STATE FLOW CREDIT ASSIGNMENT: IMPROVING CREDIT ASSIGNMENT WITH GENERATIVE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Credit assignment is a fundamental challenge in reinforcement learning and is critical for both learning stability and policy optimality. Prior work has explored counterfactual and hindsight approaches, which typically adopt a retrospective perspective, modeling influence factors from future trajectories. As a result, the credit assigned to individual actions can be entangled with the effects of subsequent actions, making it difficult to isolate their true contributions. To address this challenge, we propose *State Flow Credit Assignment* (SFCA), a novel framework that integrates generative modeling into the credit assignment process. SFCA explicitly models the influence of individual actions on future states. It represents an action's effect as the induced difference between future states and quantifies this difference to assign credit to the action. Experimental results demonstrate that SFCA leads to consistent improvements over baselines across diverse environments and achieves particularly significant gains in tasks where baseline performance plateaus.

1 INTRODUCTION

Despite its remarkable success in achieving human-level or even superhuman performance across domains such as games (Silver et al., 2016; 2017), robotics (OpenAI et al., 2019), and large language models (Ouyang et al., 2022), Reinforcement Learning (RL) faces a fundamental and unresolved challenge in *Credit Assignment*. The essence of this problem lies in quantifying the precise contribution of each individual action within a decision sequence to the discounted return, which is a task that remains a core bottleneck in complex environments (Pignatelli et al., 2024). While almost all methods measure these contributions through the advantage function (Sutton & Barto, 2018), most notably via Generalized Advantage Estimation (Schulman et al., 2016). Existing formulations tend to conflate the influence of future actions with that of the current action when estimating returns, primarily through the use of discounted rewards along the trajectory. Accurately disentangling the contribution of an individual action to the discounted return remains a fundamental and largely unresolved challenge.

Recent studies on credit assignment can be broadly categorized into two representative lines of work. (1) *Distributional value function approaches* (Bellemare et al., 2017; Dabney et al., 2018b;a; Yang et al., 2019) model the value or Q function as a full probability distribution rather than a scalar estimation. While such approaches can reduce the noise in return estimation and improve learning stability, they do not explicitly address the underlying causal relationships between actions and delayed rewards. (2) *Future-dependent approaches* (Harutyunyan et al., 2019; Mesnard et al., 2021; Rauber et al., 2019; Andrychowicz et al., 2017) leverage information from future states or outcomes to provide more causally informed credit assignment signals, with information-theoretic perspectives (Arumugam et al., 2021) offering partial theoretical foundation. Despite their empirical success, these future-dependent approaches share a fundamental limitation: they rely on future trajectories, which entangles the causal effect of a specific action with the effects of subsequent actions, as visualized by the black regions in Fig. 1.

We propose a principled approach to overcome this fundamental limitation. We assume that the transformation of the environment state is governed by an unknown function $g(s, a)$ that depends on both the state s and the action a. If we can construct a function $f(s)$ that depends only on the state, then the influence of a specific action can be characterized by the discrepancy between $g(s, a)$

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

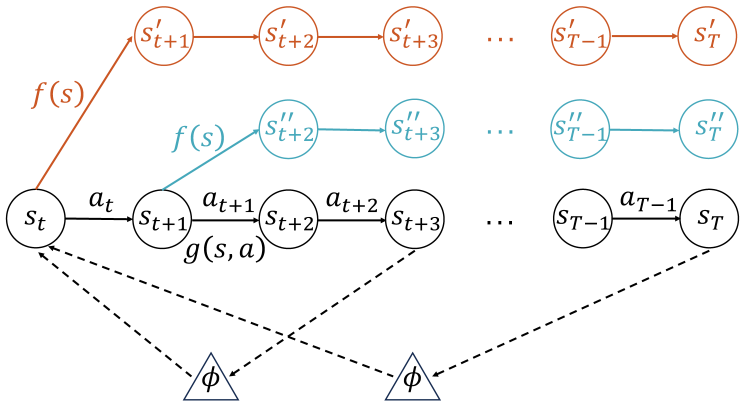


Figure 1: Illustration of future-dependent credit assignment. Prior methods introduce an auxiliary variable ϕ to capture certain relationships (e.g., causal or counterfactual) between the current state s_t and a future state s_T. Since s_T is reached after executing a sequence of actions $(a_t, a_{t+1}, \ldots, a_{T-1})$, its observed value inevitably entangles the effects of all subsequent actions, making it difficult to isolate the specific contribution of a_t. In contrast, if we define a function $f(s)$ that depends only on the current state, the effect of a specific action a_t can be cleanly separated. As shown in the figure, the discrepancy between the blue and orange states directly reflects the influence of action a_t.

and $f(s)$. Following prior work (Schrittwieser et al., 2019; Oh et al., 2017; Hafner et al., 2023), we treat environment dynamics as stochastic factors and absorb their effects into both the $g(s, a)$ and the $f(s)$.

In this work, we adopt a flow-based generative modeling approach, namely Flow Matching (Lipman et al., 2023), to model the probabilistic flow of states, which we term the *State Flow*. Compared to other generative models (Ho et al., 2020; Kingma & Welling, 2014), Flow Matching is chosen for its fast sampling and its theoretically compatible formulation, which aligns naturally with the continuous, distributional nature of state transformation. This State Flow is realized as a vector field that governs the transformation of the state distribution. We treat this vector field as the state-dependent function $f(s)$ and refer to it as the *State Flow vector field (SFVF)*. As illustrated in Fig. 1 (orange and blue regions), the dynamics induced by $f(s)$ and $g(s, a)$ lead to different future states. The discrepancy between these future states reflects the influence induced by a specific action. To transform this discrepancy into an effective learning signal for the policy, we propose two quantification schemes: (1) modeling the reward function $r(s_t, a_t)$ and leveraging the value function $V^\pi(s_t)$ to convert the discrepancy into an advantage-based learning signal; and (2) utilizing $V^\pi(s_t)$ to reformulate the discrepancy as an auxiliary reward, which constitutes a form of potential-based reward shaping (Devlin & Kudenko, 2012). This quantification process defines the mechanism of *Credit Assignment*.

Our main contributions are summarized as follows:

- **A new perspective on credit assignment.** Drawing inspiration from the concept of flow in generative modeling, we propose ***State Flow Credit Assignment*** **(SFCA)** framework. Unlike prior work that relies on future trajectories and thus entangles the effects of individual actions with those of subsequent actions, our framework directly models how an individual action influences future states, enabling the isolation of its effect.

- **Theoretical significance for credit assignment.** We demonstrate that transforming the future state discrepancy into an advantage-based learning signal can be interpreted as an approximation of generalized advantage estimation. Meanwhile, we show that reformulating this discrepancy as an auxiliary reward constitutes a form of potential-based reward shaping, which theoretically guarantees the invariance of the optimal policy.

- **A fully compatible credit assignment mechanism.** We develop a credit assignment mechanism that can be seamlessly integrated into existing reinforcement learning algorithms. It can be incorporated into standard baselines with minimal modifications.

2 Related Work

Temporal Credit Assignment. Temporal credit assignment concerns how delayed rewards should be attributed to preceding decisions in sequential decision-making processes. Building upon the policy gradient framework (Williams, 1992) and the concept of advantage functions (Harmon et al., 1994), Sutton et al. (1999) formalized the use of baselines to reduce gradient variance, laying the foundation for advantage-based formulations that underpin modern policy optimization methods. This line of work was further advanced by Generalized Advantage Estimation (GAE) (Schulman et al., 2016), which provides a low-variance yet controllably biased estimator of the advantage function, thereby stabilizing temporal credit signals during training. GAE has since become a standard component of actor–critic methods (A2C) (Konda & Tsitsiklis, 1999; Mnih et al., 2016) and proximal policy optimization (PPO) (Schulman et al., 2017), serving as a principled mechanism for temporal credit assignment. Although GAE has proven highly effective in practice, it relies purely on statistical estimation and does not explicitly capture the underlying causal structure. As a result, a growing body of subsequent work has emerged to address credit assignment from a more causal perspective.

Future-dependent Credit Assignment. To capture the causal relationship between actions and delayed rewards, prior work has primarily explored hindsight and counterfactual approaches. One of the earliest methods is Hindsight Experience Replay (HER) (Andrychowicz et al., 2017), which assumes that actions within a trajectory lead to the final outcome and learns a correspondence between them. Subsequent methods, including Hindsight Credit Assignment (HCA) (Harutyunyan et al., 2019) and Counterfactual Credit Assignment (CCA) (Mesnard et al., 2021), introduce latent variables to model dependencies between actions and future states. Specifically, HCA aims to capture the causal relationship between an action and a future state through a latent variable, while CCA seeks to model non-causal dependencies between actions and future states.

Despite their differences, HCA and CCA share a common characteristic: both rely on future trajectories and adopt a retrospective modeling perspective. Since future states are influenced by multiple correlated actions, the causal effects of individual actions tend to become entangled, making it difficult to isolate their true contributions.

Flow-based Generative Models. Flow-based generative models learn continuous mappings between probability distributions by modeling the temporal transformation of samples through an ordinary differential equation. Classical approaches such as Continuous Normalizing Flows (Chen et al., 2018) parameterize a time-dependent vector field that induces an invertible transformation, enabling exact likelihood computation and efficient density evaluation. The subsequent advances, including Flow Matching (Lipman et al., 2023) and Rectified Flows (Liu et al., 2023), relax the requirement for explicit likelihood estimation by directly supervising the vector field that transports samples from a source distribution to a target distribution. Particularly, Flow Matching has standed out as an alternative to diffusion models (Dao et al., 2023; Schusterbauer et al., 2024; Polyak et al., 2024) for its high sampling speed and competitive sample quality compared to diffusion models.

3 Preliminaries

3.1 Reinforcement Learning

Reinforcement Learning formalizes sequential decision-making as a Markov Decision Process, defined by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. The transition dynamics $P(s'|s, a)$ specify the probability of transitioning from state s to s' after taking action a, while the reward function $r(s, a)$ provides scalar feedback for each state-action pair. The discount factor $\gamma \in [0, 1)$ controls the relative importance of future rewards.

