiv.org/abs/2203.02155) (accessed 31 May 2025).
- Partee S, Ellis M, Rigazzi A, Shao AE, Bachman S, Marques G and Robbins B** (2022) Using Machine Learning at scale in numerical simulations with SmartSim: An application to ocean climate modeling. *Journal of Computational Science*, 6 April **62**, 101707. issn: 1877-7503. doi: [10.1016/j.jocs.2022.101707](https://doi.org/10.1016/j.jocs.2022.101707). <https://www.sciencedirect.com/science/article/pii/S1877750322001065> (accessed 6 April 2025).
- Lguensat R, Deshayes J, Durand H and Balaji V** (2023) Semi-Automatic Tuning of Coupled Climate Models With Multiple Intrinsic Timescales: Lessons Learned From the Lorenz96 Model. en. *Journal of Advances in Modeling Earth Systems*, 14 August **15**(5), e2022MS003367. issn: 1942-2466. doi: [10.1029/2022MS003367](https://doi.org/10.1029/2022MS003367). <https://onlinelibrary.wiley.com/doi/abs/10.1029/2022MS003367> (accessed 14 August 2025).
- McMahan HB, Moore E, Ramage D, Hampson S and Arcas Bay** (2023) *Communication-Efficient Learning of Deep Networks from Decentralized Data*. arXiv:1602.05629 [cs]. doi: [10.48550/arXiv.1602.05629](https://doi.org/10.48550/arXiv.1602.05629). Available at [http://arxiv.org/abs/1602.05629](https://arxiv.org/abs/1602.05629) (accessed 25 June 2025).
- Watanabe S** (2023) *Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance*. arXiv:2304.11127 [cs]. doi: [10.48550/arXiv.2304.11127](https://doi.org/10.48550/arXiv.2304.11127). Available at [http://arxiv.org/abs/2304.11127](https://arxiv.org/abs/2304.11127) (accessed 30 July 2025).
- Christopoulos C, Lopez-Gomez I, Beucler T, Cohen Y, Kawczynski C, Dunbar ORA and Schneider T** (2024) Online Learning of Entrainment Closures in a Hybrid Machine Learning Parameterization. en. *Journal of Advances in Modeling Earth Systems*, 20 December **16**(11), e2024MS004485. issn: 1942-2466. doi: [10.1029/2024MS004485](https://doi.org/10.1029/2024MS004485). <https://onlinelibrary.wiley.com/doi/abs/10.1029/2024MS004485> (accessed 20 December 2025).
- Fuchs D, Sherwood SC, Prasad A, Trapeznikov K and Gimlett J** (2024) TorchClim v1.0: a deep-learning plugin for climate model physics. English. *Geoscientific Model Development*, 14 August **17**(14), Publisher: Copernicus GmbH, 5459–5475. issn: 1991-959X. doi: [10.5194/gmd-17-5459-2024](https://doi.org/10.5194/gmd-17-5459-2024). <https://gmd.copernicus.org/articles/17/5459/2024/> (accessed 14 August 2025).
- Patterson A, Neumann S, White M and White A** (2024) Empirical Design in Reinforcement Learning. *Journal of Machine Learning Research*, 20 December **25**(318), 1–63. issn: 1533-7928. <http://jmlr.org/papers/v25/23-0183.html> (accessed 20 December 2025).
- Schneider T, Leung LR and Wills RCJ** (2024) Opinion: Optimizing climate models with process knowledge, resolution, and artificial intelligence. English. *Atmospheric Chemistry and Physics*, 22 March **24**(12), Publisher: Copernicus GmbH, 7041–7062. issn: 1680-7316. doi: [10.5194/acp-24-7041-2024](https://doi.org/10.5194/acp-24-7041-2024). <https://acp.copernicus.org/articles/24/7041/2024/> (accessed 22 March 2025).
- Towers M, Kwiatkowski A, Terry J, Balis JU, Cola GD, Deleu T, Goulão M, Kallinteris A, Krimmel M, KG A, Perez-Vicente R, Pierré A, Schulhoff S, Tai JJ, Tan H and Younis OG** (2024) *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. arXiv:2407.17032 [cs]. doi: [10.48550/arXiv.2407.17032](https://doi.org/10.48550/arXiv.2407.17032). Available at [http://arxiv.org/abs/2407.17032](https://arxiv.org/abs/2407.17032) (accessed 27 June 2025).
- Vasan G, Elsayed M, Azimi A, He J, Sharif F, Bellinger C, White M and Mahmood AR** (2024) *Deep Policy Gradient Methods Without Batch Updates, Target Networks, or Replay Buffers*. arXiv:2411.15370 [cs]. doi: [10.48550/arXiv.2411.15370](https://doi.org/10.48550/arXiv.2411.15370). Available at [http://arxiv.org/abs/2411.15370](https://arxiv.org/abs/2411.15370) (accessed 6 April 2025).
- Atkinson J, Elafrou A, Kasoar E, Wallwork JG, Meltzer T, Clifford S, Orchard D and Edsall C** (2025) FTorch: a library for coupling PyTorch models to Fortran. en. *Journal of Open Source Software*, 29 May **10**(107), 7602. issn: 2475-9066. doi: [10.21105/joss.07602](https://doi.org/10.21105/joss.07602). <https://joss.theoj.org/papers/10.21105/joss.07602> (accessed 29 May 2025).
- Bertoli G, Mohebi S, Ozdemir F, Jucker J, Rüdisühl S, Perez-Cruz F, Salzmann M and Schemm S** (2025) Revisiting Machine Learning Approaches for Short- and Longwave Radiation Inference in Weather and Climate Models. en. *Journal of Advances in Modeling Earth Systems*, 2 October **17**(9), e2025MS004956. issn: 1942-2466. doi: [10.1029/2025MS004956](https://doi.org/10.1029/2025MS004956). <https://onlinelibrary.wiley.com/doi/abs/10.1029/2025MS004956> (accessed 2 October 2025).
- Bonnet P, Pastorí L, Schwabe M, Giorgetta M, Iglesias-Suarez F and Eyring V** (2025) Tuning the ICON-A 2.6.4 climate model with machine-learning-based emulators and history matching. English. *Geoscientific Model Development*, 13 August **18**(12), Publisher: Copernicus GmbH, 3681–3706. issn: 1991-959X. doi: [10.5194/gmd-18-3681-2025](https://doi.org/10.5194/gmd-18-3681-2025). <https://gmd.copernicus.org/articles/18/3681/2025/> (accessed 13 August 2025).
- Chen J, Zhang M, Zhang T, Lin W and Xue W** (2025) Stable Simulation of the Community Atmosphere Model Using Machine-Learning Physical Parameterization Trained With Experience Replay. en. *Journal of Advances in Modeling Earth Systems*, 26 August **17**(6), e2024MS004722. issn: 1942-2466. doi: [10.1029/2024MS004722](https://doi.org/10.1029/2024MS004722). <https://onlinelibrary.wiley.com/doi/abs/10.1029/2024MS004722> (accessed 26 August 2025).
- Font B, Alcántara-Ávila F, Rabault J, Vinuesa R and Lehmkuhl O** (2025) Deep reinforcement learning for active flow control in a turbulent separation bubble. en. *Nature Communications*, 23 March **16**(1), Publisher: Nature Publishing Group, 1422. issn:

- 2041-1723. doi: [10.1038/s41467-025-56408-6](https://doi.org/10.1038/s41467-025-56408-6). <https://www.nature.com/articles/s41467-025-56408-6> (accessed 23 March 2025).
- Guo D et al.** (2025) DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. en. *Nature*, 28 November **645**(8081), Publisher: Nature Publishing Group, 633–638. ISSN: 1476-4687. doi: [10.1038/s41586-025-09422-z](https://doi.org/10.1038/s41586-025-09422-z). <https://www.nature.com/articles/s41586-025-09422-z> (accessed 28 November 2025).
- Ikhtiarudin A, Das A, Thakkar P and Kundu A** (2025) *BenchRL-QAS: Benchmarking reinforcement learning algorithms for quantum architecture search*. arXiv:2507.12189 [quant-ph]. doi: [10.48550/arXiv.2507.12189](https://doi.org/10.48550/arXiv.2507.12189). Available at <http://arxiv.org/abs/2507.12189> (accessed 28 July 2025).
- Lin J, Yu S, Peng L, Beucler T, Wong-Toi E, Hu Z, Gentine P, Geleta M and Pritchard M** (2025) Navigating the Noise: Bringing Clarity to ML Parameterization Design With  $\mathcal{O}(100)$  Ensembles. en. *Journal of Advances in Modeling Earth Systems*, 14 August **17**(4), e2024MS004551. ISSN: 1942-2466. doi: [10.1029/2024MS004551](https://doi.org/10.1029/2024MS004551). <https://onlinelibrary.wiley.com/doi/abs/10.1029/2024MS004551> (accessed 14 August 2025).
- Mole A, Weissenbacher M, Rigas G and Laizet S** (2025) *Reinforcement Learning Increases Wind Farm Power Production by Enabling Closed-Loop Collaborative Control*. arXiv:2506.20554 [physics]. doi: [10.48550/arXiv.2506.20554](https://doi.org/10.48550/arXiv.2506.20554). Available at <http://arxiv.org/abs/2506.20554> (accessed 10 August 2025).
- Morcrette C, Cave T, Reid H, Silva Rodrigues J da, Deveney T, Kreusser L, Van Weverberg K and Budd C** (2025) Scale-Aware Parameterization of Cloud Fraction and Condensate for a Global Atmospheric Model Machine-Learned From Coarse-Grained Kilometer-Scale Simulations. en. *Journal of Advances in Modeling Earth Systems*, 30 May **17**(4), e2024MS004651. ISSN: 1942-2466. doi: [10.1029/2024MS004651](https://doi.org/10.1029/2024MS004651). <https://onlinelibrary.wiley.com/doi/abs/10.1029/2024MS004651> (accessed 30 May 2025).
- Silver D and Sutton RS** (2025) Welcome to the Era of Experience. In *Designing an Intelligence*. MIT Press Cambridge. Available at <http://incompleteideas.net/papers/TheEraOfExperience.pdf> (accessed 31 May 2025).
- Su Z, Zhang B, Rahamanian N, Gao Y, Liao Q, Regan C, Sreenath K and Sastry SS** (2025) *HITTER: A Humanoid Table Tennis Robot via Hierarchical Planning and Learning*. arXiv:2508.21043 [cs]. doi: [10.48550/arXiv.2508.21043](https://doi.org/10.48550/arXiv.2508.21043). Available at <http://arxiv.org/abs/2508.21043> (accessed 28 November 2025).
- Ven GMvd, Soures N and Kudithipudi D** (2025) Continual Learning and Catastrophic Forgetting. In arXiv:2403.05175 [cs], 153–168. doi: [10.1101/2403.05175](https://doi.org/10.1101/2403.05175). Available at <http://arxiv.org/abs/2403.05175> (accessed 14 December 2025).

