TruKAN-RPO: An Advanced Framework for On-Policy Methods for Continuous Reinforcement Learning

Ali Bayeh[1]
alibayeh@uregina.ca

Samira Sadaoui[1]
sadaouis@uregina.ca

Malek Mouhoub[1]
mouhoubm@uregina.ca

***Abstract*— Studies on adopting Kolmogorov-Arnold Networks (KANs) for Continuous Reinforcement Learning (CRL) remain limited, even though traditional neural networks often struggle in CRL tasks. Our paper integrates a new KAN variant, termed TruKAN, into the Robust Policy Optimization (RPO) framework. TruKAN is distinguished by its knot placement, knot broadcasting and layer normalization. Knot locations can be either fixed or learned during training. Knot broadcasting determines whether knot sequences are shared across all output channels or maintained as separate, per-channel sets. We propose the TruKAN-RPO framework by integrating TruKAN into the actor and critic components of RPO. We evaluate the performance of two TruKAN-RPO models, TruKAN-RPO-Fixed and TruKAN-RPO-Learnable, and compare them with KAN-RPO and SineKAN-RPO. We train and test these different models using the "MuJoCo-MJX Humanoid" environment under different settings.**

Keywords: Kolmogorov-Arnold Networks (KAN), TruKAN, SineKAN, Truncated Power Functions, Robust Policy Optimization, MuJoCo-MJX Humanoid.

I. Introduction

Several challenges still exist in Reinforcement Learning (RL) applications, such as poor interpretability, training instability, decreased generalization capability, and model's sensitivity to noisy data. These issues are more pronounced in continuous RL (CRL) because of the large and diverse state and action spaces, which make training more difficult and less stable. Multilayer perceptrons (MLPs), used in RL algorithms, fail to effectively exploit the structural properties of agents and CRL environments. KANs (Kolmogorov-Arnold Networks) [11] are considered as an alternative to MLPs in numerous domains, leading to improved interpretability and accuracy. KANs, founded on the Kolmogorov-Arnold representation theorem, approximates a highly-dimensional function by splitting it into structured, simple functions. KANs can formalize complex non-linear relationships while maintaining high performance and increasing interpretability. Despite the success of KANs, there is limited research on their adoption for for CRL [1], [2] and [3]. Our research incorporates a new KAN variant called TruKAN [4] into both the actor and critic components of the on-policy method called Robust Policy Optimization (RPO). TruKAN enhances KANs by substituting the original B-spline function with a truncated power basis derived from the spline theory [5]. TruKAN is characterized by three main design choices as part of internal computations: knot placement, knot broadcasting and layer normalization. In TruKAN, knot locations can be fixed or learned during training. The broadcasting strategy specifies if knot sequences are shared universally among output channels or maintained as unique, per-channel sets. Normalization plays a critical role in TruKAN models, as it provides the primary mechanism for maintaining values within a bounded range while preserving interpretability and avoiding the introduction of additional nonlinearities.

TruKAN preserves the additive structure of KANs while improving expressivity, thereby providing a robust framework for function approximation. We evaluate our TruKAN-RPO framework using the "MuJoCo-MJX Humanoid" robot, a complex, bipedal humanoid figure with human-like kinematics. We use different settings of the robot: Stand, Walk and Run. We compare two variants of TruKAN-RPO, TruKAN-RPO-Fixed and TruKAN-RPO-Learnable, with two other models, KAN-RPO and SineKAN-RPO. To this end, we train all these models on the three configurations of "MuJoCo-MJX Humanoid" robot using the JAX environment.

This paper is organized as follows. Section 2 summarizes KAN architectures and RPO method. Section 3 provides related work on KAN variants. Section 4 summarizes TruKAN features and then describes its integration with the RPO framework. Section 5 explains the setup of our experiments, such as the MuJoCo-MJX Humanoid, JAX environment and training optimization. Section 6 evaluates TruKAN-RPO framework under different KAN variants. Section 7 concludes our work.