An agent interacts with the environment according to a policy $\pi_\theta(a|s)$, parameterized by θ, which defines a probability distribution over actions conditioned on the current state. The goal of RL is to find the policy that maximizes the expected discounted return:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t r(s_t, a_t)\right], \tag{1}$$

where $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T)$ denotes a trajectory generated by following π_θ.

Policy gradient methods optimize $J(\theta)$ by computing gradients with respect to the policy parameters:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(a_t|s_t)\, A_t\Big], \tag{2}$$

where A_t is the advantage function, estimating how much better or worse an action is relative to the average action at state s_t.

3.2 Flow Matching

Flow Matching (Lipman et al., 2023) aims to model a continuous transformation from a source distribution p_0 to a target distribution q. Specifically, let $x \in \mathbb{R}^d$ denote a data sample. It define a flow $\phi_\tau : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ that transports the source distribution p_0 to a terminal distribution $p_1 \approx q$. This flow is induced by a time-dependent vector field $v_\tau : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$, which generates ϕ_τ through the following ordinary differential equation (Chen et al., 2018):

$$\frac{d}{d\tau}\phi_\tau(x) = v_\tau(\phi_\tau(x)), \quad \text{with} \quad \phi_0(x) = x. \tag{3}$$

The corresponding probability flow $p_\tau : [0,1] \times \mathbb{R}^d \to \mathbb{R}_{>0}$ is defined via the formula as:

$$p_\tau(x) = p_0\big(\phi_\tau^{-1}(x)\big)\det\left|\frac{\partial \phi_\tau^{-1}(x)}{\partial x}\right|. \tag{4}$$

Flow Matching constructs a target vector field $u_\tau(x)$ that transports p_0 to q along the probability flow:

$$u_\tau(x) = \int u_\tau(x|x_1)\,\frac{p_\tau(x|x_1)\,q(x_1)}{p_\tau(x)}\,dx_1. \tag{5}$$

The goal is to learn a neural vector field $v_\tau(x;\theta)$ parameterized by θ by minimizing the following objective:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\tau,\, x\sim p_\tau}\left\|v_\tau(x;\theta) - u_\tau(x)\right\|^2, \tag{6}$$

where $\tau \sim \mathcal{U}[0,1]$.

However, computing the target vector field $u_\tau(x)$ involves evaluating an intractable integral over the marginal probability path p_τ. To address this issue, Lipman et al. (2023) introduce Conditional Flow Matching (CFM), which defines a conditional probability path $p_\tau(\cdot \mid x_1)$ and a corresponding conditional vector field $u_\tau(\cdot \mid x_1)$ given a terminal sample $x_1 \sim q$. They show that minimizing the following objective:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{\tau,\, x_1\sim q,\, x\sim p_\tau(\cdot|x_1)}\left\|v_\tau(x;\theta) - u_\tau(x \mid x_1)\right\|^2, \tag{7}$$

where $\tau \sim \mathcal{U}[0,1]$, yields the same gradient ∇_θ as Eq. 6.

There exist multiple ways to construct the conditional vector field $u_\tau(\cdot \mid x_1)$, a natural and widely adopted choice is to use an Optimal Transport (OT) path (Lipman et al., 2023). Specifically, given $x_0 \sim p_0$ and $x_1 \sim q$, where p_0 is chosen to be a standard Gaussian distribution, the OT interpolation defines a linear flow:

$$x_\tau = \phi_\tau(x_0, x_1) = (1-\tau)x_0 + \tau x_1, \tag{8}$$

which induces the conditional vector field:

$$u_\tau(x \mid x_1) = x_1 - x_0, \tag{9}$$

and the corresponding conditional probability path:

$$p_\tau(x \mid x_1) = \mathcal{N}\big(x \mid \tau x_1, (1-\tau)^2\mathcal{I}\big). \tag{10}$$

This construction makes the CFM objective (Eq. 7) tractable.

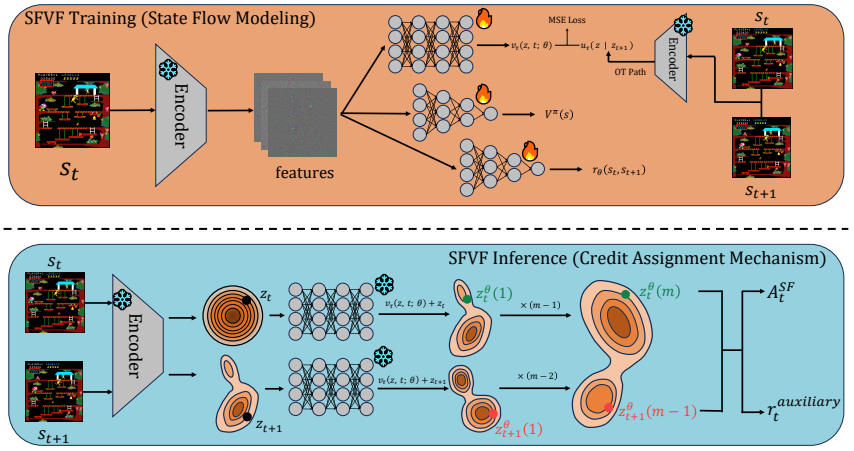


Figure 2: The pipeline of the proposed SFCA. During training, we optimize the SFVF $v_\tau(z, t; \theta)$, the value network $V^\pi(s)$, and the reward network $r_\theta(s_t, s_{t+1})$ using the MSE loss. During inference, we employ the SFVF to generate future latent states $z_t^\theta(m)$ and $z_{t+1}^\theta(m-1)$, and measure their discrepancy using the approximated advantage A_t^{SF} or the auxiliary reward $r_t^{\text{auxiliary}}$ as a credit signal for an action.

4 State Flow Credit Assignment

Prior works (Rauber et al., 2019; Harutyunyan et al., 2019; Andrychowicz et al., 2017; Mesnard et al., 2021) introduce dependencies on future states to capture the causal relationship between individual actions and subsequent states. However, subsequent states are inherently dependent on subsequent actions, which conflates the influence of the current action with that of later decisions.

To overcome this limitation, we firstly model the probability flow of states that is independent of any specific action by defining and optimizing the *State Flow vector field (SFVF)* in Sect. 4.1. Then, in Sect. 4.2, we assign credit to an action by measuring the discrepancy between the future states, which are generated by the previously trained SFVF. The overall pipeline of the proposed SFCA is illustrated in Fig. 2.

4.1 State Flow Modeling

Latent State. We consider an agent following a fixed policy π interacting with an environment. Raw environment states are encoded into latent state vectors through a representation network, such as a CNN, ViT, or MLP. Concretely, a trajectory of raw states $\tau = (s_0, s_1, \ldots, s_T)$ is mapped to a sequence of latent representations:

$$\tau' = (z_0, z_1, \ldots, z_T), \quad \text{where} \quad z_t = \Phi(s_t) \in \mathbb{R}^d$$

and Φ denotes a deterministic encoder.

Objective. The SFVF $v_\tau(z, t; \theta)$ models the transition from current to future latent states, with θ denoting the trainable parameters. Given a transition tuple (z_t, a_t, z_{t+1}), the collection of all possible z_t values defines a distribution p^t, and the set of all possible z_{t+1} values defines a distribution q^{t+1}. We define the source distribution p_0 as the mixture of all p^t and the target distribution q_1 as the mixture of all q^{t+1}. To construct the transformation from p_0 to q_1, we adopt the OT path, which is defined as:

$$z_\tau = (1 - \tau)z_t + \tau z_{t+1}. \tag{11}$$

The corresponding conditional vector field is defined as:

$$u_\tau(z \mid z_{t+1}) = z_{t+1} - z_t, \tag{12}$$

where $\tau \in [0, 1]$. Under this formulation, the CFM objective (Eq. 7) reduces to:

$$\mathcal{L}_{\text{OT}}(\theta) = \mathbb{E}_{\tau, z_t, z_{t+1}} \left\| v_\tau(z, t; \theta) - (z_{t+1} - z_t) \right\|^2, \text{with } z = (1 - \tau)z_t + \tau z_{t+1}, \tag{13}$$

where $\tau \sim \mathcal{U}[0,1]$, $z_t \sim p_0$, $z_{t+1} \sim q_1$ and the scalar t is provided as additional conditioning information to the SFVF $v_\tau(z,t;\theta)$. θ is optimized by minimizing the objective in Eq. 13.

Training Data Collection. The remaining problem lies in how to sample data from p_0 and q_1 to optimize the objective in Eq. 13. In online reinforcement learning, the agent interacts with the environment and collects rollout data, which naturally induces a sampling process. Specifically, given N parallel environments and a trajectory horizon T, each rollout yields $N \times T$ transition tuples

$$\mathcal{D} = \left\{(z_t^n, a_t^n, z_{t+1}^n)\right\}_{t=0,n=0}^{T-1,\,N},$$

which can be directly used to optimize the objective in Eq. 13.

Noise Injection. In practice, for a given transition tuple $(z_t^n, a_t^n, z_{t+1}^n)$, we augment the data by injecting Gaussian noise into the latent state. Specifically, we sample $\epsilon \sim \mathcal{N}(0, \mathcal{I})$ and construct perturbed latent states

$$\tilde{z}_t^n = z_t^n + \epsilon, \tag{14}$$

which yields multiple noisy variants of z_t^n while keeping the corresponding action a_t^n and next state z_{t+1}^n fixed. This augmentation is reasonable, as environments inherently exhibit stochasticity and observation noise.

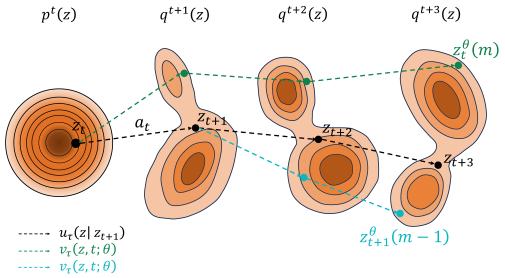


Figure 3: Illustration of latent state transformation under the learned SFVF $v_\tau(z,t;\theta)$. The discrepancy between $z_t^\theta(m)$ and $z_{t+1}^\theta(m-1)$ directly captures the influence of the specific action a_t. $v_\tau(z,t;\theta)$ can be interpreted as the function $f(s)$ illustrated in Fig. 1.

