KIPPO: Koopman-Inspired Proximal Policy Optimization

Andrei Cozma , **Landon Harris** and **Hairong Qi**

University of Tennessee, Knoxville

{acozma, lharri73}@vols.utk.edu, hqi@utk.edu

Abstract

Reinforcement Learning (RL) has made significant strides in various domains, and policy gradient methods like Proximal Policy Optimization (PPO) have gained popularity due to their balance in performance, training stability, and computational efficiency. These methods directly optimize policies through gradient-based updates. However, developing effective control policies for environments with complex and non-linear dynamics remains a challenge. High variance in gradient estimates and non-convex optimization landscapes often lead to unstable learning trajectories. Koopman Operator Theory has emerged as a powerful framework for studying non-linear systems through an infinite-dimensional linear operator that acts on a higher-dimensional space of measurement functions. In contrast with their non-linear counterparts, linear systems are simpler, more predictable, and easier to analyze. In this paper, we present Koopman-Inspired Proximal Policy Optimization (KIPPO), which learns an approximately linear latent-space representation of the underlying system's dynamics while retaining essential features for effective policy learning. This is achieved through a Koopman-approximation auxiliary network that can be added to the baseline policy optimization algorithms without altering the architecture of the core policy or value function. Extensive experimental results demonstrate consistent improvements over the PPO baseline with 6–60% increased performance while reducing variability by up to 91% when evaluated on various continuous control tasks.

1 Introduction

RL provides a powerful framework for sequential decision-making tasks, enabling agents to learn optimal behaviors through interaction with their environment [Sutton and Barto,

---

Extended version with comprehensive appendices containing ablation studies, hyperparameter analyses, pseudocode, and implementation details is available at: https://andreicozma.com/KIPPO.

2018]. Policy optimization, a core component of this framework, determines the optimal mapping from states to actions that maximizes an agent's cumulative returns. Policy gradient methods excel in continuous control tasks by directly optimizing policies through gradient-based updates [Sutton *et al.*, 1999]. However, developing effective control policies for environments with complex and non-linear dynamics remains a challenge. This challenge, combined with non-convex optimization landscapes, leads to high-variance gradient estimates and unstable updates. The optimization process often diverges or oscillates, impeding convergence to optimal policies.

The field of dynamical systems studies mathematical models of evolving processes, focusing on patterns, stability, and long-term behavior [Broer and Takens, 2010; Heij *et al.*, 2021]. Although linear systems are more predictable, many real-world systems are non-linear, where small changes in initial conditions can lead to drastically different outcomes [Mezic and Runolfsson, 2004]. Koopman Operator Theory, a powerful tool for studying non-linear systems, finds a linearized description in a higher-dimensional space of measurement functions, known as the Koopman observable space [Koopman, 1931; Brunton *et al.*, 2021]. This process maps original state variables to observable functions, extracting useful state information. The Koopman operator, an infinite-dimensional linear operator, evolves these observables linearly in time, enabling linear descriptions of non-linear systems. Data-driven methods like Dynamic Mode Decomposition (DMD) and deep learning advances have enabled approximating the Koopman operator directly from data [Schmid, 2010; Yeung *et al.*, 2017; Lusch *et al.*, 2018].

Building on these foundations, we propose KIPPO, a method that uses Koopman-inspired representation learning to address a key challenge of policy gradient methods like PPO: high-variance gradient estimates in complex, non-linear environments. Rather than seeking perfectly linear representations of non-linear systems, our approach introduces an inductive bias that encourages approximate linearity along policy trajectories. This soft constraint simplifies underlying dynamics while preserving essential features for policy learning. We achieve this through a Koopman-approximation auxiliary network and targeted constraints that balance the complexity of latent dynamics. KIPPO's architecture uses state encoders/decoders and linear transition matrices to predict future states over a fixed horizon, imposing structure on
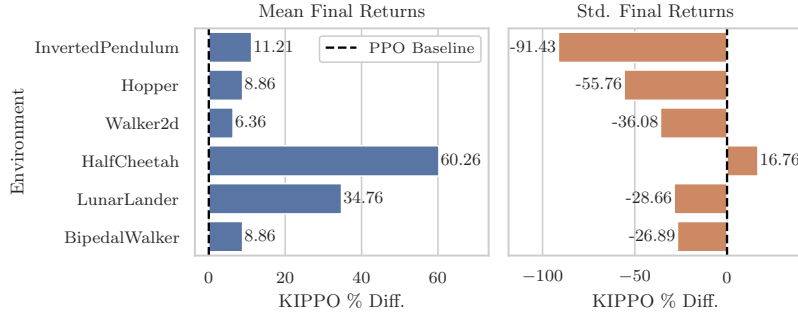
Figure 1: Visualization of KIPPO['s] improvements relative to the PPO baseline in terms of average performance (mean, higher is better — left) and consistency (std., lower is better — right) across four trials per environment.

the latent space while minimizing information loss. By combining advances in deep learning with Koopman theory principles, this approach simplifies system behavior and improves policy performance.