II. Background

A. KAN Architectures

KANs [10], [11] can learn and shape smoothly the activation functions during training. The trainable activations allow KANs to adjust dynamically to current data patterns. KAN's effectiveness makes it well-suited for scenarios where interpretability and insights into the decision-making tasks are essential. The Kolmogorov-Arnold representation theorem (KART) indicates that a continuous multi-dimensional function $f(x) = f(x_1, \ldots, x_n)$ can be represented as a sum of one-dimensional functions applied to a linear combination of inputs [11]:

$$f(x) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right)$$

Here, $\varphi_{q,p} : [0,1] \rightarrow \mathbb{R}$ are the inner one-dimensional functions and $\Phi_q : \mathbb{R} \rightarrow \mathbb{R}$ are the outer functions. These functions are represented in KANs with learnable and smooth

[1]Department of Computer Science, University of Regina, Regina, Canada

components, like B-spline. A KAN layer consists of two structural parts: 1) the midway layer with $2n+1$ nodes, related to the inner functions $\varphi_{q,p}$, and 2) the output layer that is the summation of the outer functions Φ_q. The KAN activation values are specified recursively between layers as shown below [11]:

$$x_{l+1,j} = \sum_{i=1}^{l} \hat{x}_{l,j,i} = \sum_{i=1}^{n_l} \varphi_{l,j,i}\left(x_{l,i}\right)$$

$$l = 0, \ldots, L-1, \quad i = 1, \ldots, n_l, \quad j = 1, \ldots, n_{l+1}$$

Here, L denotes the number of layers, n_l the number of nodes in layer l, $x_{l,i}$ the activation value of i^{th} node in l^{th} layer, and $\phi_{l,j,i}$ the activation function connecting node i^{th} in l^{th} layer to neuron j^{th} in $(l+1)^{th}$ layer. Deeper KANs are therefore built by stacking multiple KAN layers.

B. Robust Policy Optimization

Proximal Policy Optimization (PPO) [15] has been widely adopted as a policy gradient method that balances sample efficiency and stability during policy updates. In PPO, the agent learns by optimizing a surrogate objective function that incorporates a clipped probability ratio between the old and new policies. This approach restricts the magnitude of policy changes to prevent destabilizing shifts that could degrade performance. The clipping mechanism mitigates the issues of high variance and large updates that are common in earlier policy gradient approaches, such as REINFORCE or vanilla policy gradients. Additionally, PPO often employs an entropy regularization term to encourage exploration by maintaining policy stochasticity. However, the entropy can diminish over training as the policy converges to more deterministic behaviors. PPO operates in an on-policy manner by collecting trajectories using the current policy for use in generalized advantage estimation (GAE) to compute value targets.

Robust Policy Optimization (RPO) [13] is an extension of PPO that introduces a perturbation mechanism to make the system more resistant to changes by keeping a higher level of policy entropy during training. Unlike PPO, where entropy may decline as the agent exploits learned behaviors, RPO perturbs the base parameterized action distribution (typically a Gaussian distribution) with a scaled uniform random noise. This approach effectively blends the original distribution with a broader, more exploratory one. Consequently, the policy is encouraged to maintain stochasticity, which reduces the risk of prematurely converging to suboptimal deterministic policies and improves generalization in high-dimensional or noisy environments. RPO retains the core elements of PPO, such as clipped surrogate loss and on-policy data collection. However, RPO enhances the policy output by interpolating between unperturbed and perturbed distributions.

III. Related Work

Recent studies restored interest in KANs by enhancing their architectures and introducing variants that use polynomial and function-based methods. For instance, in the year 2024, the study [17] proposed a new network architecture, Chebyshev-KAN, which uses Chebyshev polynomials as activation functions in KANs. This method improves numerical stability in regression and classification tasks and shows better convergence rates than spline-based KANs on tabular datasets. While this variant offers benefits in addressing numerical problems and minimizing approximation errors, it may entail higher computational expenses during training due to polynomial evaluations and may be less effective for highly non-smooth functions. Another approach, WaveletKAN [6], incorporates wavelet transformations into KANs to efficiently combine local, detailed information with broader trends. Evaluated on the MNIST dataset, this model demonstrated improved handling of multiscale features and greater robustness to noise than the baseline KAN model. While the WaveletKAN model has advantages, such as enhanced representational power for dynamic environments, it has drawbacks, such as higher initialization complexity and sensitivity to wavelet basis selection. These disadvantages could hinder scalability in high-dimensional problems. Another study [7] introduces the KAGNNs model that extends KAN to graph neural networks by incorporating Kolmogorov-Arnold principles for message passing in molecular property prediction. This approach achieves state-of-the-art accuracy on benchmarks, like MoleculeNet. The proposed model offers superior interpretability of graph-structured data and efficient handling of node interactions. Nevertheless, the model's structural nature requires careful tuning of edge functions and may not scale well to very large graphs without optimization.