Algorithm 1 Future Latent State Generation

Require: Initial latent state z_t, State Flow vector field $v_\tau(z,t;\theta)$, prediction horizon m, step size $\Delta\tau$
Ensure: Generated latent state $z_t^\theta(m)$

```
z ← z_t
K ← 1/Δτ
for i = 1 to m do
    τ ← 0
    for k = 1 to K do
        v ← v_τ(z, t; θ)
        z ← z + Δτ · v
        τ ← τ + Δτ
    end for
end for
return z
```

4.2 Credit Assignment Mechanism

Difference between Future Latent States. Once the SFVF has been trained by optimizing Eq. 13, it can be utilized to generate future latent states. The process of generating future states is outlined in Alg. 1. Actually, the discrepancy between $u_\tau(z \mid z_{t+1})$ and $u_\tau(z)$ manifests in the difference between future latent states. Each action a_t corresponds to a specific conditional velocity field $u_\tau(z \mid z_{t+1})$, which induces a distinct transformation of the latent state, as illustrated by the black dashed curves in Fig. 3. In contrast, $u_\tau(z)$, which is theoretically independent of any specific z_{t+1}, induces a different transformation of the latent state, corresponding to the blue and green dashed curves in Fig. 3. To this end, the difference between the generated future latent state $z_t^\theta(m)$ and $z_{t+1}^\theta(m-1)$ serves as an effective proxy for isolating the contribution of action a_t. Here, $z_t^\theta(m)$ is denoted as the latent state obtained by generating m steps ahead from time t.

Credit Assignment via Advantage. To facilitate policy optimization, it is essential to quantify the discrepancy between $z_t^\theta(m)$ and $z_{t+1}^\theta(m-1)$. To this end, we introduce two viable quantification schemes, the first of which formulates the discrepancy as an advantage signal, denoted as *SFCA-Adv*. Specifically, we parameterize the reward function using a neural network $r_\theta(s_t, s_{t+1})$. This design choice is drawn from prior works (Fu et al., 2017; Torabi et al., 2018) and $r_\theta(s_t, s_{t+1})$ is jointly trained with the SFVF. Given a transition tuple (z_t, a_t, z_{t+1}), we obtain noisy latent state $\tilde{z}_t \sim p(\tilde{z}_t)$ and $\tilde{z}_{t+1} \sim p(\tilde{z}_{t+1})$ via the noise injection scheme defined in Eq. 14. Meanwhile, the

Algorithm 2 State Flow Credit Assignment (SFCA-Reward)

Require: Policy π_ϕ, value function V_ψ, State Flow vector field $v_\tau(z,t;\theta)$, total training epochs M, trajectory horizon T, prediction horizon m

Randomly initialize parameters ϕ, ψ, θ
for $i = 1$ to M **do**
  Collect a trajectory $\{(s_t, z_t, a_t, r_t)\}_{t=0}^{T}$ using π_ϕ
  Construct training pairs $(z_\tau,\ \tau,\ u_\tau(z \mid z_{t+1}))$ using the OT path
  Add Gaussian noise to z_τ and obtain $\tilde{z}_\tau$
  Predict $v_{\text{pred}} = v_\tau(\tilde{z}_\tau, t; \theta)$ and update v_τ by minimizing the loss
  Compute the auxiliary reward $r_t^{\text{auxiliary}}$
  Augment the environmental reward with $r_t^{\text{auxiliary}}$ and estimate the advantage A_t
  Compute the policy loss using A_t, along with the value loss and entropy loss
  Update π_ϕ and V_ψ
end for

Algorithm 3 State Flow Credit Assignment (SFCA-Adv)

Require: Policy π_ϕ, value function V_ψ, State Flow vector field $v_\tau(z,t;\theta)$, total training epochs M, trajectory horizon T, prediction horizon m

Randomly initialize parameters ϕ, ψ, θ
for $i = 1$ to M **do**
  Collect a trajectory $\{(s_t, z_t, a_t, r_t)\}_{t=0}^{T}$ using π_ϕ
  Construct training pairs $(z_\tau,\ \tau,\ u_\tau(z \mid z_{t+1}))$ using the OT path
  Add Gaussian noise to z_τ and obtain $\tilde{z}_\tau$
  Predict $v_{\text{pred}} = v_\tau(\tilde{z}_\tau, t; \theta)$ and update v_τ by minimizing the loss
  Compute the state-flow-based advantage A_t^{SF}
  Compute the standard advantage A_t
  Compute the policy loss using A_t, the state-flow-based policy loss using A_t^{SF}, along with the value loss and entropy loss
  Update π_ϕ and V_ψ
end for

SFVF $v_\tau(z,t;\theta)$ induces the distribution $p\big(\tilde{z}_t^\theta(m)\big)$ and $p\big(\tilde{z}_{t+1}^\theta(m-1)\big)$ from $p(\tilde{z}_t)$ and $p(\tilde{z}_{t+1})$, respectively. We then define the *state-flow-based advantage* as:

$$A_t^{\text{SF}} = \sum_{t'=0}^{m-1} \gamma^{t'} \mathbb{E}_{\tilde{z}_{t+1}^\theta(t')}\left[r_\theta(\tilde{z}_{t+1}^\theta(t'-1), \tilde{z}_{t+1}^\theta(t'))\right] - \sum_{t'=0}^{m-1} \gamma^{t'} \mathbb{E}_{\tilde{z}_t^\theta(t')}\left[r_\theta(\tilde{z}_t^\theta(t'), \tilde{z}_t^\theta(t'+1))\right] + \gamma^m \big(\mathbb{E}_{\tilde{z}_{t+1}^\theta(m-1)}\left[V^\pi\big(\tilde{z}_{t+1}^\theta(m-1)\big)\right] - \mathbb{E}_{\tilde{z}_t^\theta(m)}\left[V^\pi\big(\tilde{z}_t^\theta(m)\big)\right]\big), \tag{15}$$

where $\tilde{z}_{t+1}^\theta(-1) = \tilde{z}_t$, $\tilde{z}_{t+1}^\theta(m-1) \sim p\big(\tilde{z}_{t+1}^\theta(m-1)\big)$, $\tilde{z}_t^\theta(m) \sim p\big(\tilde{z}_t^\theta(m)\big)$, and V^π denotes the value function under policy π.

We claim that A_t^{SF} provides an approximate computation of the generalized advantage function:

$$A_t^\pi(s,a) = Q_t^\pi(s,a) - V_t^\pi(s) \approx A_t^{\text{SF}}. \tag{16}$$

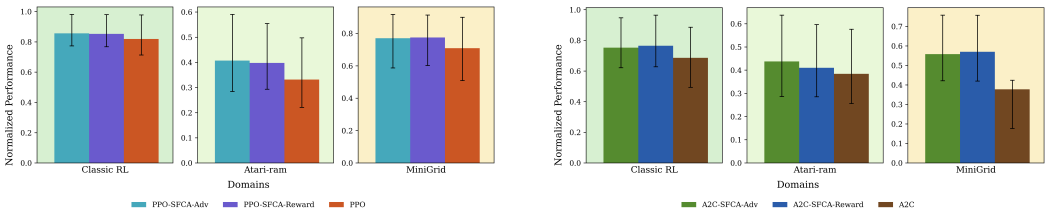
A complete derivation is provided in Appendix A.

Credit Assignment via Reward Shaping. The second quantification scheme leverages the framework of potential-based reward shaping (Devlin & Kudenko, 2012), denoted as *SFCA-Reward*. Specifically, for each action a_t, we compute an auxiliary reward as follows:

$$r_t^{\text{auxiliary}} = \gamma \cdot \mathbb{E}_{\tilde{z}_{t+1}^\theta(m)}\left[V^\pi\big(\tilde{z}_{t+1}^\theta(m)\big)\right] - \mathbb{E}_{\tilde{z}_t^\theta(m)}\left[V^\pi\big(\tilde{z}_t^\theta(m)\big)\right]. \tag{17}$$

Since both the SFVF and V^π are parameterized as neural networks, Eq. 17 conforms to the principled form of potential-based reward shaping, i.e., $F(s_t, a_t, s_{t+1}) = \gamma\Phi(s_{t+1}) - \Phi(s_t)$. Consequently, this formulation is theoretically guaranteed to preserve the optimal policy.

Summary. We first demonstrate that the SFVF can be utilized to generate future latent states and illustrate how the SFVF predictions and specific actions a_t result in a discrepancy within the future latent space, as shown in Fig. 3. Subsequently, we propose two viable quantification mechanisms to transform this discrepancy into learning signals for policy optimization: the first serves as an approximation of a generalized advantage function, while the second is formulated as a potential-based reward shaping framework. The pseudocode for our proposed methods is provided in Alg. 2 and Alg. 3.



(a) PPO-SFCA vs PPO

(b) A2C-SFCA vs A2C

Figure 4: PPO-SFCA vs. PPO and A2C-SFCA vs. A2C in three type environments. Aggregated mean and standard deviation of the normalized average return for the final 100 episodes.

4.3 Implementation Details

The performance of our method depends on the accuracy of both the SFVF and the value function. To enhance stability and effectiveness, we incorporate several practical techniques as described below.

Experience Replay. We store observations and next observations from the most recent $E = 16$ rollouts to update the SFVF. We find that training with some previous interaction data significantly benefits the learning of SFVF. Hence, we employ experience replay (Schaul et al., 2016).

Target Network. The computation of $u_\tau(z \mid z_{t+1})$ depends on the deterministic encoder Φ. However, updating the policy continually alters the encoder parameters, which can lead to instability when the neural network attempts to learn from a shifting target. To mitigate this issue, we introduce a target network (van Hasselt et al., 2016): after a fixed number of updates, the encoder parameters of the current policy are synchronized to the target encoder. By default, we perform an update every 1% $(syn\,rate)$ of the total training progress.