Our approach creates a mutually beneficial feedback loop: policy gradients identify important state space regions through exploratory rollouts, while our linearization technique reduces gradient variance specifically in these critical regions. By focusing linearization efforts locally along policy-explored trajectories instead of attempting global linearization, this targeted approach maintains computational efficiency while delivering benefits precisely where they matter most for the current policy.

The main contribution of this study is threefold:

1. **Koopman-Inspired Policy Optimization.** We propose KIPPO, an on-policy algorithm that incorporates Koopman operator principles directly into policy gradient updates. By learning an approximately linear latent-space representation, KIPPO stabilizes gradient estimates and enhances control over non-linear dynamics.

2. **Decoupled Auxiliary Representation Learning.** KIPPO adds an auxiliary network to policy gradient baselines like PPO without altering the core policy or value function architecture. This design allows the policy to train on a simpler, encoded state space while the auxiliary network enforces a linear-like structure. As a result, standard PPO hyperparameters and training loops remain largely intact.

3. **Performance and Stability Improvements.** Across MuJoCo and Box2D tasks, KIPPO consistently achieves 6–60% higher mean returns and a 26–91% reduction in variance compared to baseline PPO, as shown in Fig. 1. These empirical gains attest to the efficacy of Koopman-inspired constraints in mitigating high-variance updates and accelerating convergence.

The remainder of this paper is organized as follows: Sec. 2 presents essential background. Sec. 3 describes the KIPPO framework in detail. Sec. 4 presents extensive experimental results across multiple environments. Sec. 5 concludes the paper by summarizing our findings and outlining promising directions for future research.

2 Background and Related Works

RL provides a framework for agents to learn optimal behaviors through interactions with their environment. These interactions are formalized as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ the transition probability function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, and $\gamma \in [0,1]$ the discount factor.

This interaction mirrors the feedback loop in control systems, where the agent acts as the controller and the environment represents the system being controlled. The environment in RL can then be modeled as a continuous-time dynamical system: $x_{t+1} = \mathbf{F}(x_t, u_t)$, where $x_t \in \mathcal{S}$ and $u_t \in \mathcal{A}$ are the state and action, respectively, at time t, and $\mathbf{F} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the (often non-linear) state transition function.

RL and Policy Gradient Methods. RL algorithms fall into several categories: *value-based methods* and *policy-based methods*. The latter can be further categorized into *on-policy methods* that learn exclusively from current policy experiences and *off-policy methods* that can learn from any policy's experiences. RL algorithms can also be grouped by model-based versus model-free approaches, distinguished by whether they explicitly learn environment dynamics.

Generally speaking, policy optimization algorithms learn a parameterized policy π_θ by optimizing parameters θ through gradient descent with respect to the expected return. While successful in various tasks, policy gradient methods face fundamental challenges with complex, non-linear dynamics. High variance makes gradient estimates less reliable, and non-convex optimization landscapes often lead to unstable learning trajectories.

Actor-critic methods, such as Advantage Actor-Critic (A2C) [Mnih *et al.*, 2016], address these issues by using a value function (critic) to provide lower-variance targets for policy (actor) updates. Trust region methods like Trust Region Policy Optimization (TRPO) [Schulman *et al.*, 2017a] constrain policy updates using Kullback-Leibler (KL) divergence to ensure new policies remain within a trusted region. However, TRPO's second-order optimization approach increases computational complexity. PPO [Schulman *et al.*, 2017b] addresses these limitations with a first-order approach that avoids Hessian computations while maintaining trust region properties, becoming a leading algorithm for continuous and

discrete control tasks. For its policy component, it introduces a clipped surrogate objective function: $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t \right) \right]$ where $\hat{\mathbb{E}}_t$ denotes the empirical average over timesteps, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio between policies, $\hat{A}_t$ is the estimated advantage, and ϵ is a hyperparameter limiting policy change (typically 0.2 for continuous domains, 0.1 for discrete tasks). The algorithm alternates between collecting experiences and optimizing a combined objective (clipped policy, value function, and entropy terms) via minibatch gradient ascent. This achieves the same policy constraints as TRPO without second-order derivatives, improving performance while maintaining efficiency.

Recent advances include Stochastic Latent Actor-Critic (SLAC) [Lee *et al.*, 2020], which integrates policy optimization with latent state representation learning using variational inference for improved sample efficiency; Model-Based Policy Optimization (MBPO) [Janner *et al.*, 2021], which employs ensemble dynamics models for synthetic experience generation; and Robust Policy Optimization (RPO) [Rahman and Xue, 2022], which extends PPO with perturbed action distributions to maintain policy entropy and enhance robustness. Nevertheless, achieving robust generalization and efficient learning in complex, non-linear systems remains an open challenge.

Koopman Operator Theory provides a mathematical framework to transform non-linear dynamics into linear operators acting on observable functions [Koopman, 1931; Mezic, 2005; Rowley *et al.*, 2009]. The key insight is that while system dynamics may be highly non-linear in state space, they can be represented linearly in an appropriate space of observable functions (measurement functions that extract system information). For systems with control inputs, the Koopman formulation is:

$$g(x_{t+1}) = y_{t+1} = \mathbf{K}y_t + \mathbf{B}v_t \tag{1}$$

where $y_t = g(x_t) \in \mathbb{R}^m$ are state observables, $v_t = f(u_t) \in \mathbb{R}^k$ are control observables, and $\mathbf{K} \in \mathbb{R}^{m \times m}$ and $\mathbf{B} \in \mathbb{R}^{m \times k}$ are finite-dimensional matrices approximating the infinite-dimensional Koopman operator. The challenge lies in finding appropriate observable functions that enable effective linearization, which can be learned using deep neural networks [Lusch *et al.*, 2018; Dey and Davis, 2023].