In the year 2025, Sine-KAN [14] was introduced to replace the learnable grids of B-spline activations in KAN architectures with grids of re-weighted sine functions. This modification demonstrated comparable numerical performance to B-spline-based KANs while significantly improving computational efficiency. Finally, the research [16] developed KAN-Dreamer model, by incorporating KAN and FastKAN models into a RL framework based on DreamerV3 [9]. Evaluated on the DeepMind Control Suite, KAN-Dreamer achieves parameter efficiency and interpretability comparable to MLPs, while maintaining sample efficiency and accelerated inference via radial basis functions in FastKAN. However, this model requires tailored vectorized implementations, which can increase complexity in non-optimized environments.

IV. Methodology

A. TruKAN Overview

We summarize the foundations of TruKAN, introduced in details in [4]. TruKAN extends KAN by substituting the original B-spline with a truncated power basis drawn from the spline theory. The truncated power functions are used as univariate learnable activations. Following the KAN topology, each TruKAN layer contains two main components: 1) a spline-based on truncated power functions, and 2) a basis formed by polynomial terms. The layer computes its output as the sum of these two components. TruKAN architecture can be tailored in several configurations, distinguished by knot placement, knot broadcasting strategy and layer normalization, as explained below:

- **Fixed/Learnable Knot Locations:** Knot locations may be either fixed or learnable. Learnable locations are implemented as a trainable parameter sequence, dynamically optimized during training. This enables the interval density to adapt dynamically to the data distribution rather than relying on fixed priors. However, spline construction requires a monotonic sequence to properly define the curvature, and directly learning the knot locations can violate this monotonicity constraint. To address this issue, a stable reparameterization strategy is employed to ensure that the sequence remains ordered and monotonic throughout training. The knot set is systematically arranged in a grid-like configuration consistent with the structural design of KAN.
- **Knot Broadcasting:** Broadcasting is governed by a single hyperparameter, which determines whether a set of knot sequence is shared across all output channels, or if a distinct set is assigned to each output. With fixed and shared knots, the model aligns with KAN architectures. The broadcasting choice involves a trade-off. With shared knots the parameter count will be reduced and the structural of resulted network will be similar to baseline KAN. In contrast, per-output knot sets configuration enhance model expressivity but require more parameters and may complicate interpretation.
- **Layer Normalization:** Normalization is a regularization method mostly used to make the model more robust in the face of unseen scenarios or advanced noisy data. Due to the instability of truncated power basis functions, normalization is crucial for TruKAN model. There are several normalization methods, such as Layer Normalization, Batch Normalization and Group Normalization. While each method suits different use cases, Layer Normalization is more versatile. The latter standardizes data across each feature dimension for a given training sample. This approach is independent of hyperparameters (such as batch size and number of groups), making it less sensitive to noisy data and suitable for tasks with variable sequence lengths where data distributions may vary across batches during training.

B. Truncated Power Basis Boundary

Although the k-th truncated power basis functions span the same function space as B-splines, they exhibit distinct polynomial growth behavior in CRL settings, where numerical stability is essential. In CRL, online learning occurs without a fixed dataset, and both rewards and value estimates can vary widely and unpredictably. Even with normalized raw observations (zero mean and unit variance), minor drifts in network weights can cause functions of the form $((x-t_i))^k$ to reach magnitudes of 10^3, 10^4, or higher. When raised to the k-th power and linearly combined, these terms can easily overflow, causing numerical instability or producing infinite values during training. This issue is further exacerbated in actor-critic algorithms, where the critic is trained to minimize the temporal difference target, $Q_{\text{target}} = r + \gamma Q_{\text{next}}$,