## Appendix A. Additional Methods

### A.1. RL Algorithm Summaries

Algorithm	Properties
Truncated Quantile Critics (TQC) (Kuznetsov et al. 2020)	<ul style="list-style-type: none"> <li>1. Off-policy actor-critic algorithm that builds on SAC with a distributional critic using quantile regression.</li> <li>2. Models the full distribution of returns <math>Z(s, a)</math> as quantiles <math>\{\tau_i\}</math>, capturing uncertainty and reducing bias.</li> <li>3. Discards the top <math>k</math> quantiles before computing target values to avoid overestimation from outlier returns.</li> <li>4. Provides a smooth and robust training signal for distributional value estimation by minimising the quantile Huber loss:</li> </ul> $\mathcal{L}_\tau = \frac{1}{N} \sum_{i=1}^N \rho_\kappa(\tau_i - y)$ <p>where <math>\tau_i</math> is the predicted <math>i</math>-th quantile, <math>y</math> is the target return, <math>\rho_\kappa</math> denotes the Huber loss with threshold <math>\kappa</math>, and <math>N</math> is the number of quantile estimates used in training.</p>
Action Value Gradient (AVG) (Vasan et al. 2024)	<ul style="list-style-type: none"> <li>1. On-policy actor-critic algorithm that combines reparameterised stochastic policies with log-probability regularisation to stabilise exploration.</li> <li>2. The actor minimises an entropy-augmented objective:</li> </ul> $\mathcal{L}_{\text{actor}} = \alpha \log \pi(a s) - Q(s, a) \quad \text{where } a \sim \tanh(\mathcal{N}(\mu_\theta(s), \sigma_\theta(s)))$ <p>This promotes both high-return and high-entropy policies by penalising confident low-reward actions.</p> <ul style="list-style-type: none"> <li>3. The critic is updated using TD learning with scaled errors:</li> </ul> $\delta = \frac{r + \gamma V(s') - Q(s, a)}{\hat{\sigma}_\delta} \quad \text{where } \hat{\sigma}_\delta \text{ is a running estimate of } \text{std}(\delta)$ <p>Normalising TD errors ensures stable critic gradients even in the presence of large reward magnitudes or early-stage noise.</p> <ul style="list-style-type: none"> <li>4. Supports robust learning in continuous control by incorporating tanh-based action squashing, TD error scaling, and optional clipping of actions to respect environment bounds.</li> </ul>

Table A.1: Four point summaries (contd.)

Algorithm	Properties
REINFORCE (Williams 1992)	<ul style="list-style-type: none"> <li>1. Off-policy Monte Carlo algorithm that performs updates only at the end of full trajectories (<math>\tau</math>), and does not include a critic.</li> <li>2. Uses a stochastic policy <math>\pi_\theta(a s)</math> to generate actions and samples episodes from the environment.</li> <li>3. Optimises the expected return via the score function (log-derivative) estimator: <math>\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(\tau) R(\tau)]</math>.</li> <li>4. Suffers from high variance and slow convergence, often mitigated by introducing baselines (advantage estimates) or reward normalisation.</li> </ul>
Deterministic Policy Gradient (DPG) (Silver, Lever et al. 2014)	<ul style="list-style-type: none"> <li>1. On-policy actor-critic algorithm where the actor uses a deterministic policy <math>\mu_\theta(s)</math>, suited to continuous action spaces.</li> <li>2. Critic is trained using TD learning to estimate the Q-function, while actor update uses gradient chain rule over Q.</li> <li>3. Gradient of the objective is <math>\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \rho^\mu} [\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a) _{a=\mu_\theta(s)}]</math>.</li> <li>4. Lacks target networks, making the learning process less stable and more sensitive to hyperparameters.</li> </ul>
Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2019)	<ul style="list-style-type: none"> <li>1. Off-policy actor-critic algorithm that extends DPG using deep function approximators and additional stabilisation mechanisms.</li> <li>2. Introduces experience replay and target networks with soft target updates to decorrelate samples and improve training stability.</li> <li>3. Actor and critic networks are both updated similar to DPG.</li> <li>4. Overestimation bias and sensitivity to exploration noise often limit performance unless mitigated by design changes (e.g. TD3).</li> </ul>
Twin Delayed DDPG (TD3) (Fujimoto et al. 2018)	<ul style="list-style-type: none"> <li>1. Off-policy actor-critic method designed to reduce the overestimation bias observed in DDPG.</li> <li>2. Uses a double critic architecture where the minimum of two Q-value estimates is used for critic updates.</li> <li>3. Actor is updated less frequently than the critics, and target networks are softly updated to reduce update variance.</li> <li>4. Injects temporally correlated Gaussian noise into the target actions to promote exploration in continuous action spaces.</li> </ul>

Table A.1: Four point summaries of different RL algorithms used

Algorithm	Properties
Trust Region Policy Optimisation (TRPO) (Schulman, Levine et al. 2015)	<ul style="list-style-type: none"> <li>1. On-policy stochastic actor-critic algorithm with a theoretical guarantee of monotonic policy improvement.</li> <li>2. Formulates a constrained optimisation problem using a KL-divergence bound to control step sizes and prevent policy collapse.</li> <li>3. Uses advantage-weighted surrogate objectives with a linear approximation and solves the constraint via conjugate gradient.</li> <li>4. Highly stable but computationally expensive due to second-order updates and line search in large parameter spaces.</li> </ul>
Proximal Policy Optimisation (PPO) (Schulman, Wolski et al. 2017)	<ul style="list-style-type: none"> <li>1. On-policy stochastic actor-critic algorithm designed as a first-order alternative to TRPO with comparable stability.</li> <li>2. Uses a clipped surrogate objective to prevent excessively large policy updates:</li> </ul> $L_{\text{CLIP}}(\theta) = \mathbb{E} \left[ \min \left( r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]$ <p>where <math>r_t(\theta) = \frac{\pi_\theta(a_t s_t)}{\pi_{\theta_{\text{old}}}(a_t s_t)}</math> and <math>\hat{A}_t</math> is an estimator of the advantage function.</p> <ul style="list-style-type: none"> <li>3. Leverages Generalised Advantage Estimation (GAE) to reduce variance of the policy gradient with tunable bias-variance trade-off.</li> <li>4. Efficient, robust and widely used in practice due to its balance of ease of implementation and empirical performance.</li> </ul>
Soft Actor-Critic (SAC) (Haarnoja et al. 2018)	<ul style="list-style-type: none"> <li>1. Off-policy actor-critic method for continuous control that augments the reward with an entropy term:</li> </ul> $J(\pi) = \sum_t \mathbb{E}_{(s_t, a_t) \sim p_\pi} [r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot   s_t))]$ <p>where <math>\mathcal{H}</math> is the entropy and <math>\alpha</math> is a temperature parameter.</p> <ul style="list-style-type: none"> <li>2. Encourages exploration by optimising for both reward and entropy, using a stochastic policy <math>\pi(a s)</math>.</li> <li>3. Maintains two Q-networks and uses the minimum value to reduce overestimation bias, with delayed soft target updates.</li> <li>4. Features automatic entropy tuning, high sample efficiency, and strong empirical performance on continuous control benchmarks.</li> </ul>

Table A.1: Four point summaries (contd.)