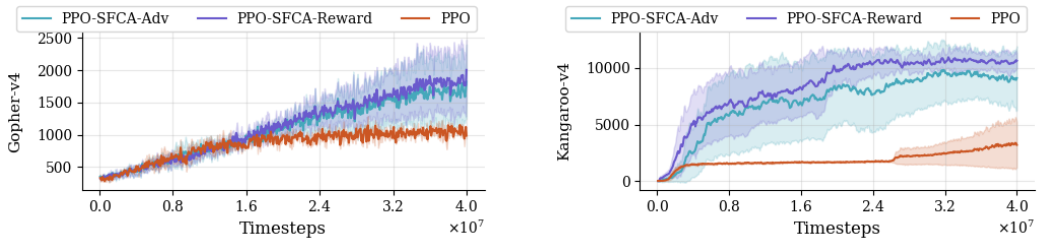
5 Experiments

Experimental Setup. In our experiments, we integrated the proposed **SFCA** into two important on-policy reinforcement learning algorithms: PPO (Schulman et al., 2017) and A2C (Mnih et al., 2016), thus resulting in **PPO-SFCA-Adv**, **PPO-SFCA-Reward**, **A2C-SFCA-Adv** and **A2C-SFCA-Reward**. For brevity, we collectively refer to PPO-SFCA and A2C-SFCA. These implementations were built upon the `Stable-Baselines3` (Raffin et al., 2021) framework. All training and evaluation were conducted using the standardized and reproducible environment provided by the `RL Baselines3 Zoo` (Raffin, 2020) infrastructure. All results were averaged over five independent runs with distinct random seeds to ensure statistical robustness. Complete training configurations, hyperparameters and other experimental details are reported in Appendix C.

Evaluation Benchmark. We first conducted experiments on classic reinforcement learning tasks using the `Classic-Control` and `Box2D` environments from `Gymnasium` (Towers et al., 2024; Brockman et al., 2016), which serve as standard benchmarks for comparison. Subsequently, we evaluated our approach on more complex high-dimensional tasks within the `Atari RAM` (Bellemare et al., 2012) environments. Finally, we assessed performance on categorical and partially observable environments, specifically focusing on the `MiniGrid` (Chevalier-Boisvert et al., 2023) domain.

Experimental Results. In Fig. 4, we report the normalized performance on three types of environments, including Classic RL, Atari-ram and MiniGrid. Each result is obtained by averaging the returns over the last 100 episodes of all experiments. To facilitate fair comparison across different environments, the returns are normalized. For example, the normalized return of PPO-SFCA is computed as

$$r_{\text{norm}} = \frac{r_{\text{mean}}^{\text{PPO-SFCA}} - r_{\text{min}}^{\{\text{PPO-SFCA, PPO}\}}}{r_{\text{max}}^{\{\text{PPO-SFCA, PPO}\}} - r_{\text{min}}^{\{\text{PPO-SFCA, PPO}\}}}.$$



(a) Gopher-v4 environment

(b) Kangaroo-v4 environment

Figure 5: Training curves of PPO-SFCA and PPO on Gopher-v4 and Kangaroo-v4 environments. PPO-SFCA achieves higher episode rewards, while PPO plateaus. Reported values are the mean and standard deviation of the entire training process, aggregated across 5 random seeds.

As a credit assignment method based on generative modeling, SFCA achieves performance improvements across a wide range of environments. These gains primarily arise from its ability to infer more accurate credit for individual actions from interaction data. In particular, when baseline algorithms reach performance plateaus and fail to make further progress, SFCA is able to find effective directions for improvement. For example, in the Gopher and Kangaroo environments, the episode reward of the baseline PPO algorithm largely converge, whereas PPO-SFCA continues to improve performance by discovering more effective credit assignments, as illustrated in Fig. 5.

Ablation Study. In this study, we investigate several key components of our proposed method, including the synchronization ratio ($syn\ rate$), the number of prediction horizon (m), and the size of the experience buffer (E). For each ablation study, we vary one component while keeping all others fixed, thereby isolating its individual contribution and identifying its optimal configuration. All experiments are conducted on the `CartPole-v1` environment using the PPO-SFCA-Adv algorithm for a total of 300k training steps. We find that a better-trained encoder, an appropriate prediction horizon, and a suitable size of experience replay buffer collectively contribute to policy optimization. Comprehensive ablation studies are provided in Appendix B.

6 Limitation

The proposed SFCA framework demonstrates competitive performance across a variety of environments, combining conceptual simplicity with strong empirical results. However, SFCA does not currently perform explicit dynamics modeling, we treat dynamics as exogenous stochasticity as most works do. From an optimization perspective, the influence of dynamics is implicitly captured within the expectation terms during the optimization of the SFVF. Specifically, the loss calculation for $v_\tau(z, t; \theta)$ (as shown in Eq. 13) accounts for the underlying dynamics by taking expectations over all pairs of (z_t, z_{t+1}).

7 Conclusion

In this work, we integrate generative modeling into the credit assignment process and propose State Flow Credit Assignment. The proposed framework comprises two core components: State Flow Modeling and Credit Assignment. Specifically, State Flow Modeling characterizes the influence of an action by learning a State Flow Vector Field, which maps actions to discrepancy in future latent states. Subsequently, the Credit Assignment phase introduces two viable mechanisms to quantify these discrepancies: generalized advantage approximation and potential-based reward shaping. These mechanisms provide informative learning signals, thereby providing guidance for the policy optimization process.

In future work, we plan to extend the SFCA framework to off-policy and offline settings. We believe this is a highly promising direction, as it would enable SFCA to leverage vast offline datasets, thereby unlocking superior performance and broader applicability.

REFERENCES

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Proc. of NeurIPS*, pp. 5048–5058, 2017.

Dilip Arumugam, Peter Henderson, and Pierre-Luc Bacon. An information-theoretic perspective on credit assignment in reinforcement learning. *arXiv preprint arxiv:2103.06224*, 2021.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *arXiv preprint arxiv:1207.4708*, 2012.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proc. of ICML*, pp. 449–458, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arxiv:1606.01540*, 2016.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proc. of NeurIPS*, pp. 6572–6583, 2018.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan K. Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Proc. of NeurIPS*, 2023.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *Proc. of ICML*, pp. 1104–1113, 2018a.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proc. of AAAI*, pp. 2892–2901, 2018b.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arxiv:2307.08698*, 2023.

Sam Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *Proc. of AAMAS*, pp. 433–440, 2012.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arxiv:1710.11248*, 2017.

Danijar Hafner, J. Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arxiv:2301.04104*, 2023.

Mance E. Harmon, Leemon C. Baird III, and A. Harry Klopf. Advantage updating applied to a differrential game. In *Proc. of NeurIPS*, pp. 353–360, 1994.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, and Rémi Munos. Hindsight credit assignment. In *Proc. of NeurIPS*, pp. 12467–12476, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proc. of NeurIPS*, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. of ICLR*, 2014.

Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *Proc. of NeurIPS*, pp. 1008–1014, 1999.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *Proc. of ICLR*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *Proc. of ICLR*, 2023.

Thomas Mesnard, Theophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Thomas S. Stepleton, Nicolas Heess, Arthur Guez, Eric Moulines, Marcus Hutter, Lars Buesing, and Rémi Munos. Counterfactual credit assignment in model-free reinforcement learning. In *Proc. of ICML*, pp. 7654–7664, 2021.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proc. of ICML*, pp. 1928–1937, 2016.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *Proc. of NeurIPS*, 2017.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *arXiv preprint arxiv:1910.07113*, 2019.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proc. of NeurIPS*, 2022.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *Trans. Mach. Learn. Res.*, 2024, 2024.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam S. Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali K. Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dmitry Vengertsev, Edgar Schönfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie gen: A cast of media foundation models. *arXiv preprint arxiv:2410.13720*, 2024.

Antonin Raffin. Rl baselines3 zoo, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.*, 22:268:1–268:8, 2021.

Paulo E. Rauber, Avinash Ummadisingu, Filipe Mutz, and Jürgen Schmidhuber. Hindsight policy gradients. In *Proc. of ICLR*, 2019.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *Proc. of ICLR*, 2016.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, L. Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588:604 – 609, 2019.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proc. of ICLR*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arxiv:1707.06347*, 2017.

Johannes Schusterbauer, Ming Gui, Pingchuan Ma, Nick Stracke, Stefan Andreas Baumann, Vincent Tao Hu, and Björn Ommer. Fmboost: Boosting latent diffusion with flow matching. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Proc. of ECCV*, volume 15119, pp. 338–355, 2024.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arxiv:1712.01815*, 2017.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction, 2nd Edition*. MIT Press, 2nd edition, 2018.

Richard S. Sutton, David A. McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proc. of NeurIPS*, pp. 1057–1063, 1999.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arxiv:1807.06158*, 2018.

Mark Towers, Ariel Kwiatkowski, Jordan K. Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arxiv:2407.17032*, 2024.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proc. of AAAI*, pp. 2094–2100, 2016.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992.

Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tie-Yan Liu. Fully parameterized quantile function for distributional reinforcement learning. In *Proc. of NeurIPS*, pp. 6190–6199, 2019.

A Theoretical Derivation

In this section, we provide a rigorous mathematical derivation to establish that the proposed state-flow-based advantage function serves as a principled approximation of the generalized advantage. We show how the State Flow vector field facilitates credit assignment by matching future state distributions in the latent space.