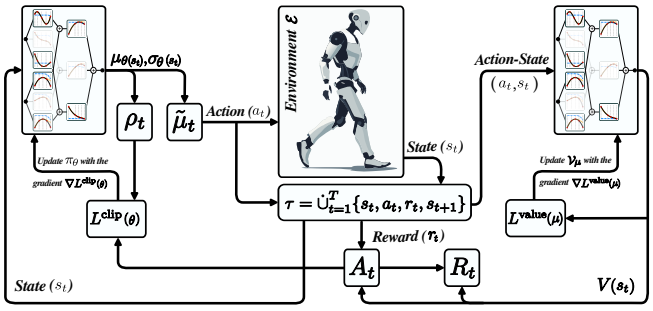

Fig. 1. TruKAN-RPO Framework for Continuous Reinforcement Learning: Incorporating TruKAN into Actor and Networks of RPO.

and if the rewards or bootstrapped Q-values are large, the estimated values can diverge by hundreds, thus destabilizing the learning process.

All previous studies used the original KAN with B-splines as the basis for comparison. Truncated power bases are more susceptible to numerical instability than B-splines, particularly at higher degrees k, due to rounding errors and numerically instability (e.g., near-collinear basis functions for distant knots). This contrasts with the stability and locality of B-splines, which minimize error propagation. In practice, TruKAN may exhibit volatile gradients or slow convergence if the inputs are not normalized or span a wide range and optimization may stall or diverge without careful initialization. In theory, both bases can approximate the same function classes (e.g., continuous piecewise polynomials). However, truncated powers may introduce sharper transitions or less uniform smoothness. Normalization can improve this issue, resulting in outputs that are more adaptive to layer statistics. Also, in CRL (or noisy RL environments), the normalization could improve robustness to noise by regularizing perturbations. While the simplicity of truncated powers could reduce overfitting to spurious patterns, instability could amplify the effects of noise without proper tuning.

Moreover, using RPO can improve sensitivity where RPO counters this by enforcing higher entropy via perturbations, which act as an implicit regularizer. Thus, high-entropy policies are less likely to commit to spurious patterns in noise because they sample from a broader action distribution rather than converging to weak, low-variance outputs. In overall, RPO has demonstrated greater stability and higher returns than vanilla PPO, especially when maintaining entropy levels that improve generalization [13]. Since RPO is a minimal modification to PPO (e.g., adding a scaled uniform noise to the action mean), it can be seamlessly applied to KAN-based policies. The perturbation could regularize KAN's flexible splines, improve the regularization while preserving interpretability.

C. TruKAN-RPO Design

Figure 1 depicts how TruKAN is integrated into the RPO framework for CRL setting. Its general steps are presented in Algorithm 1. With respect to hyperparameters, beyond the ones required by PPO, RPO algorithm introduces an additional scale factor. This factor controls the impact of noise on actions. Furthermore, the TruKAN model incorpo-

Algorithm 1 TruKAN-RPO Framework

Inputs:
Environment $\mathcal{E}$, parameter λ, Discount factor γ, Perturbation scale ζ, $N_{\text{iterations}}$, grid intervals g, Clipping ϵ, polynomial order k