Research integrating Koopman theory with control and RL has progressed along two paths: system modeling with control, and integration with model-free RL algorithms. The former includes work by [Han *et al.*, 2020] and [Shi and Meng, 2022], who developed frameworks combining Koopman operators with linear control methods like Linear Quadratic Regulators (LQRs) and Model Predictive Controls (MPCs). [Yin *et al.*, 2022] combined Koopman theory with LQR to create differentiable policies embedding optimal control principles. The latter focuses on model-free RL, including work by [Song *et al.*, 2021] that introduced Deep Koopman Reinforcement Learning (DKRL), which uses local Koopman operators to improve data efficiency. [Weissenbacher *et al.*, 2022] proposed Koopman Forward (Conservative) Q-Learning (KFC), an offline algorithm that leverages Koopman theory to infer symmetries in system dynamics.

While model-based methods using Koopman theory with MPCs have been thoroughly investigated [Korda and Mezić, 2018], they typically incur high computational costs due to optimization requirements at each timestep. Model-free approaches avoid this overhead but have received less attention.

KIPPO differs from existing approaches by 1) focusing on on-policy learning improvement rather than global linearization, 2) integrating representation learning directly into policy optimization, 3) decoupling representation learning from policy updates, and 4) measuring success through policy performance and stability rather than global approximation quality.

3 Methodology

KIPPO introduces a Koopman-inspired representation learning framework that operates independently alongside the core policy optimization process. This approach unifies traditional RL with Koopman operator theory while maintaining practical implementability. The framework builds on several key principles:

- **Decoupled Optimization**: Representation learning and policy optimization are deliberately separated to prevent interference between objectives, where the core policy algorithm remains unchanged, operating on encoded states without modification to its optimization process

- **Local Linearization**: Rather than attempting global linearization, the framework focuses on simplifying dynamics along policy-explored trajectories

- **Balanced Complexity**: Loss functions are designed to balance the competing objectives of simplification and information preservation

KIPPO's key innovation is its targeted approach to linearity, expressed as $\boldsymbol{\varphi}_x(x_{t+1}) \approx \mathbf{K}\boldsymbol{\varphi}_x(x_t) + \mathbf{B}\boldsymbol{\varphi}_u(u_t)$. Unlike networks with standard linear output layers, KIPPO enforces linear dynamics across time steps as a soft constraint, applying this only to policy-explored trajectories. This creates an inductive bias on temporal transitions rather than static mappings, promoting stable gradient flow while avoiding the computational burden of global linearization.

3.1 Architecture Design

Building upon these design principles, the core innovation of KIPPO lies in learning a latent representation where complex, non-linear environment dynamics can be effectively approximated by linear operations within regions of the state space explored by the current policy.

While sharing similarities with representation learning and data compression, KIPPO employs a higher-dimensional latent space than the original state space. This design choice follows from Koopman theory, which demonstrates that non-linear dynamics can be linearized through appropriate lifting to higher-dimensional spaces of observable functions [Budišić *et al.*, 2012].

As shown in Fig. 2, KIPPO comprises several interconnected components:

- A state autoencoder consisting of an encoder $\boldsymbol{\varphi}_x : \mathcal{S} \to \mathbb{R}^m$ and decoder $\boldsymbol{\varphi}_x^{-1} : \mathbb{R}^m \to \mathcal{S}$, which respectively
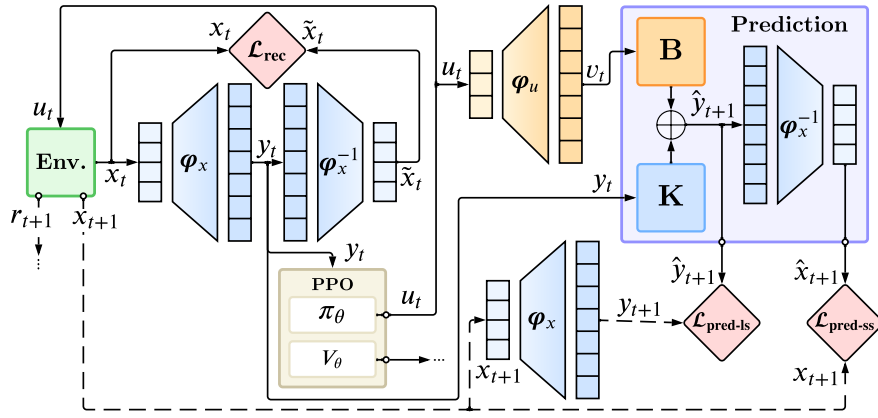
Figure 2: The KIPPO framework architecture. The state autoencoder (encoder φ_x and decoder φ_x^{-1}) learns a compact latent representation of environment states. The action encoder φ_u maps actions to this feature space. Within the latent space, dynamics are governed by the linear state-transition matrix $\mathbf{K}$ and control matrix $\mathbf{B}$. The policy optimization algorithm operates on the encoded states $y_t = \varphi_x(x_t)$. This architecture enables the reformulation of nonlinear environments into a structure aligned with Koopman control theory Eq. 1.