## A.2. RL Algorithm Pseudocodes

### A.2.1. REINFORCE

---

#### Algorithm 1 REINFORCE

---

- 1: **Input:** Gym environment, Number of episodes  $M$ , Steps per episode  $N$ , Learning rate  $\alpha$ , Discount factor  $\gamma$
- 2: **Initialise:** Policy network parameters  $\theta$ , Actor network  $\pi_\theta$ , Learning rate  $\alpha$
- 3: **Pre-Setup:** Configure seed and environment variables, prepare environment and logging
- 4:
- 5: **for**  $episode = 1$  **to**  $M$  **do**
- 6:     Initialise episode buffer  $B \leftarrow \emptyset$
- 7:     Observe initial state  $s_0$
- 8:     **for**  $t = 0$  **to**  $N - 1$  **do**
- 9:         Select action  $a_t \sim \pi_\theta(s_t)$
- 10:         Execute action  $a_t$  and observe reward  $r_t$  and new state  $s_{t+1}$
- 11:         Store transition  $(s_t, a_t, r_t)$  in  $B$
- 12:          $s_t \leftarrow s_{t+1}$
- 13:     **end for**
- 14:      $G \leftarrow 0$ ,  $\mathcal{L}(\theta) \leftarrow 0$
- 15:     **for**  $t$  **in**  $B$  reversed **do**
- 16:          $G \leftarrow r_t + \gamma G$
- 17:          $\mathcal{L}(\theta) \leftarrow \mathcal{L}(\theta) - \nabla_\theta G \log \pi_\theta(a_t | s_t)$
- 18:     **end for**
- 19:     Update policy parameters  $\theta$  using accumulated gradients:
- 20:
- $$\theta \leftarrow \theta + \eta \frac{1}{|B|} \mathcal{L}(\theta)$$
- 21: **end for**

---

### A.2.2. Deterministic Policy Gradient (DPG)

---

#### Algorithm 2 Deterministic Policy Gradient (DPG)

---

- 1: **Input:** Gym environment, Total timesteps  $T$ , Discount factor  $\gamma$ , Learning rate for policy  $\eta_\pi$ , Learning rate for Q-network  $\eta_Q$ , Batch size  $B$ , Exploration noise  $\sigma$
- 2: **Initialise:** Policy network parameters  $\theta$ , Q-function network parameters  $\phi$
- 3: **Pre-Setup:** Configure seed and environment variables, prepare environment and logging
- 4:
- 5: **for**  $t = 1$  **to**  $T$  **do**
- 6:     Observe state  $s$  and select action  $a = \pi_\theta(s) + \epsilon$ , where  $\epsilon \sim \mathcal{N}(0, \sigma)$
- 7:     Execute action  $a$  and observe next state  $s'$ , reward  $r$ , and termination signal  $d$
- 8:     Calculate Q-values:  

$$y(r, s', d) = r + \gamma(1 - d)Q_\phi(s', \pi_\theta(s'))$$
- 9:
- 10:    Update Q-function by minimising the loss:  

$$\phi \leftarrow \phi - \eta_Q \nabla_\phi (Q_\phi(s, a) - y(r, s', d))^2$$
- 11:
- 12:    Update policy by one step of gradient ascent:  

$$\theta \leftarrow \theta + \eta_\pi \nabla_\theta Q_\phi(s, \pi_\theta(s))$$
- 13:
- 14: **end for**

---

### A.2.3. Deep Deterministic Policy Gradient (DDPG)

---

**Algorithm 3** Deep Deterministic Policy Gradient (DDPG)

---

1: **Input:** Gym environment, Total timesteps  $T$ , Replay buffer size  $N$ , Discount factor  $\gamma$ , Target smoothing coefficient  $\tau$ , Batch size  $B$ , Learning rate  $\eta$ , Exploration noise  $\sigma$

2: **Initialise:** Policy network parameters  $\theta$ , Q-function network parameters  $\phi$ , target network parameters  $\theta_{\text{targ}}$ ,  $\phi_{\text{targ}}$ , empty replay buffer  $\mathcal{D}$

3: **Pre-Setup:** Configure seed and environment variables, prepare environment and logging

4:

5: **for**  $t = 1$  to  $T$  **do**

6:     Observe state  $s$  and select action  $a = \pi_\theta(s)$

7:     Add exploration noise  $a \leftarrow a + \epsilon$ , where  $\epsilon \sim \mathcal{N}(0, \sigma)$  if required

8:     Execute action  $a$  and observe next state  $s'$ , reward  $r$ , and termination signal  $d$

9:     Store transition  $(s, a, r, s', d)$  in  $\mathcal{D}$

10:    **if**  $t \geq \text{learning\_starts}$  **then**

11:       Sample a minibatch of  $B$  transitions  $(s, a, r, s', d)$  from  $\mathcal{D}$

12:       Compute target for Q-function update:

13:

$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \pi_{\theta_{\text{targ}}}(s'))$$

14:       Update Q-function by minimising the loss:

15:

$$\phi \leftarrow \phi - \eta \nabla_\phi \frac{1}{|B|} \sum_{(s, a, r, s', d) \in B} (Q_\phi(s, a) - y(r, s', d))^2$$

16:       Update policy by one step of gradient ascent:

17:

$$\theta \leftarrow \theta + \eta \nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \pi_\theta(s))$$

18:       Soft-update target networks:

19:

$$\theta_{\text{targ}} \leftarrow \tau\theta + (1 - \tau)\theta_{\text{targ}}, \quad \phi_{\text{targ}} \leftarrow \tau\phi + (1 - \tau)\phi_{\text{targ}}$$

20:      **end if**

21: **end for**

---

#### A.2.4. Twin Delayed DDPG (TD3)

---

**Algorithm 4** Twin Delayed DDPG (TD3)

---

1: **Input:** Gym environment, Total timesteps  $T$ , Learning rate  $\eta$ , Replay buffer size  $N$ , Discount factor  $\gamma$ , Target smoothing coefficient  $\tau$ , Batch size  $B$ , Policy noise  $\sigma_\pi$ , Noise clip  $\sigma_{\text{clip}}$ , Exploration noise  $\sigma_{\text{exploration}}$ , Policy update frequency  $f_\pi$

2: **Initialise:** Actor network  $\theta$ , Critic networks  $\phi_1, \phi_2$ , Target networks  $\theta_{\text{targ}}, \phi_{\text{targ},1}, \phi_{\text{targ},2}$ , Empty replay buffer  $\mathcal{D}$

3: **Pre-Setup:** Configure seed and environment variables, prepare environment and logging

4:

5: **for**  $t = 1$  **to**  $T$  **do**

6:     Observe state  $s$  and select action  $a = \pi_\theta(s)$

7:     Add exploration noise  $a \leftarrow a + \epsilon$ , where  $\epsilon \sim \mathcal{N}(0, \sigma_{\text{exploration}})$  if required

8:     Execute action  $a$  and observe next state  $s'$ , reward  $r$ , and done signal  $d$

9:     Store transition  $(s, a, r, s', d)$  in  $\mathcal{D}$

10:    **if**  $t \geq \text{learning\_starts}$  **then**

11:       Sample a minibatch of  $B$  transitions  $(s, a, r, s', d)$  from  $\mathcal{D}$

12:       Compute target actions:

13:

$$a' \leftarrow \pi_{\theta_{\text{targ}}}(s') + \text{clip}(\mathcal{N}(0, \sigma_\pi), -\sigma_{\text{clip}}, \sigma_{\text{clip}})$$

14:       Compute target Q-values:

15:

$$y(r, s', d) \leftarrow r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a')$$

16:       Update critic networks by minimising the loss:

17:

$$\phi_i \leftarrow \phi_i - \eta \nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2, \text{ for } i = 1, 2$$

18:       **if**  $t \bmod f_\pi = 0$  **then**

19:           Update actor network by policy gradient:

20:

$$\theta \leftarrow \theta + \eta \nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \pi_\theta(s))$$

21:       Soft update target networks:

22:

$$\theta_{\text{targ}} \leftarrow \tau\theta + (1 - \tau)\theta_{\text{targ}}, \quad \phi_{\text{targ},i} \leftarrow \tau\phi_i + (1 - \tau)\phi_{\text{targ},i} \text{ for } i = 1, 2$$

23:       **end if**

24:    **end if**

25: **end for**

---

### A.2.5. Trust Region Policy Optimization (TRPO)

---

**Algorithm 5** Trust Region Policy Optimization (TRPO)

---

- 1: **Input:** Gym environment, Total timesteps  $T$ , Mini-batch size  $M$ , Number of steps per episode  $N$ , Discount factor  $\gamma$ , GAE lambda  $\lambda$ , KL divergence limit  $\delta$ , Trust region update size  $\beta$
- 2: **Initialise:** Policy parameters  $\theta$ , Value function parameters  $\phi$
- 3: **Pre-Setup:** Configure seed and environment variables, prepare environment and logging
- 4:
- 5: **for**  $iteration = 1, 2, \dots, \frac{T}{N}$  **do**
- 6:     Collect set of trajectories  $\mathcal{D} = \{\tau_i\}$  by running policy  $\pi_\theta$  in the environment
- 7:     Compute returns  $\{R_i\}$  and advantage estimates  $\{\hat{A}_i\}$  using GAE
- 8:     **for**  $epoch = 1, 2, \dots, K$  **do**
- 9:         Shuffle  $D$  to create  $M$  mini-batches
- 10:         **for** each mini-batch  $t$  **do**
- 11:             Update value function by minimising the MSE loss:
- 12:
- 13:             Compute the surrogate objective (policy loss):
- 14:             
$$L^\pi(\theta) = \frac{1}{M} \sum_t \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t$$
- 15:             Compute policy gradient  $\nabla_\theta L^\pi(\theta)$
- 16:             Apply conjugate gradient to estimate the natural policy gradient  $\hat{g}$
- 17:             
$$\hat{g} \approx (\nabla_\theta^2 KL(\pi_{\theta_{old}} \| \pi_\theta))^{-1} \nabla_\theta L^\pi(\theta)$$
- 18:             Compute step size  $\alpha$  using line search:
- 19:             
$$\alpha = \sqrt{\frac{2\delta}{\hat{g}^T H \hat{g}}}, \text{ where } H \text{ is the Hessian of } KL(\pi_{\theta_{old}} \| \pi_\theta)$$
- 20:             Update policy  $\theta \leftarrow \theta + \alpha \hat{g}$  using an exponential increment strategy
- 21:         **end for**
- 22:         **break** if  $KL(\pi_{\theta_{old}} \| \pi_\theta) > \delta$
- 23:     **end for**
- 24: **end for**