1. Set TruKAN-policy π_θ (parameterized by θ, g and k)
2. Set TruKAN-value function $\mathcal{V}_\mu$ (parameterized by μ, g and k)
3. **WHILE** training is not converging:
   3.1. Gather trajectories $\tau = \dot{\cup}_{t=0}^{T}\{s_t, a_t, r_t, s_{t+1}\}$ from $\mathcal{E}$ using perturbed π_θ
   3.2. **FOR each step** t**:**
      3.2.1. Compute mean action $\mu_\theta(s_t)$ and std $\sigma_\theta(s_t)$ using π_θ
      3.2.2. Sample perturbation $z_t \sim \mathcal{U}(-\zeta, \zeta)$ (element-wise for vector actions)
      3.2.3. Set perturbed mean $\tilde{\mu}_t = \mu_\theta(s_t) + z_t$
      3.2.4. Sample action $a_t \sim \mathcal{N}(\tilde{\mu}_t, \sigma_\theta(s_t))$
   ENDFOR
   3.3. Calculate advantages $A_t = \sum_{i=0}^{T} (\gamma\lambda)^i \, \delta_{t+i}$ using λ
   3.4. Calculate returns $R_t = A_t + V(s_t)$
   3.5. **FOR epoch** $= 1$ **to** $N_{\text{iterations}}$**:**
      3.5.1. Calculate ratio using unperturbed distribution for log-prob $\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
      3.5.2. Calculate clipped policy loss $L^{\text{clip}}(\theta) = E_t\left[\min(\rho_t A_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A_t)\right]$
      3.5.3. Calculate clipped value $V^{\text{clipped}} = V_{\text{old}}(s_t) + \text{clip}(R_t - V_{\text{old}}(s_t), -\epsilon, \epsilon)$
      3.5.4. Calculate clipped value loss with $\mathcal{V}_\mu$ $L^{\text{clipped value}}(\mu) = \frac{E_t\left[\max((V(s_t)-R_t)^2, (V(s_t)-V^{\text{clipped}}(s_t))^2)\right]}{2}$
      3.5.5. Adapt π_θ with gradients of $L^{\text{clip}}(\theta)$
      3.5.6. Adapt $\mathcal{V}_\mu$ with gradients of $L^{\text{value}}(\mu)$
   ENDFOR
ENDWHILE

rates three specific hyperparameters: 1) the number of knots which determines the fineness of the input domain division for approximating the activation function, 2) a boolean flag which specifies whether knot locations remain fixed or are learned during training, and 3) the order of the piecewise polynomial representing the degree of the polynomial function in each segment of the truncated power functions, which will governs the smoothness of the splines at the junctions between segments.

In the context of KAN models, these additional hyperparameters apply to standard KAN and SineKAN models, with one exception: the Boolean flag. Standard KAN and SineKAN models operate on fixed intervals, and the locations of the knots (called "grid" in KAN and SineKAN terminology) are predefined. Consequently, this hyperparameter is unnecessary for these models.

V. Experiment Setup

A. MuJoCo-MJX Humanoid Environment

We evaluate the numerous CRL models using the "MuJoCo-MJX Humanoid" [18] robotic model from the MuJoCo Playground framework [1]. This environment (built on the MJX [2]) is used to evaluate algorithms for CRL systems in massively parallel training and simulation-to-real transfer scenarios. The agent is a complex, bipedal humanoid figure with human-like kinematics. It is tasked with achieving efficient forward locomotion while maintaining balance and an upright posture dynamically through coordinated, multi-limb movements. The observation space comprises a unified set of features across locomotion tasks, including gravity projected in the body frame, joint positions and their velocities, linear and angular velocities and foot phase variables for each foot, which enable gait synchronization. The action space consists of continuous torque values applied to the actuated joints. The agent receives a composite reward formed as a weighted sum of clipped, nonnegative terms, including exponential penalties for tracking errors in linear and angular velocities and bonuses for sufficient foot airtime and clearance, phase synchronization, reduced foot slip, upright orientation, and nominal joint poses. There are also penalties for excessive joint torques, action rates, energy consumption, termination events, and undesired vertical or lateral velocities. This combination encourages the development of stable, efficient, and human-like walking or running behaviors without falling. All experiments are evaluated using version 3.4 of the Mujoco library.

B. JAX Framework

We conduct all the experiments using the JAX framework for training our models. The primary structure of RPO was adapted from the Brax library [8]. For each task, we employ the latest available environment versions to maintain compatibility with current state-of-the-art results. We measure the model performance under RPO using the average episodic return, computed over a fixed set of evaluation episodes. Under JAX+MJX, we use 4096 parallel environments (and agents) for training and 128 parallel environments (and agents) for evaluation. While highly parallel environments provide massive amounts of data for each step, we apply the mini-batch technique to the collected data, using 32 mini-batches for each step. The policy is updated after every eight mini-batches, meaning the policy is updated four times for each collected data set per step.