map states to latent representations and reconstruct original states

- An action encoder $\varphi_u : \mathcal{A} \to \mathbb{R}^k$ that maps actions to the latent space

- Linear system matrices $\mathbf{K} \in \mathbb{R}^{m \times m}$ and $\mathbf{B} \in \mathbb{R}^{m \times k}$ that govern the dynamics within the latent space

The encoder and decoder networks use Multi-Layer Perceptrons (MLPs) with hyperbolic tangent (tanh) activation functions, chosen for their smooth gradients and bounded output range. All trainable parameters employ Xavier uniform initialization [Glorot and Bengio, 2010] to promote stable learning in deep networks, except for $\mathbf{K}$, which uses orthogonal initialization [Saxe *et al.*, 2014] for stable gradient flow, and $\mathbf{B}$, which starts with zeros to allow gradual learning of control effects. The number of layers and neurons per layer remain consistent across the state encoder, decoder, and action encoder networks, typically using 2–3 hidden layers with 64–256 units each. This architectural consistency helps maintain balanced representational capacity across components while remaining computationally efficient.

The use of non-linear activation functions might appear counterintuitive given our goal of linear dynamics. However, these non-linearities are essential for learning effective Koopman observables, enabling the networks to discover appropriate lifting functions that map the original system to a space where linear approximations become effective along policy-relevant trajectories. While the Koopman operator governing the evolution of observables is inherently linear, the method of obtaining these observables need not be linear. These non-linearities enable the MLPs to act as universal function approximators, making them well-suited for approximating the Koopman observables in the latent space.

The dimensions of the state-transition matrix $\mathbf{K}$ and control matrix $\mathbf{B}$ correspond to the chosen latent space dimensionality. This is typically set to 2–4 times the state dimension, providing sufficient capacity to capture complex dynamics without excessive computational overhead. These matrices are learnable parameters optimized alongside other components, enabling

the framework to learn environment-specific representations directly from experience.

These components work together to approximate the Koopman operator's action on observable functions, with the encoders serving as learnable observable functions and the linear matrices capturing the evolution of these observables. This connection to Koopman theory provides theoretical grounding for our approach while remaining practically implementable within the RL context.

3.2 Future State Prediction Process

The prediction process forecasts states over horizon H using learned linear latent-space dynamics. For an initial state x_0 and an action sequence $u_{0:H}$, the process begins with the initial encoding of the state into a latent representation, $y_0 = \phi_x(x_0)$. This is followed by an iterative prediction using learned dynamics, expressed as $\hat{y}_{h+1} = \mathbf{K}y_h + \mathbf{B}\varphi_u(u_h)$. This process yields a sequence of predicted latent states, which can be decoded back to the original state space using $\hat{x}_{h+1} = \varphi_x^{-1}(\hat{y}_{h+1})$.

The process implements the finite-dimensional approximation of the Koopman-based control formulation from Eq. 1, where φ_x and φ_u represent g and f, respectively. This process primarily constrains the learning of the latent representation to reduce gradient variance, rather than for generating additional training data or performing planning. Importantly, this serves as a soft constraint; perfect linearity is not required, but the representation is encouraged to be approximately linear along policy trajectories to enable more stable policy optimization.

The multi-step prediction shapes representations by enforcing temporal consistency only along current-policy trajectories, avoiding unrealistic global linearity assumptions. This allows KIPPO to benefit from structured representations without requiring lookahead or model predictive control. Unlike model-based planning methods, we never use the learned model for planning; our method focuses solely on variance reduction through temporal coherence. This novel application of predictive models specifically addresses the noisy updates that challenge policy gradient methods.

The prediction horizon H balances computational cost against prediction depth. Empirically, horizons of 8–32 steps are effective, with longer horizons benefiting environments with significant temporal dependencies or sparse rewards. Our analysis shows that longer horizons benefit moderately complex environments but offer diminishing returns or instability in very simple or highly complex ones. The supplementary material includes both a detailed examination of prediction horizon effects and implementation steps for the complete prediction process.

3.3 Loss Formulation

Drawing inspiration from Koopman operator theory, KIPPO employs three complementary loss components that shape the latent space to achieve four key properties: 1) *informativeness*—preserving essential state information for accurate decision-making; 2) *simplification*—enabling linear approximations specifically along policy trajectories; 3) *predictability*—supporting accurate multi-step predictions for better temporal coherence and reduced gradient variance; and 4) *consistency*—ensuring alignment with true environment dynamics for effective learning.