---

### A.2.6. Proximal Policy Optimization (PPO)

---

**Algorithm 6** Proximal Policy Optimization (PPO)

---

1: **Input:** Gym environment, Total timesteps  $T$ , Number of steps per episode  $N$ , Mini-batch size  $M$ , Update epochs  $K$ , Learning rate  $\alpha$ , Discount factor  $\gamma$ , GAE lambda  $\lambda$ , Clipping parameter  $\epsilon$ , VF coefficient  $c_1$ , Entropy coefficient  $c_2$ , KL divergence limit  $\delta$

2: **Initialise:** Policy parameters  $\theta$ , Value function parameters  $\phi$

3: **Pre-Setup:** Configure seed and environment variables, prepare environment and logging

4:

5: **for**  $iteration = 1, 2, \dots, \frac{T}{N}$  **do**

6:   Collect set of trajectories  $D = \{\tau_i\}$  by running policy  $\pi_\theta$  in the environment

7:   Compute returns  $\{R_i\}$  and advantage estimates  $\{\hat{A}_i\}$  using GAE

8:   **for**  $epoch = 1, 2, \dots, K$  **do**

9:     Shuffle  $D$  to create  $M$  mini-batches

10:    **for** each mini-batch  $t$  **do**

11:     Compute ratio  $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$

12:     Compute clipped surrogate objective (policy loss):

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[ \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t) \right]$$

13:     Compute value function loss:

$$L^{VF}(\phi) = \left( V_\phi(s_t) - \hat{R}_t \right)^2$$

14:     Compute entropy:  $S[\pi_\theta](s_t)$

15:     Compute total loss:

$$L(\theta, \phi) = -L^{CLIP}(\theta) + c_1 L^{VF}(\phi) - c_2 S[\pi_\theta](s_t)$$

16:     Update  $\theta$  and  $\phi$  using stochastic gradient descent

17:    **end for**

18:    **break** if  $KL(\pi_{\theta_{old}} \| \pi_\theta) > \delta$

19:   **end for**

20: **end for**

---

### A.2.7. Soft Actor-Critic (SAC)

---

#### Algorithm 7 Soft Actor-Critic (SAC)

---

1: **Input:** Gym environment, Total timesteps  $T$ , Replay buffer size  $N$ , Discount factor  $\gamma$ , Target smoothing coefficient  $\tau$ , Batch size  $B$ , Learning rate for policy  $\eta_\pi$ , Learning rate for Q-network  $\eta_Q$

2: **Initialise:** Policy network parameters  $\theta$ , Critic network parameters  $\phi_1, \phi_2$ , Target critic parameters  $\phi_{\text{targ},1}, \phi_{\text{targ},2}$ , Empty replay buffer  $\mathcal{D}$ , actor  $\pi_\theta$ , Entropy coefficient  $\alpha$ , Target entropy coefficient  $\alpha_{\text{targ}}$

3: **Pre-Setup:** Configure seed and environment variables, prepare environment and logging

4:

5: **for**  $t = 1$  **to**  $T$  **do**

6:     Observe state  $s$  and select action  $a \sim \pi_\theta(s)$  with exploration strategy if required

7:     Execute action  $a$  and observe next state  $s'$ , reward  $r$ , and termination signal  $d$

8:     Store transition  $(s, a, r, s', d)$  in  $\mathcal{D}$

9:     **if**  $t \geq \text{learning\_starts}$  **then**

10:         Sample a minibatch of  $B$  transitions  $(s, a, r, s', d)$  from  $\mathcal{D}$

11:         Compute targets for critic updates:

12:

$$y(r, s', d) = r + \gamma(1 - d) \left( \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s') \right)$$

13:         where  $\tilde{a}' \sim \pi_\theta(s')$

14:         Update Q-functions by one step of gradient descent:

15:

$$\phi_i \leftarrow \phi_i - \eta_Q \nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2 \text{ for } i = 1, 2$$

16:         Update policy by one step of gradient ascent:

17:

$$\theta \leftarrow \theta + \eta_\pi \nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left( \min_{i=1,2} Q_{\phi_i}(s, \pi_\theta(s)) - \alpha \log \pi_\theta(a|s) \right)$$

18:         Soft-update target networks:

19:

$$\phi_{\text{targ},i} \leftarrow \tau \phi_i + (1 - \tau) \phi_{\text{targ},i} \text{ for } i = 1, 2$$

20:         Optionally adjust  $\alpha$  based on entropy targets:

21:

$$\alpha \leftarrow \alpha + \eta_Q \nabla_\alpha \frac{\alpha}{|B|} \sum_{s \in B} (\log \pi_\theta(a|s) + \alpha_{\text{targ}})$$

22:         **end if**

23:     **end for**

---

### A.2.8. Truncated Quantile Critics (TQC)

---

**Algorithm 8** Truncated Quantile Critics (TQC)

---

1: **Input:** Gym environment, Total timesteps  $T$ , Replay buffer size  $N$ , Discount factor  $\gamma$ , Smoothing coefficient  $\tau$ , Batch size  $B$ , Learning rate  $\eta$ , Number of quantiles  $N_q$ , Number of critics  $N_c$ , Drop quantiles  $N_{\text{drop}}$ , Entropy coefficient  $\alpha$ , Target entropy coefficient  $\alpha_{\text{targ}}$

2: **Initialise:** Actor network  $\theta$ , Critic network parameters  $\phi_1, \dots, \phi_{N_c}$ , Target critic network parameters  $\phi_{\text{targ},1}, \dots, \phi_{\text{targ},N_c}$ , Replay buffer  $\mathcal{D}$

3: **Pre-Setup:** Configure seed and environment variables, prepare environment and logging

4:

5: **for**  $t = 1$  **to**  $T$  **do**

6:     Select action  $a \sim \pi_\theta(s)$  based on current policy and exploration strategy

7:     Execute action  $a$  and observe next state  $s'$ , reward  $r$ , and done signal  $d$

8:     Store transition tuple  $(s, a, r, s', d)$  in  $\mathcal{D}$

9:     **if**  $t \geq \text{learning\_starts}$  **then**

10:         **for**  $i = 1$  **to**  $N_c$  **do**

11:             Sample a minibatch of  $B$  transitions  $(s, a, r, s', d)$  from  $\mathcal{D}$

12:             Compute target quantile values for critic  $\phi_{\text{target},i}$ :

13:             
$$y(r, s', d) = r + \gamma(1 - d) \left( Q_{\phi_{\text{targ},i}}(s', \tilde{a}', N_{\text{drop}}) - \alpha \log \pi_\theta(\tilde{a}'|s') \right)$$

14:             where  $\tilde{a}' \sim \pi_\theta(s')$

15:             Update critic  $\phi_i$  by minimising the quantile Huber loss:

16:             
$$\mathcal{L}^{\phi_i} = \frac{1}{N_q} \sum_{k=1}^{N_q} \text{HuberLoss}(Q_{\phi_i}(s_j, a_j, \tau_k) - y_j)$$

17:             where  $\tau_k$  are the quantile fractions

18:         **end for**

19:         Update policy by one step of gradient ascent:

20:         
$$\theta \leftarrow \theta + \eta \nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left( -\alpha \log \pi_\theta(a|s) + \frac{1}{N_c} \sum_{i=1}^{N_c} Q_{\phi_i}(s, \pi_\theta(s)) \right)$$

21:         Soft-update target networks:

22:         **Optionally adjust  $\alpha$  based on entropy targets:**

23:         
$$\alpha \leftarrow \alpha + \eta \nabla_\alpha \frac{\alpha}{|B|} \sum_{s \in B} (\log \pi_\theta(a|s) + \alpha_{\text{targ}})$$

24:         **end if**

25:     **end for**

---

### A.2.9. Action Value Gradient (AVG)

---

#### Algorithm 9 Action Value Gradient (AVG)

---

1: **Input:** Gym environment, Total timesteps  $T$ , Discount factor  $\gamma$ , Entropy coefficient  $\alpha$   
 2: **Initialise:** Actor network parameters  $\theta$ , Critic network parameters  $\phi$ , actor  $\pi_\theta(a|s)$ , critic  $Q_\phi(s, a)$ , TD error normaliser  $\hat{\sigma}_\delta$ , Return  $G$   
 3: **Pre-Setup:** Configure seed and environment variables, prepare environment and logging  
 4:  
 5: **for**  $t = 1$  **to**  $T$  **do**  
 6:     Select action  $a \sim \pi_\theta(s)$  using squashed  $\tanh$   
 7:     Execute action  $a$  and observe next state  $s'$ , reward  $r$ , and done signal  $d$   
 8:     Compute entropy-augmented reward:  $r_{\text{ent}} = r - \alpha \log \pi_\theta(a|s)$   
 9:     Accumulate return:  $G \leftarrow G + r_{\text{ent}}$   
 10:    **if** episode done **then**  
 11:      Update TD error scaler  $\hat{\sigma}_\delta$  using  $G$  and  $r_{\text{ent}}$   
 12:       $G \leftarrow 0$   
 13:    **else**  
 14:      Update TD error scaler  $\hat{\sigma}_\delta$  using  $\gamma$  and  $r_{\text{ent}}$   
 15:    **end if**  
 16:    Compute target value:  $V(s') = Q_\phi(s', \pi_\theta(s')) - \alpha \log \pi_\theta(a'|s')$  where  $a' \sim \pi_\theta(s')$   
 17:    Compute the critic loss as the square of the scaled TD error:  

$$\mathcal{L}^\phi = \left[ \frac{r + \gamma(1-d)V(s') - Q_\phi(s, a)}{\hat{\sigma}_\delta} \right]^2$$
  