Regarding the network architectures, we design three-layer networks for the actor and critic components. The number of neurons per layer varies based on each KAN model to maintain an overall consistent number of learnable parameters for all the experiments (approximately 50,000 parameters for the policy network and 38,000 for the value network). We use identical configurations for actor and critic parts in all variations.

C. Training Optimization

To mitigate the impact of randomness of network initialization (i.e. weight and bias) and inherent stochasticity of RPO algorithm (via exploratory noise), each experimental configuration is executed with five distinct random seeds. Performance curves are then averaged across these five trials to improve the stability of the results. Each experiment is executed for a duration of 32 million environment steps in

[1] https://playground.mujoco.org
[2] JAX-accelerated MuJoCo simulator

TABLE I

FINE-TUNING OF HYPERPARAMETERS

Training Parameters	
Learning decay rate δ	1e-4
$N_{\text{iterations}}$	32e6
Batch size	256
RPO Parameters	
Discount factor γ	0.97
GAE parameter λ	0.95
Clipping factor ϵ	0.2
TruKAN Parameters	
Polynomial's order k	3
Number of knots (Grid)	5

total, with evaluations conducted after every one million step. The evaluation process involves the calculation of the mean values of five episodes, with each episode restricted to a maximum of 1,000 steps. We note that the episodes were terminated after reaching this threshold.

To ensure a level playing field and replicability of the results, a set of constant hyperparameters is employed, remaining consistent across all experimental conditions. The value of the RPO's noise scale factor varies based on the complexity of the environment and exploration/exploitation needs. The noise factor for the HumanoidStand, HumanoidWalk and HumanoidRun tasks is set to 0.001, 0.1, and 0.2, respectively. Similarly, we use different values for the entropy cost factor. We set the value to $3e-3$ for the HumanoidStand task and to $2e-3$ for the other two tasks.

The discount factor is set to 0.97, and clipped objective uses a clipping factor of 0.2 to restrict policy updates. We train the models by minimizing their objective functions with the AdamW optimizer; $\beta_1 = 0.9$, $\beta_2 = 0.99$, and weight decay of 1×10^{-4} [12]. The learning rate remains fixed during training, but varies across environments. For the HumanoidStand task, we select a smaller learning rate value compared to two other tasks and train the policy and value networks with a learning rate set to $\alpha = 2 \times 10^{-5}$. For the HumanoidWalk and HumanoidRun tasks, we increase this value to 1×10^{-3}.

For the TruKAN model with learnable knot locations, standard learning rate values (typically ranging from 1×10^{-3} to 3×10^{-4}) proved suitable for optimizing coefficients. However, these values were excessively large for adjusting knot locations in small range (e.g. $[-1, 1]$). Such rates induced instability in adaptation and degraded network performance. Consequently, the TruKAN model with learnable knots employed two separate optimizers: one for coefficients and other parameters, and another one for knots. The learning rate for the latter optimizer is set to one-tenth of that for the former one, 2×10^{-6} for HumanoidStand and 1×10^{-4} for HumanoidWalk and HumanoidRun. Compared to classical PPO, the policy and value functions in RPO are based on KAN, parameterized by θ and μ. All the models employ learnable activations, which are defined with a set of coefficients, θ and μ. Table I presents the hyperparameter's values for general training, RPO and TruKAN, and these values remain constant for all the models.

VI. VALIDATION

As reported in Figure 2, we evaluate and compare two variants of TruKAN-RPO (TruKAN-RPO-Fixed and TruKAN-RPO-Learnable) against two other models, KAN-RPO and SineKAN-RPO. The objective of "HumanoidStand" task is for the agent to maintain a stable, upright posture with no horizontal movement, despite gravitational forces and minor environmental perturbations that render passive standing infeasible without active control. In this task, both TruKAN variants and SineKAN achieved the maximum reward, whereas KAN-RPO lagged behind with an average score below half of the maximum. Notably, although the TruKAN variants converged more rapidly than SineKAN, their performance declined more quickly after convergence. A similar degradation pattern was also observed in the SineKAN models; however, because SineKAN converged more slowly, a larger number of training iterations was required before experiencing the same type of performance decline observed in the TruKAN variants.