Reconstruction Loss

The reconstruction loss ensures the latent space retains sufficient information for accurate state reconstruction:

$$\mathcal{L}_{\text{rec}}(t) = \left\{ \boldsymbol{\varphi}_x^{-1}(\boldsymbol{\varphi}_x(x_t)) - x_t \right\}^2 \quad (2)$$

This loss primarily addresses "informativeness" while aligning with Koopman theory principles, where observable functions are typically assumed to be invertible. This formulation allows a bijective mapping between the original state space and latent space, maintaining a meaningful connection that supports both representation learning and policy optimization.

Note that the action reconstruction loss is omitted since the sole purpose of the action encoder is to influence state transitions in the latent space, and the accuracy of action encoding is implicitly enforced through future state prediction losses. Empirical studies also confirm that including action reconstruction terms does not yield significant performance improvements.

Latent-Space Prediction Loss

The latent-space prediction loss primarily targets "simplification" and "predictability" within policy-relevant regions. It encourages learning a representation where dynamics can be effectively approximated by linear operations along policy-explored trajectories:

$$\mathcal{L}_{\text{pred-ls}}(t) = \frac{1}{H} \sum_{h=1}^{H} m_{t,h} \left(\hat{y}_{t+h} - \boldsymbol{\varphi}_x(x_{t+h}) \right)^2 \quad (3)$$

where $m_{t,h}$ handles episode boundaries through a binary mask:

$$m_{t,h} = \begin{cases} 1, & \text{if trajectory not ended by step } (t+h-1), \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

The binary mask is essential for handling variable-length trajectories, such that the prediction process is not penalized for discontinuities introduced by environment resets at episode boundaries.

The loss term facilitates the learning of representations where dynamics can be effectively approximated by linear operations, as $\hat{y}_{t+h}$ is generated using linear matrices $\mathbf{K}$ and $\mathbf{B}$. It also enhances predictability by minimizing multi-step prediction errors directly in the latent space, while supporting simplification by encouraging the encoder to find representations where linear predictions maintain accuracy over multiple timesteps.

This loss term is particularly important for maintaining the framework's Koopman-inspired aspects, as it drives the learning of representations that align with Koopman theory's principle of lifting non-linear dynamics to spaces where linear approximations become effective.

State-Space Prediction Loss

The state-space prediction loss primarily addresses "consistency" and "predictability", maintaining fidelity to true dynamics while ensuring meaningful state predictions:

$$\mathcal{L}_{\text{pred-ss}}(t) = \frac{1}{H} \sum_{h=1}^{H} m_{t,h} \left(\boldsymbol{\varphi}_x^{-1}(\hat{y}_{t+h}) - x_{t+h} \right)^2 \quad (5)$$

The loss helps prevent the latent space from diverging too far from physically meaningful representations, which is essential for learning effective control policies.

This dual-space prediction approach ensures the latent dynamics align with true environment behavior when mapped back to state space. It also promotes learning of latent representations that maintain predictive power across multiple timesteps, indirectly reinforcing informativeness by requiring accurate long-term state reconstruction. The improved temporal consistency from these representations helps reduce gradient variance in policy updates, though we never use these predictions for planning.

Total Representation Loss

The total representation loss is a weighted sum of the three components:

$$\mathcal{L}_{\text{KI}} = \frac{1}{T} \sum_{t=0}^{T} \left(\omega_1 \mathcal{L}_{\text{rec}}(t) + \omega_2 \mathcal{L}_{\text{pred-ls}}(t) + \omega_3 \mathcal{L}_{\text{pred-ss}}(t) \right) \quad (6)$$

where T represents the number of steps collected during rollouts. The weights ω_1, ω_2, and ω_3 incorporate several factors, including relative scales between latent and state space dimensionalities, task-specific requirements, and environment characteristics. Sec. 4.3 investigates the impact of each loss term on both the return and stability of learning.

3.4 Overview of Training Process

The training process in KIPPO alternates between rollout and optimization phases until reaching a predetermined number of environment steps. During rollouts, the agent collects states,

actions, rewards, and additional sequences needed for representation losses, storing them in separate buffers. Each rollout phase collects 2,048 environment steps across multiple trajectories, resetting the environment when necessary to ensure diverse experiences.

Both KIPPO and PPO use identical on-policy rollouts and operate with the same available information. KIPPO utilizes future states solely as auxiliary loss targets (never as policy inputs), which is consistent with standard auxiliary objective practices in on-policy RL. During both training and inference, both methods receive identical trajectories and current-state information, with KIPPO applying state encoding while PPO uses raw states.

The optimization phase processes the collected data to update all components. The algorithm divides 2,048 steps into 32 mini-batches, computing the three key losses to update representation learning components.

The total framework loss combines the weighted representation loss and standard PPO loss:

$$\mathcal{L}_{\text{KIPPO}} = \mathcal{L}_{\text{KI}} + \mathcal{L}_{\text{PPO}} \qquad (7)$$

Both components update their parameters using the Adam optimizer. The optimization process runs for 10 epochs, allowing refinement of both the latent representation and the policy. After optimization, a new rollout phase begins, continuing this cycle until reaching 1 million environment steps.

The latent representation is learned incrementally throughout training, with parameters adapting gradually across rollout-optimization cycles. We observe stability, with representations evolving smoothly between updates, maintaining consistent state encodings, and preventing disruptive changes that could destabilize learning. This is particularly important for policy gradient methods sensitive to sudden representation shifts.