A.1 Generalized Advantage Function

We denote $d_t^\pi(s \mid s_0)$ as the probability distribution of states at time t induced by policy π starting from state s_0. Similarly, $d_t^\pi(s \mid s_0, a_0)$ represents the distribution starting from s_0 and taking an initial action a_0. Let $\gamma \in (0, 1)$ be the discount factor and T be the trajectory horizon. For any

$0 < m < T$, the value function $V^\pi(s)$ can be decomposed into an m-step discounted reward sum and a discounted value:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t r_t \;\middle|\; s_0 = s\right] \\ &= \sum_{t=0}^{m-1}\gamma^t \mathbb{E}_{s_t\sim d_t^\pi(\cdot|s)}[r_t] + \mathbb{E}_{s_m\sim d_m^\pi(\cdot|s)}[\gamma^m V^\pi(s_m)]. \end{aligned} \tag{18}$$

Analogously, the action-value function $Q^\pi(s,a)$ is expressed as:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t r_t \;\middle|\; s_0 = s, a_0 = a\right] \\ &= \sum_{t=0}^{m-1}\gamma^t \mathbb{E}_{s_t\sim d_t^\pi(\cdot|s,a)}[r_t] + \mathbb{E}_{s_m\sim d_m^\pi(\cdot|s,a)}[\gamma^m V^\pi(s_m)]. \end{aligned} \tag{19}$$

By the definition of the advantage function $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, we obtain:

$$\begin{aligned} A^\pi(s,a) = &\sum_{t=0}^{m-1}\gamma^t\left(\mathbb{E}_{s_t\sim d_t^\pi(\cdot|s,a)}[r_t] - \mathbb{E}_{s_t\sim d_t^\pi(\cdot|s)}[r_t]\right) \\ &+ \gamma^m\left(\mathbb{E}_{s_m\sim d_m^\pi(\cdot|s,a)}[V^\pi(s_m)] - \mathbb{E}_{s_m\sim d_m^\pi(\cdot|s)}[V^\pi(s_m)]\right). \end{aligned} \tag{20}$$

A.2 State Distribution Approximation via State Flow

Let Φ be a deterministic encoder mapping states to a latent space $\mathcal{Z}$, such that $z = \Phi(s)$. We define $v_\tau(z,t;\theta)$ as the State Flow vector field parameterized by θ. Under the deterministic mapping Φ, the distributions $d_t^\pi(s \mid \cdot)$ correspond uniquely to $d_t^\pi(z \mid \cdot)$ in $\mathcal{Z}$. Thus, the advantage function in Eq. equation 20 can be reformulated in the latent space as:

$$\begin{aligned} A^\pi(s,a) = &\sum_{t'=0}^{m-1}\gamma^{t'}\left(\mathbb{E}_{z_{t'}\sim d_{t'}^\pi(\cdot|z,a)}[r_{t'}] - \mathbb{E}_{z_{t'}\sim d_{t'}^\pi(\cdot|z)}[r_{t'}]\right) \text{ (\textit{reward difference})} \\ &+ \gamma^m\left(\mathbb{E}_{z_m\sim d_m^\pi(\cdot|z,a)}[V^\pi(z_m)] - \mathbb{E}_{z_m\sim d_m^\pi(\cdot|z)}[V^\pi(z_m)]\right) \text{ (\textit{value difference}).} \end{aligned} \tag{21}$$

To compute this, we leverage the State Flow vector field $v_\tau(z,t;\theta)$. Let $z^\theta(m)$ be the latent state generated by integrating $v_\tau(z,t;\theta)$ for m steps. Following the flow matching paradigm (Lipman et al., 2023), the trained model induces distributions such that:

$$p(\tilde{z}_t^\theta(1)) \approx p(z_{t+1} \mid z_t) = d_1^\pi(\cdot \mid z_t). \tag{22}$$

By composition, the m-step distribution is approximated by $p(\tilde{z}_t^\theta(m)) \approx d_m^\pi(\cdot \mid z_t)$. Given the noise injection scheme (Eq. 14) and the observed Markovian transition, we assume that $d_1^\pi(\cdot \mid z_t, a_t) = p(\tilde{z}_{t+1})$, and further obtain:

$$p(\tilde{z}_{t+1}^\theta(m-1)) \approx p(z_{t+m} \mid z_{t+1}) = d_m^\pi(\cdot \mid z_t, a_t). \tag{23}$$

Substituting these approximations into the reward and value components of Eq. equation 21, and modeling rewards via $r_\theta(z_t, z_{t+1})$, the reward difference is approximated as:

$$\begin{aligned} \Delta R \approx &\sum_{t'=0}^{m-1}\gamma^{t'}\mathbb{E}_{\tilde{z}_{t+1}^\theta(t')}[r_\theta(\tilde{z}_{t+1}^\theta(t'-1), \tilde{z}_{t+1}^\theta(t'))] \\ &- \sum_{t'=0}^{m-1}\gamma^{t'}\mathbb{E}_{\tilde{z}_t^\theta(t')}[r_\theta(\tilde{z}_t^\theta(t'), \tilde{z}_t^\theta(t'+1))], \end{aligned} \tag{24}$$

and the value difference becomes:

$$\Delta V \approx \gamma^m\left(\mathbb{E}_{\tilde{z}_{t+1}^\theta(m-1)}[V^\pi(\tilde{z}_{t+1}^\theta(m-1))] - \mathbb{E}_{\tilde{z}_t^\theta(m)}[V^\pi(\tilde{z}_t^\theta(m))]\right). \tag{25}$$

Combining Eq. equation 24 and Eq. equation 25, we arrive at the state-flow-based advantage A_t^{SF}:

$$\begin{aligned} A_t^{\pi}(s,a) \approx & \sum_{t'=0}^{m-1} \gamma^{t'} \mathbb{E}_{\tilde{z}_{t+1}^{\theta}(t')} \left[r_\theta(\tilde{z}_{t+1}^{\theta}(t'-1), \tilde{z}_{t+1}^{\theta}(t')) \right] - \sum_{t'=0}^{m-1} \gamma^{t'} \mathbb{E}_{\tilde{z}_{t}^{\theta}(t')} \left[r_\theta(\tilde{z}_{t}^{\theta}(t'), \tilde{z}_{t}^{\theta}(t'+1)) \right] \\ & + \gamma^m \left(\mathbb{E}_{\tilde{z}_{t+1}^{\theta}(m-1)}[V^{\pi}(\tilde{z}_{t+1}^{\theta}(m-1))] - \mathbb{E}_{\tilde{z}_{t}^{\theta}(m)}[V^{\pi}(\tilde{z}_{t}^{\theta}(m))] \right) \\ = & A_t^{\mathrm{SF}}, \end{aligned} \tag{26}$$

where $\tilde{z}_{t+1}^{\theta}(-1) = \tilde{z}_t$. This confirms that A_t^{SF} is a direct approximation of the generalized advantage through latent state distribution matching.

B Ablation Study

In this study, we investigate several key components of our proposed method, including the synchronization ratio ($syn\ rate$), the number of prediction horizon (m), and the size of the experience buffer (E). For each ablation study, we vary one component while keeping all others fixed, thereby isolating its individual contribution and identifying its optimal configuration. All experiments are conducted on the `CartPole-v1` environment using the PPO-SFCA-Adv algorithm for a total of 300k training steps.

Effect of Different Synchronization Ratios. We employ the latent state z obtained by encoding the state through the encoder. As the synchronization ratio increases, both the performance and stability of the algorithm exhibit a clear degradation trend, as shown in Fig. 6. We attribute this behavior in part to the quality of the encoder: a better trained encoder tends to produce more discriminative latent state, thereby facilitating the SFVF learning and improving credit assignment accuracy.

Choosing an Appropriate Prediction Horizon. Due to the potentially large effective horizon of the Markov Decision Process, uncertainty accumulates over long temporal depths and gradually decreases as the trajectory approaches terminal states. As a result, predicting further into the future provides more reliable signals for credit assignment, leading to more stable learning. However, increasing the prediction horizon also makes network optimization more challenging, requiring additional training data and computational resources. As illustrated in Fig. 7, the algorithm achieves better performance when the prediction horizon is set to 2, 4, 6, or 8, compared to horizons of 1 or 10. Moreover, in complex environments (e.g., Robotics or real-world scenarios), accurately predicting long-term futures becomes inherently difficult. Therefore, we hypothesize that for relatively simple environments, a larger prediction horizon can be beneficial, while for increasingly complex environments, a shorter prediction horizon is more appropriate.

Impact of Experience Replay Size on Performance. Choosing an appropriate experience replay size is a non-trivial design decision. A replay buffer that is too small provides insufficient coverage for modeling the SFVF, whereas an excessively large buffer may introduce distributional shift due to stale data. As the experience replay buffer size increases, the algorithm's performance improves initially; however, an excessively large buffer leads to a performance degradation, as illustrated in Fig. 8.

Overall, the ablation results suggest that the performance of SFCA is sensitive to the synchronization ratio, the number of prediction horizon, and the size of the experience buffer. We do not include ablations on the sampling strategy of the time steps τ, as these design choices are not central to the core ideas of SFCA.

C Experimental Details

We report the complete set of hyperparameters for the proposed **PPO-SFCA** and **A2C-SFCA** in Tab. 1. For quantitative comparison, we report the average episode reward over the final 100 training episodes in Tab. 2, 3, 4. In addition, training curves over the entire training process are presented in Fig. 9, 10, 11, 12, 13. All experiments were conducted on a single NVIDIA GeForce RTX 4090 GPU, with each individual run taking a maximum of 14 hours.

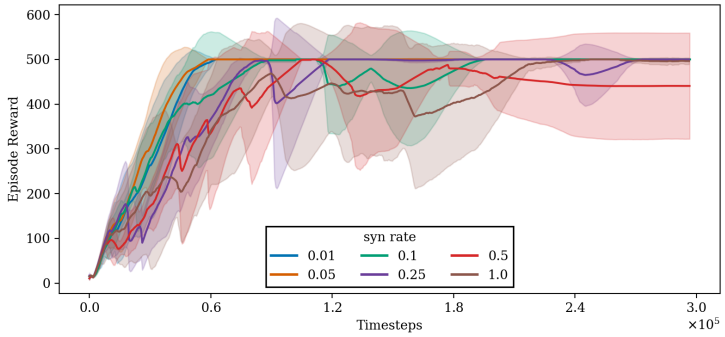


Figure 6: **Ablation: Effect of Different Synchronization Ratios.** Mean and standard deviation of the average episodic reward during training. All other hyperparameters are kept fixed. As the synchronization ratio increases, both performance and training stability consistently degrade.