 18:    Compute the actor loss:  

$$\mathcal{L}^\theta = \alpha \log \pi_\theta(a|s) - Q_\phi(s, a)$$
  
 19:    Update both  $\theta$  and  $\phi$  through one step of gradient descent  
 20: **end for**

---

### A.3. climateRL Experiment Codes

Group	Environment	Experiment ID
Simple Climate Bias Correction	SimpleClimateBiasCorrection-v0	scbc-v0-optim-L scbc-v0-optim-L-60k scbc-v0-homo-64L scbc-v0-homo-64L-60k
	SimpleClimateBiasCorrection-v1	scbc-v1-optim-L scbc-v1-optim-L-60k scbc-v1-homo-64L scbc-v1-homo-64L-60k
	SimpleClimateBiasCorrection-v2	scbc-v2-optim-L scbc-v2-optim-L-60k scbc-v2-homo-64L scbc-v2-homo-64L-60k
Radiative Convective Equilibrium (RCE)	RadiativeConvectiveModel-v0	rce-v0-optim-L rce-v0-optim-L-10k rce-v0-homo-64L rce-v0-homo-64L-10k
	RadiativeConvectiveModel17-v0	rce17-v0-optim-L rce17-v0-optim-L-10k rce17-v0-homo-64L rce17-v0-homo-64L-10k
	RadiativeConvectiveModel17-v1	rce17-v1-optim-L rce17-v1-optim-L-10k rce17-v1-homo-64L rce17-v1-homo-64L-10k
Energy Balance Model (EBM)	EnergyBalanceModel-v0	ebm-v0-optim-L ebm-v0-optim-L-20k ebm-v0-homo-64L ebm-v0-homo-64L-20k
	EnergyBalanceModel-v1	ebm-v1-optim-L ebm-v1-optim-L-20k ebm-v1-homo-64L ebm-v1-homo-64L-20k

Table A.2: Experiment codes for each single-agent RL environment (each run across 10 seeds)

<b>Group</b>	<b>FedRL Environment</b>	<b>Experiment ID</b>
Energy Balance Model (EBM)	EnergyBalanceModel-v2	<b>ebm-v2-optim-L-20k-a6-fed05</b> <b>ebm-v2-optim-L-20k-a6-fed10</b> <b>ebm-v2-optim-L-20k-a6-nofed</b>
		<b>ebm-v2-optim-L-20k-a2-fed10</b> <b>ebm-v2-optim-L-20k-a2-nofed</b> <b>ebm-v2-optim-L-20k-a2-fed10</b>
Energy Balance Model (EBM)	EnergyBalanceModel-v3	<b>ebm-v3-optim-L-20k-a6-fed05</b> <b>ebm-v3-optim-L-20k-a6-fed10</b> <b>ebm-v3-optim-L-20k-a6-nofed</b>
		<b>ebm-v3-optim-L-20k-a2-fed05</b> <b>ebm-v3-optim-L-20k-a2-fed10</b> <b>ebm-v3-optim-L-20k-a2-nofed</b>

Table A.3: Experiment codes for each FedRL environment (each run across 10 seeds)

## Appendix B. Additional Results

### B.1. Single-agent RL

#### B.1.1. SCBC Environment

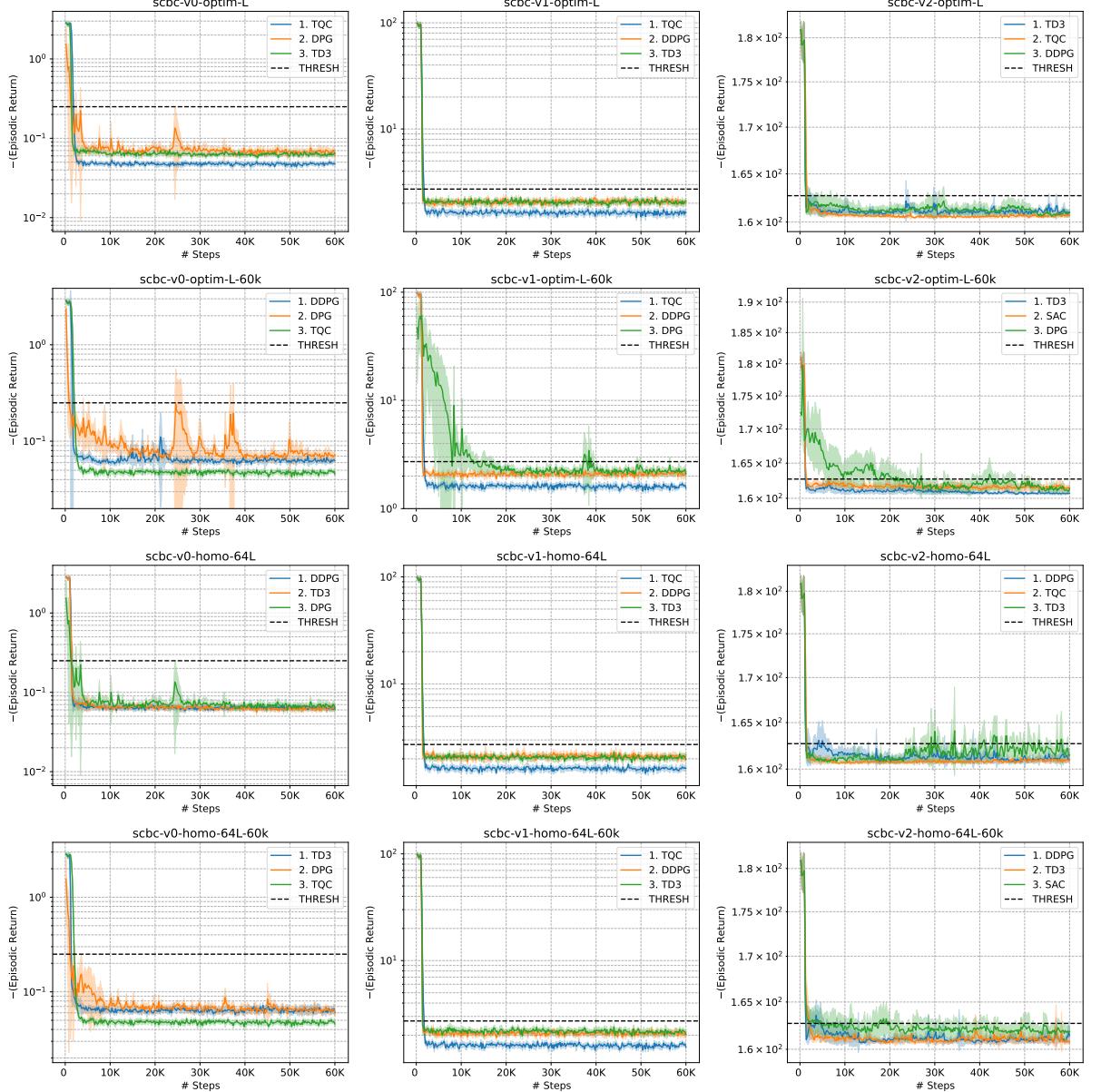


Figure B.1: Training curves across 10 seeds for SCBC environments (v0, v1, v2) across all twelve tuning configurations (scbc-v0/1/2-optim-L-60k reproduced for easy reference). Episodic returns are plotted on a log scale. Shaded regions denote  $\pm 1.96$  standard deviation (95% confidence intervals). Threshold values are mentioned in Table 3.