A key feature is the complete decoupling of representation learning from policy optimization. The representation learning components optimize independently from policy and value networks, ensuring improvements stem directly from the learned representation. We refer readers to the supplementary materials for complete implementation details and pseudocode.

4 Experiments and Results

We evaluate the effectiveness of KIPPO compared to baseline PPO and RPO algorithms across diverse continuous control tasks, measuring both performance improvements and reduced variability.

4.1 Experimental Setup

Environments

We evaluate six continuous control environments from Gymnasium [Towers *et al.*, 2023] using MuJoCo [Todorov *et al.*, 2012] and Box2D [Catto, 2007], forming a comprehensive testbed with diverse complexity levels and control challenges. The environments' varying non-linearity and temporal dependencies help evaluate the algorithm's robustness and its ability to learn effective representations in the Koopman observable space. We chose these testbeds to systematically evaluate

how our approach reduces gradient variance across different complexity levels while keeping the analysis tractable.

To facilitate discussion, we classify the six environments by their complexity levels, defined using the dimensions of the state space, $|\mathcal{S}|$, and the action space, $|\mathcal{A}|$. An environment is considered to have "low" complexity if $|\mathcal{S}| + |\mathcal{A}| < 10$, "medium" complexity if $10 \leq |\mathcal{S}| + |\mathcal{A}| < 20$, and "high" complexity if $|\mathcal{S}| + |\mathcal{A}| \geq 20$. This is summarized in Table 1. These environments include Inverted Pendulum, Lunar Lander, Bipedal Walker, Hopper, Walker2d, and HalfCheetah, providing a diverse set of control challenges. Complete environment specifications are available in the supplementary material.

Environment	Complexity	$|\mathcal{S}|$	$|\mathcal{A}|$
InvertedPendulum-v4	Low	4	1
LunarLanderCont.-v2	Medium	8	2
Hopper-v4	Medium	11	3
BipedalWalker-v3	High	24	4
Walker2d-v4	High	17	6
HalfCheetah-v4	High	17	6

Table 1: Environments and their complexity levels based on the dimensionality of their state and action spaces.

Training Configuration

For meaningful comparisons, we implement our benchmarks using the PPO and RPO implementations from the CleanRL library [Huang *et al.*, 2022]. We maintain CleanRL's default hyperparameters for both algorithms.

Each experiment uses 4 random initialization seeds (1, 2, 3, 4) per environment. We selected 4 seeds as a balance between the original PPO paper's 3 seeds [Schulman *et al.*, 2017b] and CleanRL's standard 5 seeds. These seeds determine both the environment's initial states and model parameter initialization. To ensure fair comparison, we use identical random seeds and initialization patterns across all methods. Each training run consists of exactly 1 million environment steps. Hardware specifications and reference runtime are provided in the supplementary material.

Evaluation Metrics

Given the inherent stochasticity in both environment and learning processes, we employ the Exponentially Weighted Moving Average (EWMA) of episodic returns to capture learning trends effectively:

$$\text{EWMA}_t = \alpha \cdot \text{EWMA}_{t-1} + (1 - \alpha) \cdot G_t, \qquad (8)$$

where G_t is the expected (discounted) return, summing rewards weighted by γ^t at each time step; and empirically determined $\alpha = 0.05$. α balances responsiveness to recent changes with historical context, reducing noise by filtering short-term fluctuations and ensuring robustness against outliers.

We also use the Cumulative Temporal Error (CTE) metric:

$$\text{CTE} = \frac{1}{H} \sum_{h=1}^{H} \frac{1}{h} \sum_{k=1}^{h} |\hat{x}_k - x_k| \qquad (9)$$

where k indices the individual timestep. While EWMA evaluates the overall agent performance, CTE specifically measures the representation quality by comparing predicted states ($\hat{x}$) with actual states (x) across varying horizons.

For a comprehensive evaluation, we analyze both metrics through their means and standard deviations (std.) across 4 independent training runs.

4.2 Comparison with Baselines

We conduct the first set of experiments by comparing KIPPO's performance with two baseline policy gradient methods, PPO and RPO, selected due to either popularity or state-of-the-art performance. The comparison is summarized in Table 2. Fig. 1 presents the percent difference in these metrics relative to the PPO baseline.

From both Table 2 and Fig. 1, we observe that KIPPO achieves overwhelmingly better mean performance in all environments, with improvements ranging from 6.36% to 60.26% for the PPO baseline and from 11.86% to 142.18% for the RPO baseline. KIPPO also shows lower std. in most environments, demonstrating enhanced consistency across seeds, reducing variance by 26.89–91.43% versus PPO (one exception) and 58.94–90.21% versus RPO (two exceptions). We will further discuss the exceptional cases in the ablation study.

This dual improvement in performance and stability highlights the fundamental advantage of incorporating Koopman-inspired representation. Complete learning curves for all environments are provided in the supplementary material.

4.3 Ablation Study of Loss Components

To understand the mechanisms underlying KIPPO's performance advantages, we conduct a systematic ablation study of its core components. Table 3 shows the result of one environment. Results for the other five environments are available in the supplementary material.