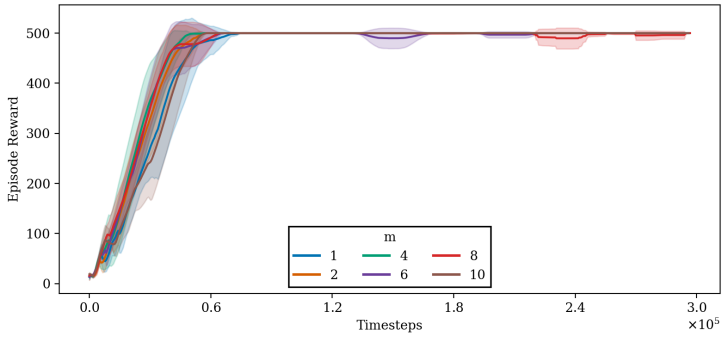


Figure 7: **Ablation: Choosing an Appropriate Prediction Horizon.** Mean and standard deviation of the average episodic reward during training. All other hyperparameters are kept fixed. We observe that predicting a larger and appropriate future steps leads to improved training stability.

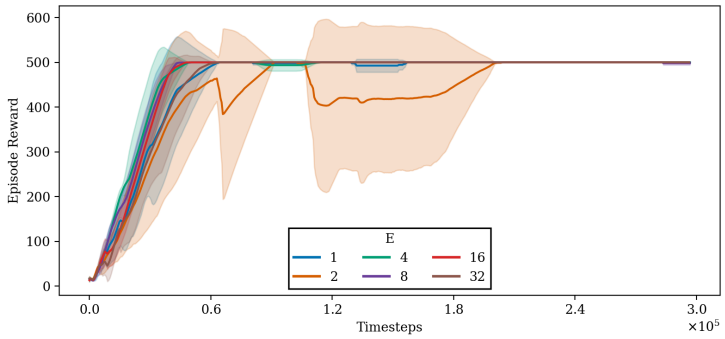


Figure 8: **Ablation: Impact of Experience Replay Size on Performance.** Mean and standard deviation of the average episodic reward during training. All other hyperparameters are kept fixed. We observe improved performance with a larger and appropriate experience replay size.

Table 1: **SFCA Hyperparameters** for each environment. SFCA introduces three additional hyperparameters.

(a) PPO-SFCA hyperparameters.

	Core Hyperparameters									
Env	batch size	clip range	ent coef	gae λ	γ	epochs	steps	envs	policy lr	value lr
Acrobot	4096	0.2	0.0	0.94	0.99	4	256	16	3e-4	3e-4
CartPole	256	0.2	0.0	0.8	0.98	20	32	8	0.001	0.001
LunarLander	64	0.2	0.0	0.98	0.999	4	1024	16	3e-4	3e-4
MountainCar	256	0.2	0.0	0.98	0.99	4	16	16	3e-4	3e-4
Pendulum	4096	0.2	0.0	0.95	0.9	10	1024	4	1e-3	1e-3
Atari-Ram	256	0.1	0.01	0.95	0.99	4	128	8	2.5e-4	2.5e-4
MiniGrid	64	0.2	0.0	0.95	0.99	10	128	8	2.5e-4	2.5e-4

	Additional Hyperparameters		
Env	m (prediction horizon)	E (Experience Replay Size)	syn rate
Acrobot	10	16	0.01
CartPole	10	16	0.01
LunarLander	6	16	0.01
MountainCar	6	16	0.01
Pendulum	10	16	0.01
Atari-Ram	1	16	0.01
MiniGrid	6	16	0.01

(b) A2C-SFCA hyperparameters.

	Core Hyperparameters								
Env	batch size	ent coef	gae λ	γ	epochs	steps	envs	policy lr	value lr
Acrobot	80	0.0	1.0	0.99	1	5	16	7e-4	7e-4
CartPole	40	0.0	1.0	0.99	1	5	8	7e-4	7e-4
LunarLander	40	1e-5	1.0	0.995	1	5	8	8e-4	8e-4
MountainCar	80	0.0	1.0	0.99	1	5	16	7e-4	7e-4
Pendulum	40	0.0	1.0	0.99	1	5	8	7e-4	7e-4
Atari-Ram	80	0.01	1.0	0.99	1	5	16	7e-4	7e-4
MiniGrid	40	0.0	0.95	0.99	1	5	8	2.5e-4	2.5e-4

	Additional Hyperparameters		
Env	m (prediction horizon)	E (Experience Replay Size)	syn rate
Acrobot	10	32	0.01
CartPole	10	32	0.01
LunarLander	6	32	0.01
MountainCar	6	32	0.01
Pendulum	10	32	0.01
Atari-Ram	1	32	0.01
MiniGrid	6	32	0.01

Table 2: Continuous-Control and Box2D environments: Reported values are the mean and standard deviation of the average episode reward over the final 100 training episodes, aggregated across 5 random seeds.

	Acrobot	CartPole	LunarLander
PPO	-78.84 ± 22.97	499.91 ± 1.48	236.34 ± 65.07
PPO-SFCA-Adv	-75.47 ± 21.94	499.30 ± 18.39	**250.81 ± 48.72**
PPO-SFCA-Reward	-76.06 ± 18.75	498.48 ± 16.15	**249.90 ± 64.03**
A2C	-82.84 ± 21.90	500.00 ± 0.0	43.84 ± 119.75
A2C-SFCA-Adv	-84.09 ± 28.28	482.55 ± 52.02	**73.68 ± 121.60**
A2C-SFCA-Reward	-82.28 ± 21.90	496.99 ± 20.50	**76.70 ± 106.95**

	MountainCar	Pendulum-v1
PPO	-115.99 ± 21.54	-193.68 ± 126.14
PPO-SFCA-Adv	**-106.50 ± 19.27**	**-162.65 ± 99.98**
PPO-SFCA-Reward	**-106.15 ± 18.96**	**-173.15 ± 109.81**
A2C	-116.53 ± 26.38	-319.88 ± 372.14
A2C-SFCA-Adv	-114.04 ± 24.51	**-159.52 ± 94.75**
A2C-SFCA-Reward	-112.56 ± 23.50	**-162.16 ± 99.84**

Table 3: Atari-ram-v4 environments: Reported values are the mean and standard deviation of the average episode reward over the final 100 training episodes, aggregated across 5 random seeds.

	Alien	Amidar	Asteroids	Breakout
PPO	1670.60 ± 645.10	1179.04 ± 326.66	1031.05 ± 314.74	27.80 ± 16.82
PPO-SFCA-Adv	1540.90 ± 532.80	1206.84 ± 388.84	998.50 ± 323.51	25.87 ± 9.56
PPO-SFCA-Reward	**1870.95 ± 681.41**	839.83 ± 490.56	1025.35 ± 421.50	26.55 ± 16.63
A2C	1252.50 ± 528.11	907.78 ± 200.51	693.25 ± 300.85	26.64 ± 12.25
A2C-SFCA-Adv	**1672.13 ± 679.10**	854.25 ± 205.10	729.95 ± 284.57	25.39 ± 9.91
A2C-SFCA-Reward	1476.20 ± 552.13	902.74 ± 218.02	698.17 ± 305.40	26.42 ± 11.43

	Gopher	Kangaroo	Krull	Pong
PPO	1046.30 ± 530.18	3365.00 ± 2374.93	6611.70 ± 769.89	-6.14 ± 6.68
PPO-SFCA-Adv	**1601.27 ± 833.08**	**8957.17 ± 3462.25**	6510.13 ± 709.93	**0.33 ± 7.22**
PPO-SFCA-Reward	**1730.70 ± 921.29**	**10477.80 ± 2159.95**	6661.13 ± 629.17	-7.37 ± 8.01
A2C	1035.87 ± 686.76	1402.00 ± 344.67	5918.85 ± 649.22	-2.32 ± 7.37
A2C-SFCA-Adv	973.90 ± 486.65	1480.67 ± 299.38	5663.37 ± 646.78	**13.61 ± 9.19**
A2C-SFCA-Reward	**1341.70 ± 975.64**	1535.00 ± 310.44	5827.50 ± 674.32	-5.56 ± 6.97

	MsPacman	SpaceInvaders
PPO	2391.70 ± 590.83	1057.72 ± 453.68
PPO-SFCA-Adv	**2924.97 ± 873.48**	1051.57 ± 442.58
PPO-SFCA-Reward	**3229.10 ± 732.45**	972.72 ± 450.43
A2C	2942.00 ± 833.00	902.17 ± 418.18
A2C-SFCA-Adv	**3593.90 ± 1369.80**	831.52 ± 357.24
A2C-SFCA-Reward	**3659.05 ± 1426.15**	1038.65 ± 419.74

Table 4: MiniGrid environments: Reported values are the mean and standard deviation of the average episode reward over the final 100 training episodes, aggregated across 5 random seeds.

	DoorKey-5x5	Unlock	Fetch-5x5-N2	FourRooms	GoToDoor-5x5
PPO	0.96 ± 0.02	0.95 ± 0.03	0.92 ± 0.21	0.55 ± 0.33	0.95 ± 0.05
PPO-SFCA-Adv	0.95 ± 0.04	0.96 ± 0.02	0.94 ± 0.15	0.44 ± 0.38	0.94 ± 0.10
PPO-SFCA-Reward	0.96 ± 0.04	0.96 ± 0.02	0.94 ± 0.15	0.43 ± 0.39	0.95 ± 0.07
A2C	0.95 ± 0.05	0.41 ± 0.34	0.33 ± 0.46	0.32 ± 0.39	0.05 ± 0.21
A2C-SFCA-Adv	0.96 ± 0.01	0.36 ± 0.35	$\mathbf{0.54 \pm 0.48}$	$\mathbf{0.40 \pm 0.39}$	$\mathbf{0.24 \pm 0.41}$
A2C-SFCA-Reward	0.93 ± 0.11	0.35 ± 0.36	$\mathbf{0.63 \pm 0.46}$	$\mathbf{0.39 \pm 0.39}$	$\mathbf{0.33 \pm 0.45}$