## B.1.2. RCE Environment

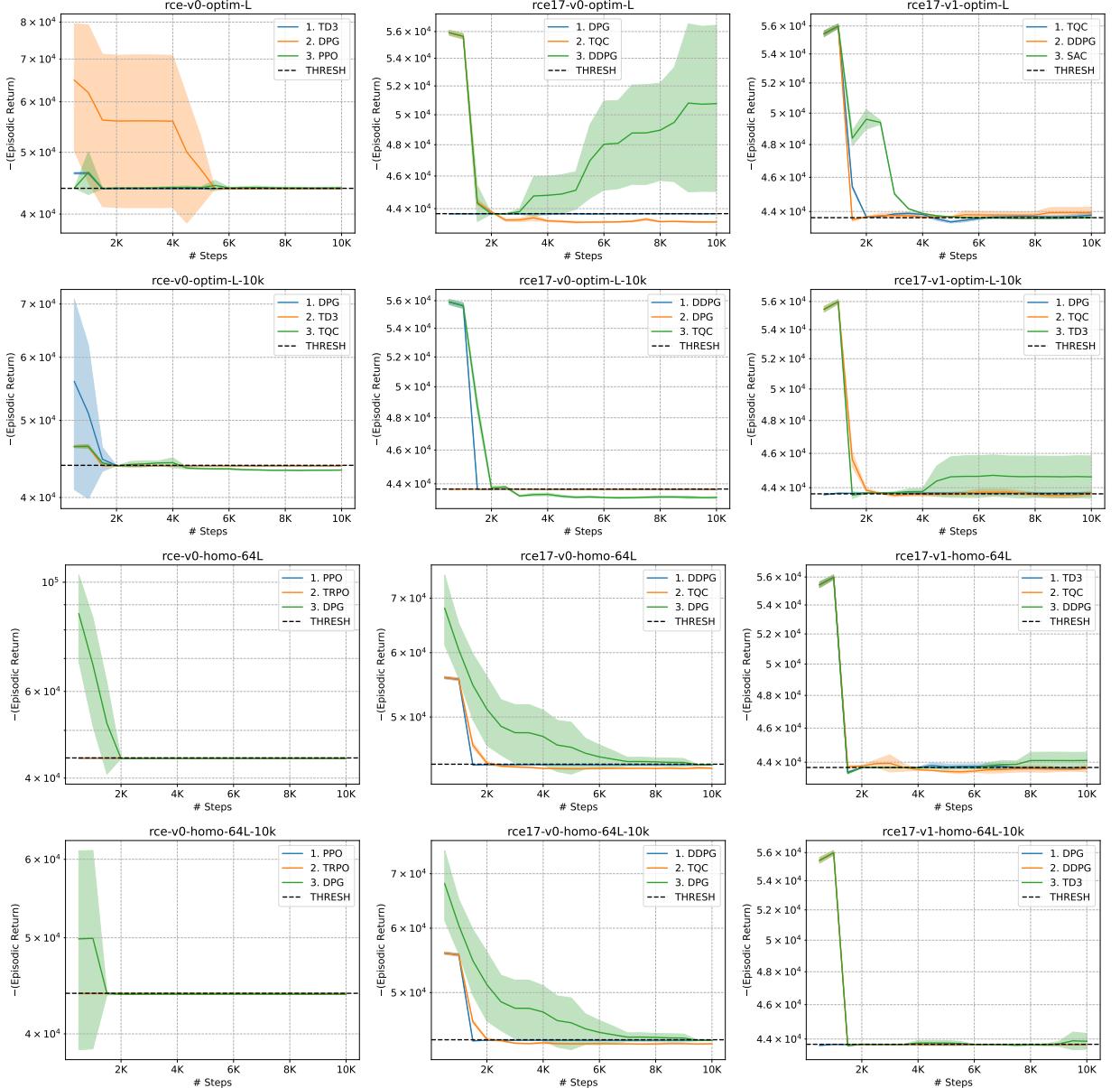


Figure B.2: Training curves for RCE environments (rce-v0, rce17-v0, rce17-v1) across all twelve tuning configurations (rce-v0/rce17-v0/1-optim-L-10k reproduced for easy reference). Episodic returns are plotted on a log scale. Shaded regions denote  $\pm 1.96$  standard deviation (95% confidence intervals). Threshold values are mentioned in Table 3.

### B.1.3. EBM Environment

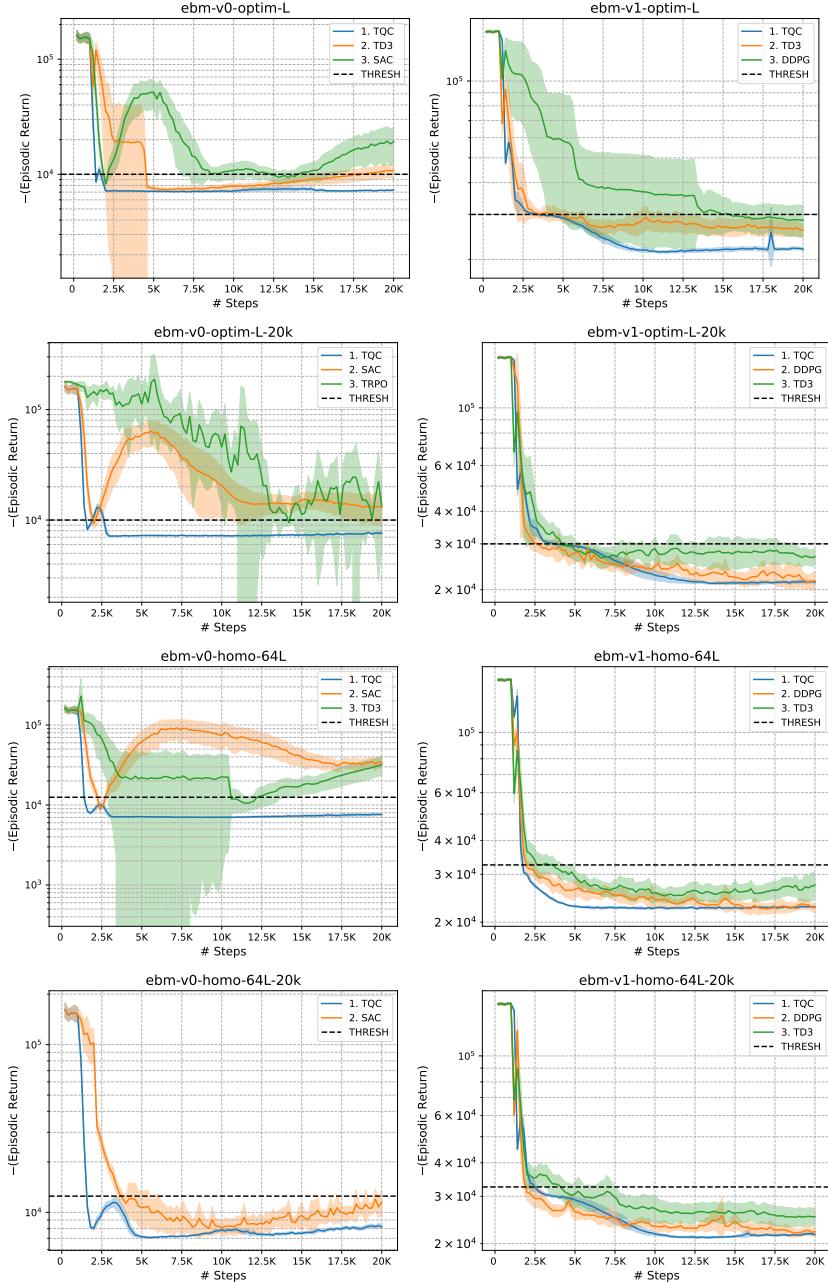


Figure B.3: Episodic return curves (log-scaled) with 95% spreads over 10 seeds for the top-3 RL algorithms across eight single-agent EBM configurations (ebm-v0/1-optim-L-20k reproduced for easy reference). Threshold values are mentioned in Table 3.

## B.2. Multi-agent RL

### B.2.1. FedRL Skill Metrics

		fed05			fed10			nofed				
		climlab	Mean ± Std	Gain %	Mean ± Std	Gain %	Mean ± Std	Gain %	Mean ± Std	Gain %	climlab	Gain %
90°S–60°S	11.453	5.11 ± 0.533	37.550	6.14 ± 0.964	25.040	16.17 ± 14.718	-97.490	8.19 ± 3.433				
60°S–30°S	7.768	3.08 ± 1.155	51.030	3.39 ± 0.775	46.080	10.30 ± 10.647	-63.620	6.30 ± 4.019				
30°S–0°	2.730	3.46 ± 1.984	48.890	3.92 ± 1.642	42.080	12.24 ± 13.889	-81.120	6.76 ± 3.013				
0°–30°N	3.746	2.80 ± 2.384	39.050	1.89 ± 1.325	58.760	2.23 ± 1.246	51.460	4.59 ± 2.068				
30°N–60°N	6.398	2.35 ± 0.866	11.350	2.26 ± 1.307	14.730	2.38 ± 1.253	10.360	2.65 ± 0.997				
60°N–90°N	5.566	1.60 ± 0.682	54.090	1.95 ± 0.995	44.060	2.49 ± 0.758	28.400	3.48 ± 1.790				

		fed05			fed10			nofed				
		climlab	Mean ± Std	Gain %	Mean ± Std	Gain %	Mean ± Std	Gain %	Mean ± Std	Gain %	climlab	Gain %
90°S–60°S	11.453	8.00 ± 1.380	17.850	6.87 ± 0.825	29.520	7.72 ± 1.371	20.750	9.74 ± 3.330				
60°S–30°S	7.768	7.32 ± 2.327	-2.870	5.51 ± 2.337	22.530	4.98 ± 1.696	29.980	7.12 ± 3.800				
30°S–0°	2.730	4.47 ± 1.889	-17.750	4.77 ± 2.313	-25.480	3.49 ± 1.969	8.210	3.80 ± 1.713				
0°–30°N	3.746	4.03 ± 2.409	-42.080	5.67 ± 3.152	-100.100	4.06 ± 2.277	-43.360	2.83 ± 1.247				
30°N–60°N	6.398	4.30 ± 2.556	14.860	4.79 ± 3.138	5.170	4.39 ± 1.941	13.060	5.06 ± 2.602				
60°N–90°N	5.566	3.54 ± 1.867	34.680	3.50 ± 1.518	35.360	5.63 ± 1.948	-4.030	5.42 ± 2.721				

		fed05			fed10			nofed				
		climlab	Mean ± Std	Gain %	Mean ± Std	Gain %	Mean ± Std	Gain %	Mean ± Std	Gain %	climlab	Gain %
90°S–60°S	11.453	8.28 ± 0.896	8.550	8.13 ± 0.765	10.300	8.85 ± 1.124	2.320	9.06 ± 1.549				
60°S–30°S	7.768	7.46 ± 1.618	6.440	7.08 ± 1.382	11.120	8.24 ± 1.716	-3.390	7.97 ± 1.459				
30°S–0°	2.730	2.83 ± 1.195	-26.510	2.34 ± 1.053	-4.390	3.32 ± 1.269	-48.170	2.24 ± 0.868				
0°–30°N	3.746	2.30 ± 0.511	-25.400	2.19 ± 0.604	-19.480	2.49 ± 1.074	-35.470	1.84 ± 0.561				
30°N–60°N	6.398	0.93 ± 0.222	61.460	0.92 ± 0.149	61.860	1.23 ± 0.423	49.160	2.42 ± 0.706				
60°N–90°N	5.566	1.34 ± 0.219	42.680	1.33 ± 0.352	43.140	1.44 ± 0.297	38.240	2.33 ± 0.767				