Several key findings emerge from Table 3. Note that the first row per environment represents baseline PPO performance. First, the reconstruction loss alone yields results comparable to those of the baseline. Second, integrating all losses produces the best mean EWMA and lowest standard deviation in most cases. Third, the mean CTE decreases systematically with the incorporation of additional loss components. And finally, the latent-space prediction loss consistently reduces both CTE mean and standard deviation.

While the ablation results in Table 3 highlight the contributions of each loss component, they also reveal that KIPPO's overall gains depend strongly on the level of non-linearity (complexity) of each environment. We hypothesize a *sublinear* ("logarithmic") relationship between performance gain and complexity, meaning that beyond a certain point, additional non-linearity or complexity diminishes marginal returns and raises variance.

Fig. 3 shows our evaluation of KIPPO across varying environment complexities. We analyze two key metrics compared to the PPO baseline: relative improvement in average performance (mean) and consistency of results (std.).

Overall, KIPPO scales effectively across a broad range of non-linear control tasks but hits an upper limit as complexity grows. In simpler domains, overemphasizing latent-
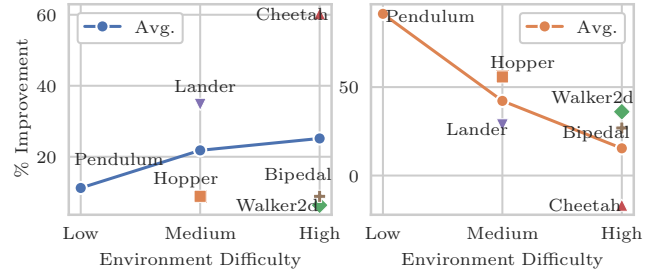

Figure 3: The mean percent improvement of environments with various levels of complexity in performance gain (left) and variance reduction (right) of final returns by KIPPO compared to PPO.

space predictions can harm stability unless balanced by state-space constraints. In extreme tasks, significant raw gains come with heightened variance. Thus, KIPPO extends PPO's performance boundary significantly, but the underlying non-linearities impose a *logarithmic* or sublinear bound on further improvements.

4.4 Sensitivity Analysis and Limitations

In this set of experiments, we conduct extensive parameter sensitivity studies, particularly focusing on the latent dimension and prediction horizon to better understand KIPPO's limitations. We refer readers to the supplementary material for comprehensive quantitative results examining effects of latent dimension and prediction horizon.

We observe that performance gains diminish in environments with highly discontinuous transitions (e.g., collisions), contact-rich interactions, or multi-modal behaviors, as the linear latent dynamics struggle with abrupt changes. Despite enforcing approximate linearity only along policy trajectories as a soft constraint, environments with highly chaotic dynamics remain challenging.

In environments with sparse rewards, the advantage over baseline PPO is less pronounced, suggesting that reduced gradient variance benefits are most impactful with frequent feedback signals. We view Koopman-based dynamics as an inductive bias particularly well-suited for certain control problems rather than as a universally valid model. These selected environments provide a controlled setting to test our core hypothesis: linearized latent dynamics can reduce gradient variance in policy optimization.

Training KIPPO takes approximately 15% longer than PPO (15 hours vs. 13 hours for 24 parallel models) due to (1) construction of prediction sequences and (2) computation of multi-step prediction losses. However, this computational overhead exists only during training; at inference time, only the encoder is used with negligible additional computational cost.

To identify the most influential hyperparameters, we train a random forest regressor to predict final returns from hyperparameter configurations. Fig. 4 shows that the latent-space prediction loss weight (ω_2) has the highest importance, followed by the latent dimension and state-space prediction loss weight (each 0.20). For return variability, the three loss weights (each 0.20) dominate, followed by the prediction horizon (0.15).

Environment	PPO		RPO		KIPPO	
	Mean	Std.	Mean	Std.	Mean	Std.
InvertedPendulum-v4	897.57	41.61	892.33	36.46	**998.18**	**3.57**
Hopper-v4	2315.43	226.87	1970.75	390.81	**2520.53**	**100.36**
Walker2d-v4	3126.73	450.30	2451.58	**168.85**	**3325.74**	287.85
HalfCheetah-v4	1927.59	1030.66	1275.56	**706.52**	**3089.20**	1203.42
LunarLanderCont.-v2	208.38	11.59	150.26	20.14	**280.81**	**8.27**
BipedalWalker-v3	235.09	19.03	191.62	35.18	**255.91**	**13.91**

Table 2: Per-environment overview of the main results comparing KIPPO and the baseline PPO and RPO in terms of mean and std. of final episodic returns EWMA across four trials. The bold font highlights the best performance.