	KeyCorridorS3R1	PutNear-6x6-N2	RedBlueDoors-6x6	Empty-Random-5x5
PPO	0.76 ± 0.38	0.26 ± 0.35	0.33 ± 0.46	0.97 ± 0.01
PPO-SFCA-Adv	0.79 ± 0.35	$\mathbf{0.54 \pm 0.33}$	$\mathbf{0.64 \pm 0.45}$	0.96 ± 0.02
PPO-SFCA-Reward	0.76 ± 0.38	$\mathbf{0.53 \pm 0.34}$	$\mathbf{0.69 \pm 0.42}$	0.96 ± 0.02
A2C	0.76 ± 0.38	0.14 ± 0.30	0.41 ± 0.48	0.96 ± 0.02
A2C-SFCA-Adv	$\mathbf{0.95 \pm 0.00}$	$\mathbf{0.43 \pm 0.36}$	$\mathbf{0.49 \pm 0.48}$	0.96 ± 0.02
A2C-SFCA-Reward	$\mathbf{0.95 \pm 0.00}$	$\mathbf{0.44 \pm 0.36}$	$\mathbf{0.53 \pm 0.48}$	0.97 ± 0.01

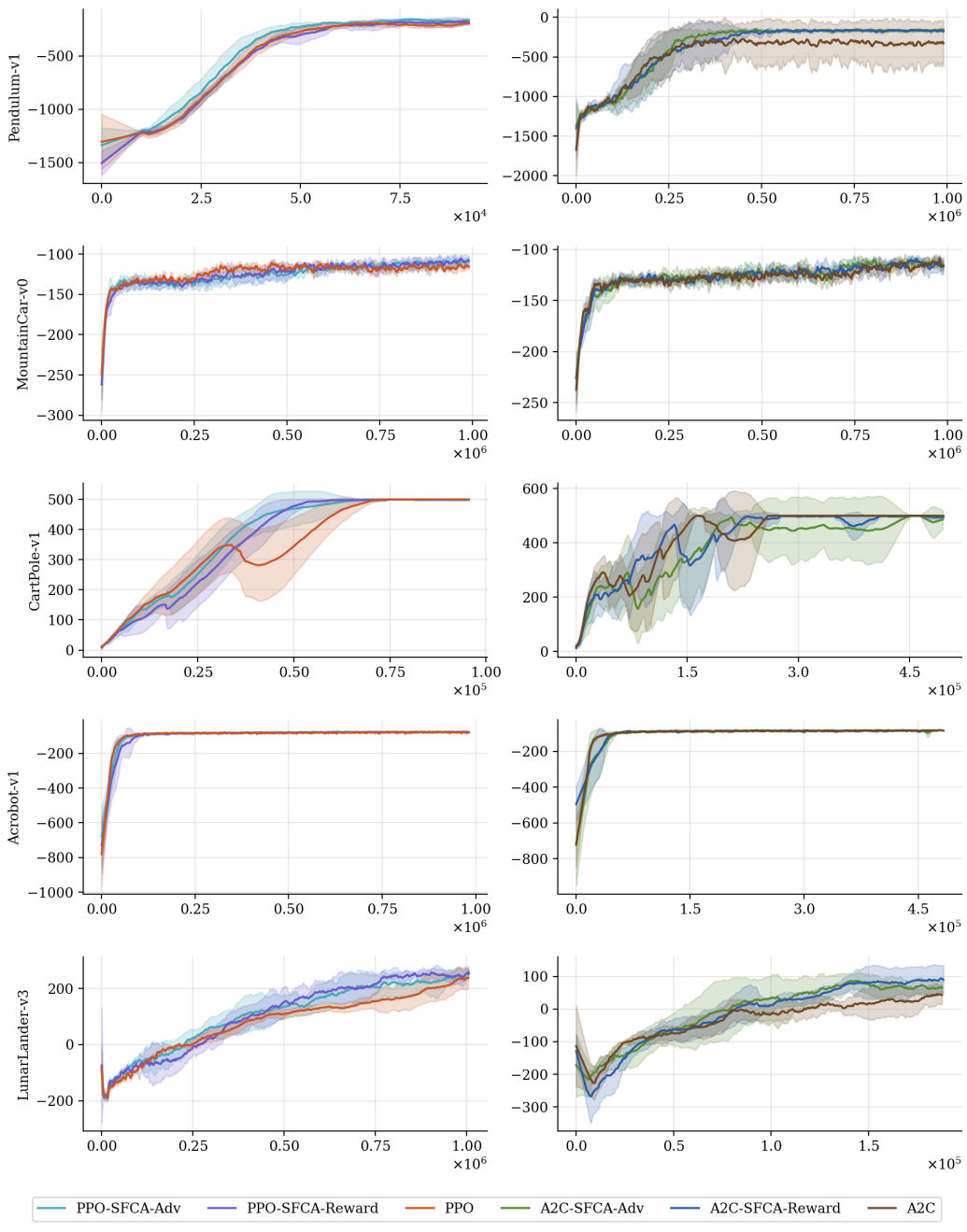


Figure 9: Classic Control and Box2D environments: Training reward as a function of environment steps. Reported values are the mean and standard deviation of the average episode reward, aggregated across 5 random seeds.

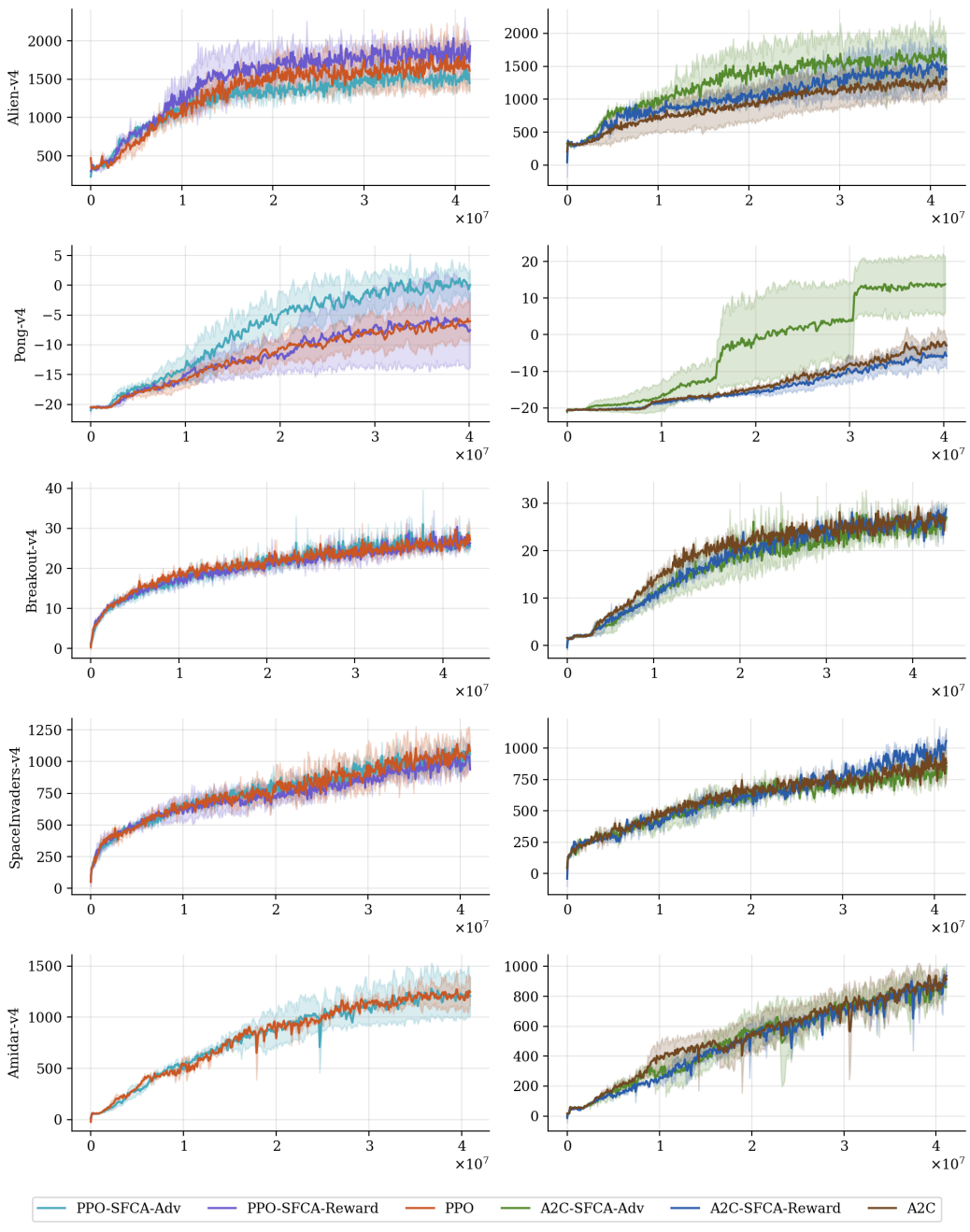


Figure 10: Atari-ram-v4 environments: Training reward as a function of environment step. Reported values are the mean and standard deviation of the average episode reward, aggregated across 5 random seeds.