Table B.4: Zonal-band errors for ebm-v2-optim-L-20k-a2. Each subtable reports mean ± std and relative gain % versus ebm-v1 for three regimes fed05, fed10 and nofed, along with a comparison against the static baseline climlab.

(a) DDPG

climlab	fed@5			fed@10			nofed			ebm-v1		
	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %
90°S–60°S	11.453	7.26 ± 1.845	11.340	8.17 ± 2.886	0.260	20.52 ± 11.476	-150.600	8.19 ± 3.433				
60°S–30°S	7.768	1.63 ± 1.478	74.100	1.51 ± 0.324	76.020	1.62 ± 0.843	74.240	6.30 ± 4.019				
30°S–0°	2.730	1.92 ± 1.029	71.600	1.79 ± 0.624	73.580	2.31 ± 0.776	65.870	6.76 ± 3.013				
0°–30°N	3.746	1.89 ± 1.032	58.750	2.49 ± 1.509	45.850	2.59 ± 1.607	43.640	4.59 ± 2.068				
30°N–60°N	6.398	1.71 ± 0.697	35.620	2.19 ± 0.643	17.530	1.81 ± 0.998	31.730	2.65 ± 0.997				
60°N–90°N	5.566	2.43 ± 1.643	30.190	2.30 ± 1.338	34.050	2.42 ± 1.224	30.670	3.48 ± 1.790				

(b) TD3

climlab	fed@5			fed@10			nofed			ebm-v1		
	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %
90°S–60°S	11.453	7.01 ± 2.553	28.000	6.04 ± 0.933	38.030	15.90 ± 9.258	-63.150	9.74 ± 3.330				
60°S–30°S	7.768	3.16 ± 1.133	55.560	3.52 ± 1.276	50.510	3.52 ± 1.627	50.480	7.12 ± 3.800				
30°S–0°	2.730	7.68 ± 2.389	-102.170	8.16 ± 1.813	-114.730	6.10 ± 2.088	-60.670	3.80 ± 1.713				
0°–30°N	3.746	7.27 ± 2.023	-156.640	7.63 ± 1.859	-169.470	6.05 ± 2.532	-113.500	2.83 ± 1.247				
30°N–60°N	6.398	3.92 ± 1.599	22.380	3.80 ± 0.851	24.770	3.47 ± 1.360	31.460	5.06 ± 2.602				
60°N–90°N	5.566	2.97 ± 1.614	45.090	2.48 ± 1.563	54.120	5.55 ± 5.617	-2.380	5.42 ± 2.721				

(c) TQC

climlab	fed@5			fed@10			nofed			ebm-v1		
	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %
90°S–60°S	11.453	34.96 ± 7.034	-285.900	28.35 ± 7.956	-212.890	33.90 ± 7.319	-274.160	9.06 ± 1.549				
60°S–30°S	7.768	1.68 ± 1.087	78.890	1.69 ± 1.215	78.780	1.28 ± 0.393	83.920	7.97 ± 1.459				
30°S–0°	2.730	1.97 ± 1.814	11.850	2.25 ± 1.590	-0.480	0.80 ± 0.213	64.350	2.24 ± 0.868				
0°–30°N	3.746	1.21 ± 0.670	34.310	1.77 ± 1.363	3.430	0.75 ± 0.190	59.300	1.84 ± 0.561				
30°N–60°N	6.398	1.97 ± 1.492	18.750	1.73 ± 0.551	28.600	1.17 ± 0.330	51.920	2.42 ± 0.706				
60°N–90°N	5.566	30.70 ± 8.534	-1215.560	32.88 ± 9.606	-1308.920	43.12 ± 14.594	-1747.640	2.33 ± 0.767				

Table B.5: Zonal-band errors for ebm-v2-optim-L-20k-a6. Each subtable reports mean ± std and relative gain % versus ebm-v1 for three regimes fed@5, fed@10 and nofed, along with a comparison against the static baseline climlab.

(a) DDPG

climlab	fed@5			fed@10			nofed		
	Mean ± Std	Gain %	ebm-v1						
90°S–60°S	11.453	6.69 ± 1.766	18.240	7.52 ± 2.718	8.180	7.76 ± 2.193	5.200	8.19 ± 3.433	
60°S–30°S	7.768	3.30 ± 1.168	47.540	4.20 ± 1.272	33.290	3.98 ± 2.135	36.810	6.30 ± 4.019	
30°S–0°	2.730	4.84 ± 2.151	28.340	3.77 ± 2.190	44.220	3.26 ± 1.873	51.720	6.76 ± 3.013	
0°–30°N	3.746	2.42 ± 1.786	47.180	2.96 ± 2.276	35.560	2.49 ± 1.461	45.840	4.59 ± 2.068	
30°N–60°N	6.398	2.00 ± 0.885	24.650	2.73 ± 1.056	2.980	2.67 ± 1.400	-0.700	2.65 ± 0.997	
60°N–90°N	5.566	1.96 ± 0.681	43.670	1.69 ± 1.035	51.400	1.63 ± 1.024	53.230	3.48 ± 1.790	

(b) TD3

climlab	fed@5			fed@10			nofed		
	Mean ± Std	Gain %	ebm-v1						
90°S–60°S	11.453	7.53 ± 1.444	22.700	7.42 ± 1.159	23.880	7.54 ± 1.390	22.630	9.74 ± 3.330	
60°S–30°S	7.768	5.99 ± 2.663	15.800	5.51 ± 2.616	22.590	5.00 ± 2.586	29.790	7.12 ± 3.800	
30°S–0°	2.730	3.84 ± 2.378	-1.030	3.65 ± 2.640	3.990	3.32 ± 1.380	12.690	3.80 ± 1.713	
0°–30°N	3.746	7.52 ± 2.875	-165.300	7.54 ± 3.263	-166.260	5.94 ± 2.701	-109.610	2.83 ± 1.247	
30°N–60°N	6.398	6.55 ± 1.837	-29.630	6.85 ± 2.390	-35.450	7.02 ± 2.581	-38.880	5.06 ± 2.602	
60°N–90°N	5.566	3.94 ± 2.302	27.250	4.00 ± 1.708	26.240	4.49 ± 1.536	17.170	5.42 ± 2.721	

(c) TQC

climlab	fed@5			fed@10			nofed		
	Mean ± Std	Gain %	Mean ± Std	Gain %	Mean ± Std	Gain %	Mean ± Std	Gain %	ebm-v1
90°S–60°S	11.453	8.27 ± 0.378	8.700	10.78 ± 4.522	-18.950	8.26 ± 0.303	8.840	9.06 ± 1.549	
60°S–30°S	7.768	7.51 ± 0.869	5.780	10.76 ± 4.459	-35.000	7.62 ± 0.729	4.450	7.97 ± 1.459	
30°S–0°	2.730	2.60 ± 0.718	-15.960	7.27 ± 5.706	-224.680	3.02 ± 0.490	-34.920	2.24 ± 0.868	
0°–30°N	3.746	2.84 ± 0.774	-54.750	9.67 ± 12.637	-426.670	3.90 ± 0.434	-112.690	1.84 ± 0.561	
30°N–60°N	6.398	3.59 ± 1.179	-48.070	11.05 ± 17.379	-355.740	4.13 ± 0.461	-70.350	2.42 ± 0.706	
60°N–90°N	5.566	3.35 ± 1.003	-43.330	11.17 ± 18.477	-378.590	3.36 ± 0.394	-43.800	2.33 ± 0.767	

Table B.6: Zonal-band errors for `ebm-v3-optim-L-20k-a2`. Each subtable reports mean ± std and relative gain % versus `ebm-v1` for three regimes `fed@5`, `fed@10` and `nofed`, along with a comparison against the static baseline `climlab`.