Environment	$\mathcal{L}_{\text{rec}}$	$\mathcal{L}_{\text{pred-ls}}$	$\mathcal{L}_{\text{pred-ss}}$	EWMA		CTE	
				Mean	Std.	Mean	Std.
InvertedPendulum-v4	—	—	—	897.57	41.61	—	—
	✓	—	—	897.21	57.50	0.927	0.137
	✓	✓	—	911.27	93.60	0.015	0.003
	✓	—	✓	977.78	21.09	**0.001**	**0.000**
	✓	✓	✓	**998.18**	**3.57**	**0.001**	**0.000**

Table 3: Effect of the loss function components on final episodic returns (EWMA) values and prediction error (CTE). The baseline is shown as the first row for each environment.
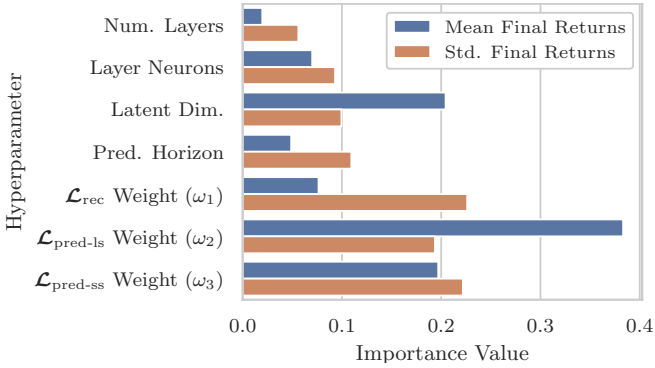

Figure 4: Hyperparameter importance scores derived from a random forest regressor.

5 Conclusions and Future Work

Koopman-Inspired Proximal Policy Optimization (KIPPO) addresses key challenges in policy gradient methods through stable policy optimization for complex non-linear control tasks. Our experiments demonstrate the effectiveness of Koopman-inspired representation learning in policy optimization as showcased in PPO and RPO. This architecture naturally extends to other on-policy algorithms, including TRPO and A2C.

The effectiveness of KIPPO stems from a synergistic bidirectional relationship: policy gradients generate exploratory rollouts that guide which latent regions to linearize, while the resulting representations reduce gradient variance in precisely those regions, creating a more effective feedback loop than decoupled representation learning. This mechanism retains gradient variance reduction benefits even when extended beyond on-policy methods.

Future directions include extending to: 1) off-policy algorithms like Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3), and Soft Actor-Critic (SAC); 2) value-based methods for enhancing Q-function learning; and 3) discrete domains through appropriate latent space formulations. Further research opportunities involve handling discontinuous dynamics and investigating representation robustness under noise and distribution shifts.

Acknowledgements

References

[Broer and Takens, 2010] H. Broer and F. Takens. *Dynamical Systems and Chaos*. Applied Mathematical Sciences. Springer New York, 2010.

[Brunton *et al.*, 2021] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern koopman theory for dynamical systems, 2021.

[Budišić *et al.*, 2012] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4), December 2012.

[Catto, 2007] Erin Catto. Box2d: A 2d physics engine for games. https://box2d.org/, 2007. Accessed: October 10, 2023.

[Dey and Davis, 2023] Sourya Dey and Eric Davis. Dlkoopman: A deep learning software package for koopman theory, 2023.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[Han et al., 2020] Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control, 2020.

[Heij et al., 2021] C. Heij, A.C.M. Ran, and F. van Schagen. *Introduction to Mathematical Systems Theory: Discrete Time Linear Systems, Control and Identification*. Springer International Publishing, 2021.

[Huang et al., 2022] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

[Janner et al., 2021] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization, 2021.

[Koopman, 1931] B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[Korda and Mezić, 2018] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, July 2018.

[Lee et al., 2020] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

[Lusch et al., 2018] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1), November 2018.

[Mezic and Runolfsson, 2004] I. Mezic and T. Runolfsson. Uncertainty analysis of complex dynamical systems. In *Proceedings of the 2004 American Control Conference*, volume 3, pages 2659–2664 vol.3, June 2004.

[Mezic, 2005] Igor Mezic. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, 08 2005.

[Mnih et al., 2016] 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, 2016.

[Rahman and Xue, 2022] Md Masudur Rahman and Yexiang Xue. Robust policy optimization in deep reinforcement learning, 2022.

[Rowley et al., 2009] Clarence Rowley, Igor Mezic, SHERVIN BAGHERI, Philipp Schlatter, and DAN HENNINGSON. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115 – 127, 12 2009.

[Saxe et al., 2014] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2014.

[Schmid, 2010] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[Schulman et al., 2017a] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[Schulman et al., 2017b] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[Shi and Meng, 2022] Haojie Shi and Max Q. H. Meng. Deep koopman operator with control for nonlinear systems, 2022.

[Song et al., 2021] Lixing Song, Junheng Wang, and Junhong Xu. A data-efficient reinforcement learning method based on local koopman operators. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 515–520, 2021.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[Sutton et al., 1999] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[Todorov et al., 2012] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct 2012.

[Towers et al., 2023] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.

[Weissenbacher et al., 2022] Matthias Weissenbacher, Samarth Sinha, Animesh Garg, and Yoshinobu Kawahara. Koopman q-learning: Offline reinforcement learning via symmetries of dynamics, 2022.

[Yeung et al., 2017] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems, 2017.

[Yin et al., 2022] Hang Yin, Michael C. Welle, and Danica Kragic. Embedding koopman optimal control in robot policy learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13392–13399, 2022.