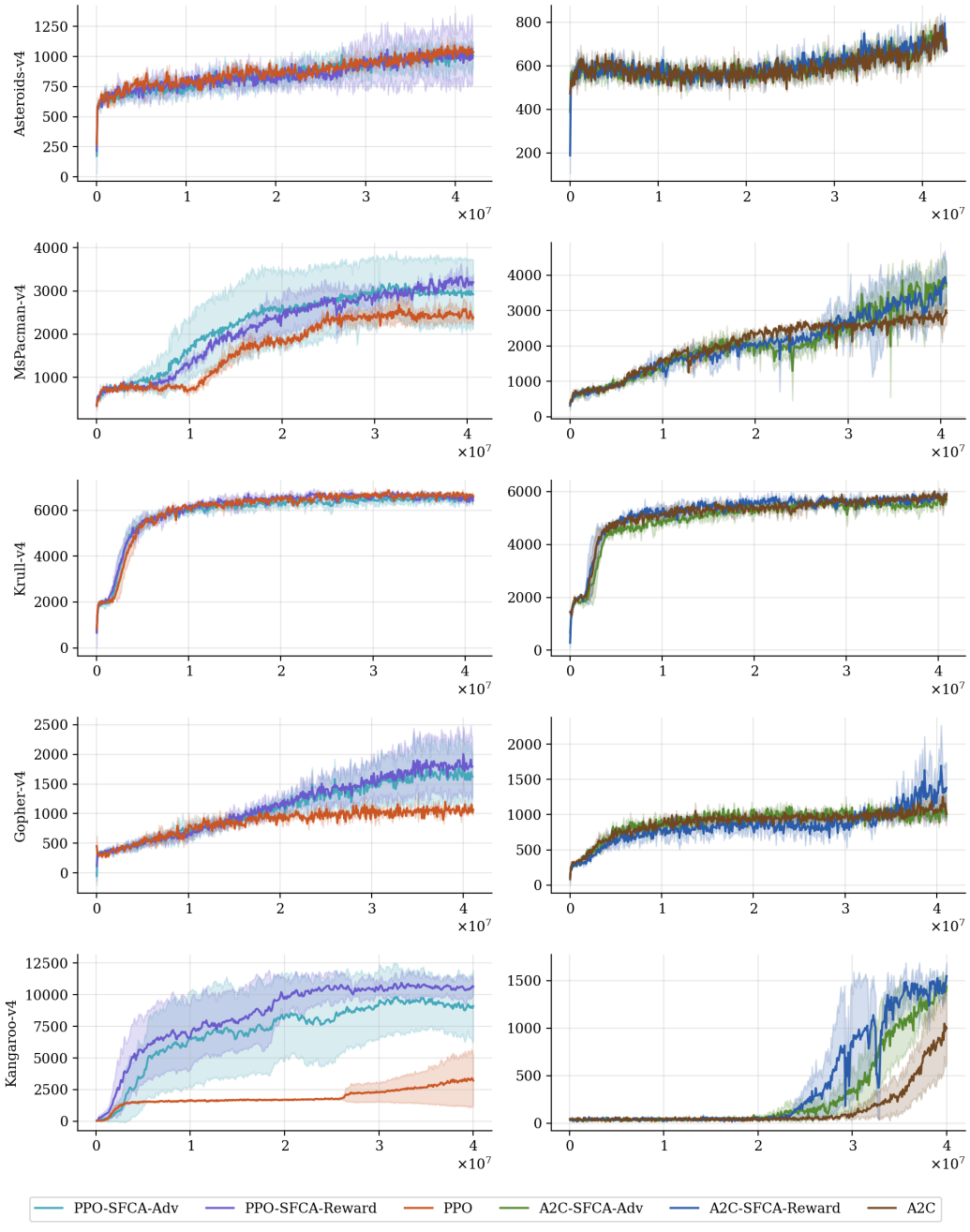


Figure 11: Atari-ram-v4 environments: Training reward as a function of environment step. Reported values are the mean and standard deviation of the average episode reward, aggregated across 5 random seeds.

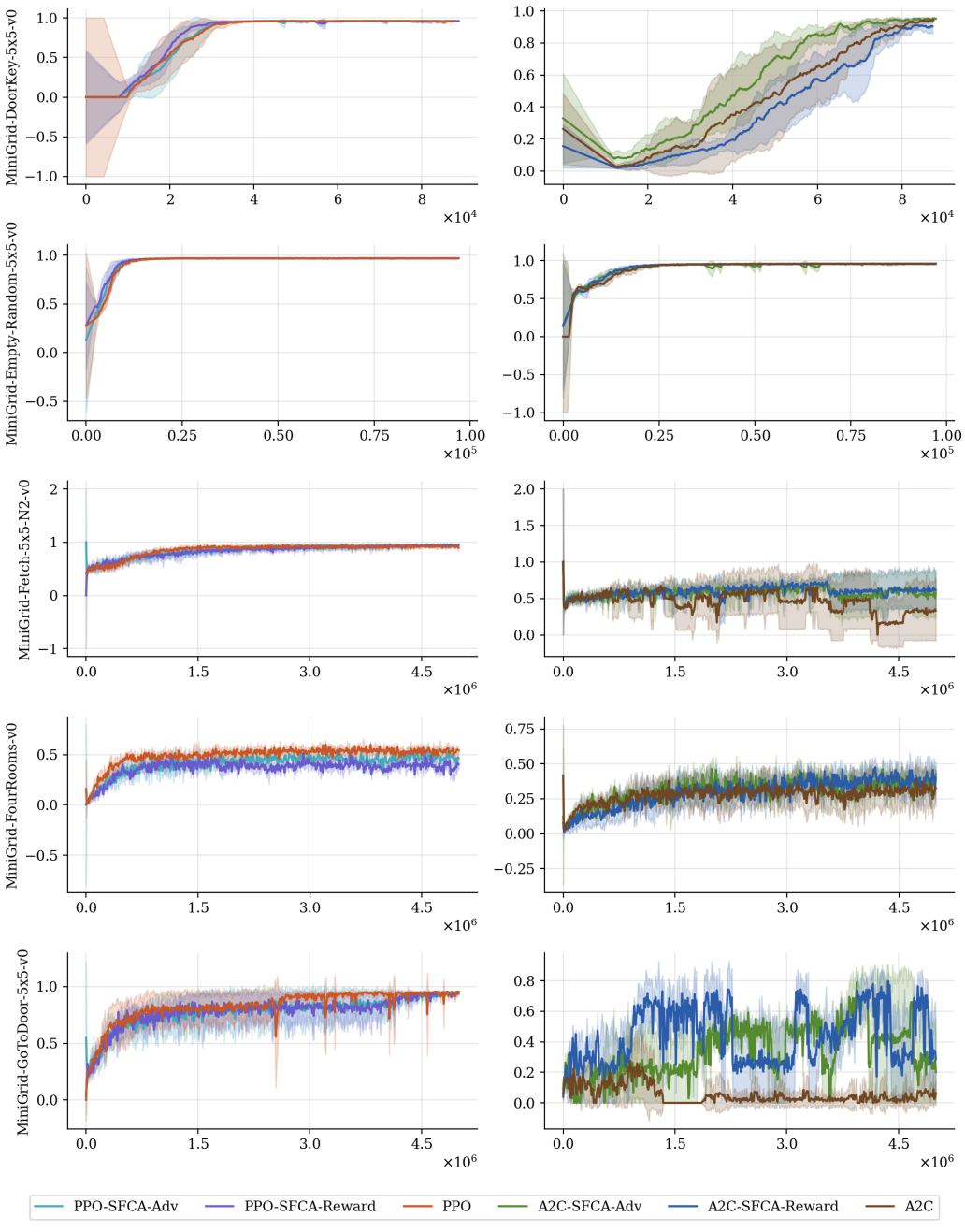


Figure 12: MiniGrid environments: Training reward as a function of environment step. Reported values are the mean and standard deviation of the average episode reward, aggregated across 5 random seeds.

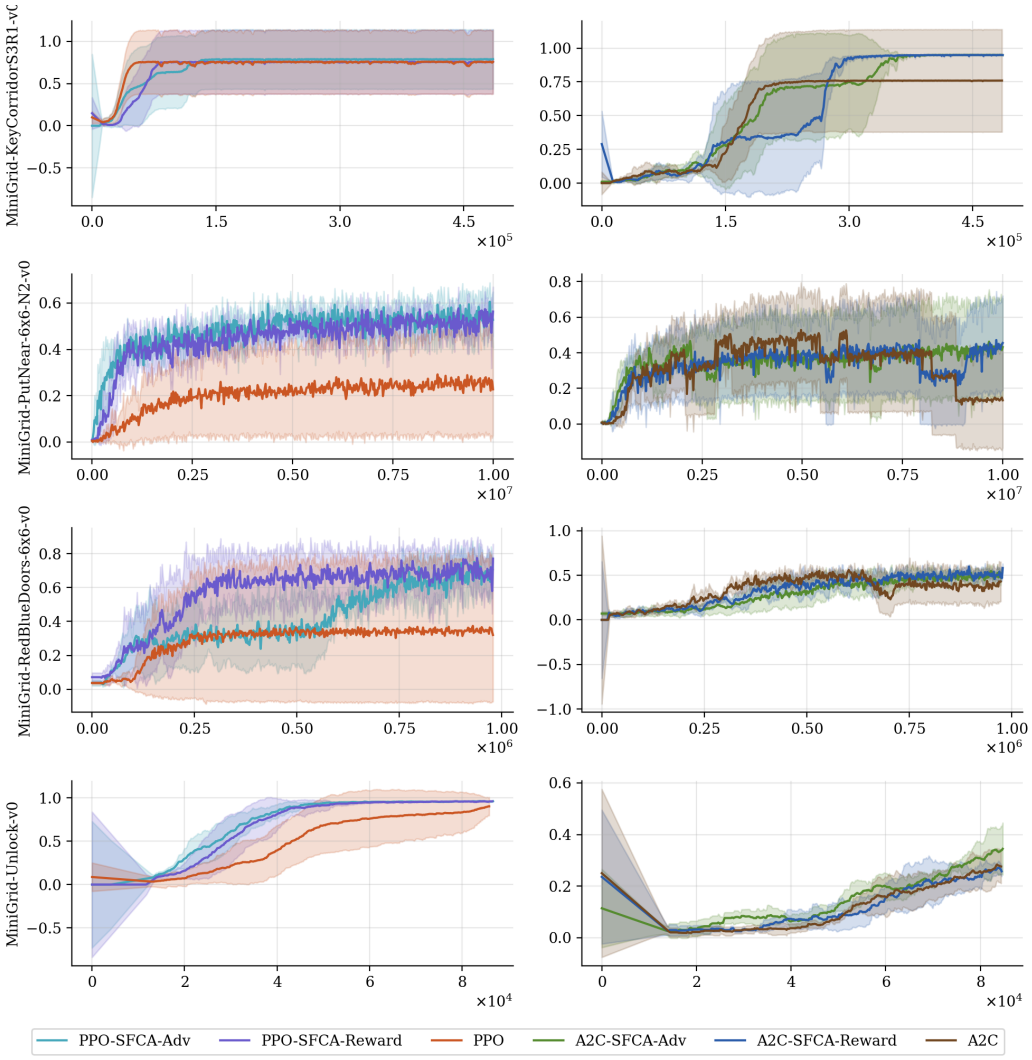


Figure 13: MiniGrid environments: Training reward as a function of environment step. Reported values are the mean and standard deviation of the average episode reward, aggregated across 5 random seeds.