(a) DDPG

climlab	fed05			fed10			nofed			ebm-v1		
	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %
90°S–60°S	11.453	7.01 ± 2.782	14.340	7.22 ± 1.399	11.800	7.23 ± 1.652	11.690	8.19 ± 3.433				
60°S–30°S	7.768	4.34 ± 3.372	31.000	3.69 ± 1.102	41.450	8.08 ± 4.816	-28.370	6.30 ± 4.019				
30°S–0°	2.730	1.25 ± 0.695	81.440	1.22 ± 0.526	81.890	19.92 ± 4.845	-194.680	6.76 ± 3.013				
0°–30°N	3.746	1.48 ± 0.617	67.830	1.71 ± 1.218	62.710	17.39 ± 5.346	-278.850	4.59 ± 2.068				
30°N–60°N	6.398	1.51 ± 0.710	43.220	1.57 ± 0.796	40.880	5.92 ± 6.272	-123.010	2.65 ± 0.997				
60°N–90°N	5.566	1.17 ± 0.442	66.490	1.39 ± 0.680	60.240	1.76 ± 1.176	49.480	3.48 ± 1.790				

(b) TD3

climlab	fed05			fed10			nofed			ebm-v1		
	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %
90°S–60°S	11.453	14.05 ± 2.941	-44.230	15.86 ± 1.632	-62.780	12.47 ± 8.601	-28.000	9.74 ± 3.330				
60°S–30°S	7.768	10.90 ± 0.644	-53.120	10.59 ± 0.481	-48.860	9.36 ± 5.302	-31.570	7.12 ± 3.800				
30°S–0°	2.730	21.30 ± 0.639	-460.730	21.14 ± 0.473	-456.380	17.06 ± 4.570	-349.090	3.80 ± 1.713				
0°–30°N	3.746	21.41 ± 0.467	-655.550	21.23 ± 0.611	-649.370	14.34 ± 2.496	-405.970	2.83 ± 1.247				
30°N–60°N	6.398	9.47 ± 0.516	-87.290	9.36 ± 0.581	-85.160	5.78 ± 0.977	-14.340	5.06 ± 2.602				
60°N–90°N	5.566	4.57 ± 0.484	15.600	4.48 ± 0.536	17.280	9.27 ± 6.494	-71.180	5.42 ± 2.721				

(c) TQC

climlab	fed05			fed10			nofed			ebm-v1		
	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %	Mean	± Std	Gain %
90°S–60°S	11.453	21.12 ± 1.703	-133.090	23.23 ± 8.282	-156.370	21.01 ± 1.702	-131.930	9.06 ± 1.549				
60°S–30°S	7.768	18.84 ± 0.971	-136.340	22.08 ± 8.752	-177.000	18.43 ± 1.047	-131.180	7.97 ± 1.459				
30°S–0°	2.730	9.12 ± 0.779	-307.220	10.84 ± 4.123	-383.720	8.54 ± 0.511	-281.410	2.24 ± 0.868				
0°–30°N	3.746	9.41 ± 0.789	-412.530	10.82 ± 2.018	-489.610	8.62 ± 0.503	-369.770	1.84 ± 0.561				
30°N–60°N	6.398	11.87 ± 2.754	-389.580	15.20 ± 5.150	-527.000	9.85 ± 0.519	-306.400	2.42 ± 0.706				
60°N–90°N	5.566	14.21 ± 7.349	-509.050	20.12 ± 8.341	-762.090	9.07 ± 1.682	-288.430	2.33 ± 0.767				

Table B.7: Zonal-band errors for ebm-v3-optim-L-20k-a6. Each subplot reports mean ± std and relative gain % versus ebm-v1 for three regimes fed05, fed10 and nofed, along with a comparison against the static baseline climlab.

## B.2.2. Local Skill Plots

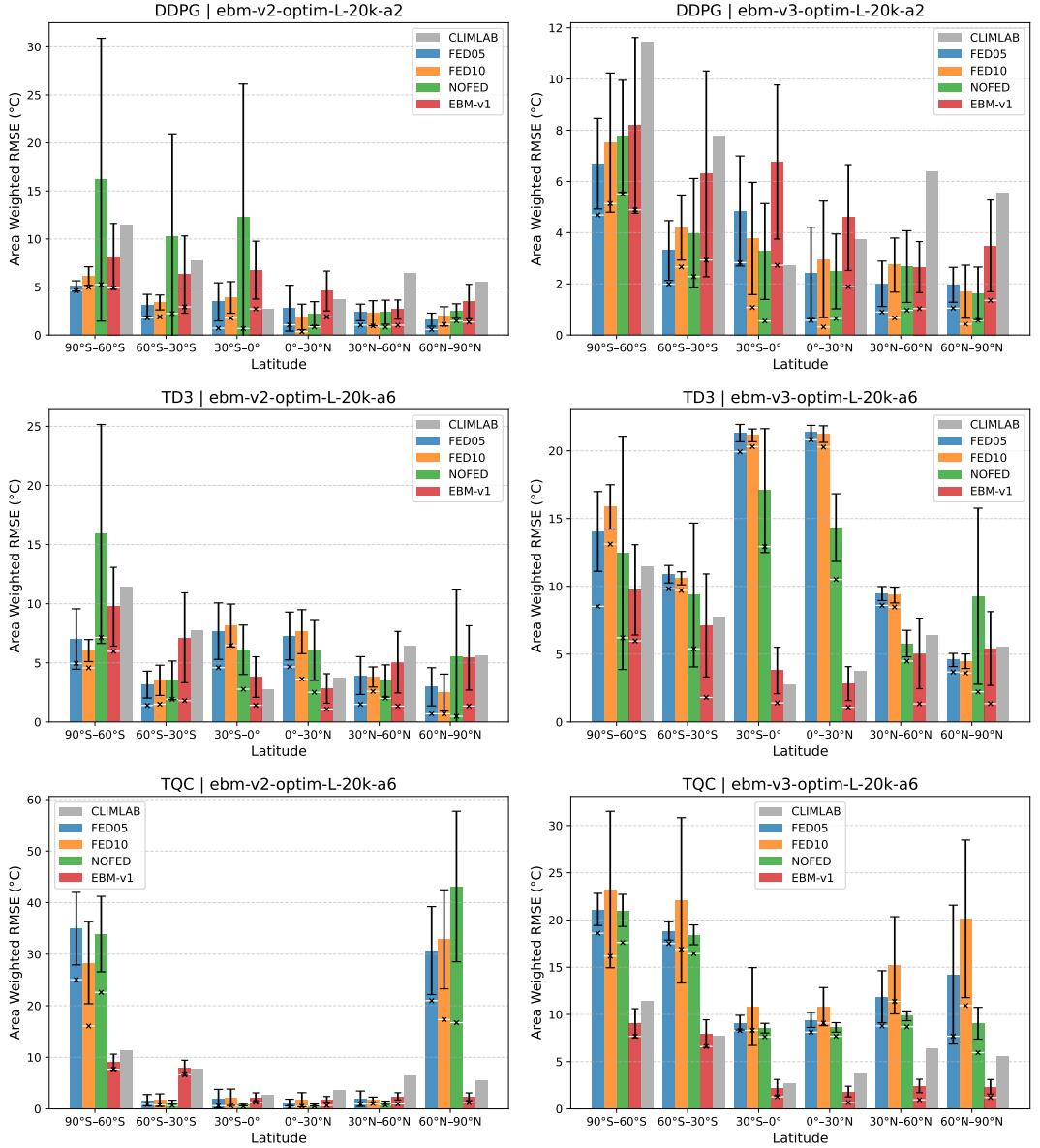


Figure B.4: Comparison of zonal skill achieved by DDPG, TD3 and TQC under FedRL coordination in *ebm-v2* and *ebm-v3*, using the 2-agent spatial decomposition (DDPG) and 6-agent spatial decomposition (a6 - TD3 and TQC). Skill is evaluated using areaWRMSE between predicted and reference temperature profiles, averaged with 95% CI spreads over 10 seeds. Each subplot reports results for three FedRL schemes: *fed05*, *fed10*, *nofed*, along with single-agent *ebm-v1* and the static *climlab* baseline. White horizontal bars with a cross indicate the best-performing seed for each scheme. Both setups adopt the same policy network architecture and hyperparameters as *ebm-v1*. Tabulated results in Appendix B.2.1.

For TD3 (second row in Figure B.4), `ebm`-v2 results show competitive skill in tropical and mid-latitude zones under `fed05`, often matching or surpassing the single-agent `ebm`-v1. However, variance rises sharply in the polar bands, particularly in the Southern Hemisphere, where strong gradients prove harder to capture. In `ebm`-v3, TD3 exhibits more pronounced instability: although `fed05` remains the most stable regime, episodic collapses at high latitudes drive RMSE higher than in `ebm`-v2. This behaviour suggests that the reduced, region-specific inputs of `ebm`-v3 may conflict with hyperparameters tuned for global inputs in `ebm`-v1, causing mismatches in critic ensemble updates and amplifying instability.

TQC (third row in Figure B.4) performs strongly in the tropical bands of `ebm`-v2 under `fed05`, achieving clear gains over the single-agent `ebm`-v1. Yet the method shows instability at high latitudes and wider error spreads under `fed10`, highlighting its dependence on frequent synchronisation for stability. In `ebm`-v3, performance degrades further, with mid-latitude instability and polar areaWRMSE in several cases exceeding that of `ebm`-v1. The large critic ensemble that benefits global contexts may be less effective when learning from regional input profiles and localised rewards, leading to overfitting or noisy updates. Overall, these results suggest that while both TD3 and TQC can deliver strong performance in favourable regimes, DDPG’s simpler architecture is more resilient to the structural shifts between `ebm`-v2 and `ebm`-v3.