For the "HumanoidWalk" task, the objective extends the "HumanoidStand" requirements to include forward locomotion at a moderate velocity (target of approximately 1 unit per second). Here, the performances of all models were comparable; TruKAN with learnable knots yielded the highest results. The TruKAN variant (with fixed knots) and SineKAN exhibited nearly identical performance, while KAN model again recorded the lowest score. The objective of "HumanoidRun" task parallels that of "HumanoidWalk" but requires forward locomotion at a higher velocity (target of approximately 10 units per second) while maintaining upright stability. The results for this task reveal smaller performance differences among the models compared to the prior tasks, with SineKAN exhibiting steady improvement. The TruKAN variants demonstrated stronger initial performance but converged to final scores similar to SineKAN's score. While KAN performed comparably to TruKAN models at the outset, its final score lagged behind and was the lowest overall. Across all tasks, the TruKAN models achieved higher or comparable scores relative to the SineKAN, whereas KAN consistently recorded the worst performance. However, the observed patterns indicate that the TruKAN models may require a distinct set of hyperparameters, as the truncated power basis appears to promote greater exploration compared to other models with similar hyperparameter set.

VII. DISCUSSION

The degradation in returns after reaching peak performance in the "HumanoidStand" task highlights a known vulnerability of highly expressive, dynamic architectures in narrow-distribution continuous control. Standing is an unstable equilibrium task and once the policy achieves a highly rewarding posture, the state distribution it experiences becomes extremely narrow. Because KANs dynamically adapt their basis functions to current data distribution, the network can rapidly overfit to this localized "standing" state. This pattern suggests that models exhibiting faster convergence may require smaller learning rates, along with additional

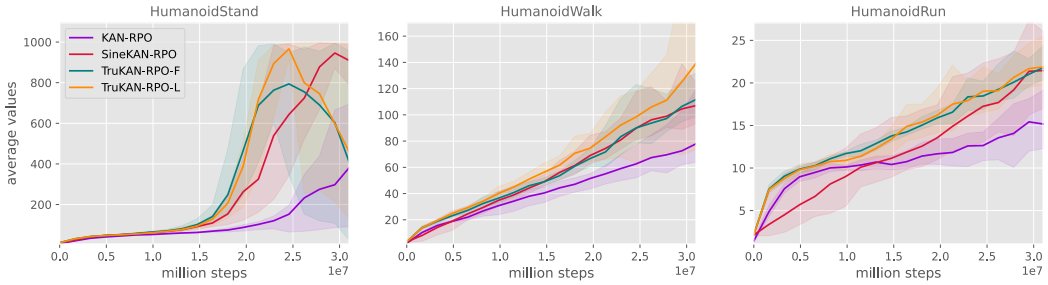

Fig. 2. Performance of TruKAN-RPO-Fixed, TruKAN-RPO-Learnable, KAN-RPO and SineKAN-RPO on three configurations of "MuJoCo-MJX Humanoid". The curves are the average Return over five runs with distinct random seeds. The shaded areas are complete range of Returns observed across trials.

regularization techniques, to mitigate overfitting and improve generalization performance. Despite the instability in the standing task, the learning curves across all environments strongly suggest that the core gains of TruKAN stem from a superior balance of exploration and exploitation. Unlike pure B-spline KAN and SineKAN that can oscillate wildly early in training, the truncated power basis provides a smoother, non-saturating gradient landscape. In the figures, this is evidenced by the significantly steeper initial climb of the TruKAN variants in "HumanoidWalk" and "HumanoidRun". This rapid early ascent demonstrates highly efficient exploration where the basis functions allow the agent to quickly map the broad state-action space and identify high-reward regions.

Regarding the TruKAN variants, while all model parameters were kept identical except for the knot configurations, the observed differences in performance can be attributed primarily to the knot properties. The variant with learnable knots demonstrated superior performance by adaptively adjusting its grid parameters to maximize performance along the identified optimal trajectory, often achieving higher peak results than the fixed-knot variant. In contrast, the fixed-knot variant exhibited a more stable and regularized learning behavior, which acted as a safeguard against the overfitting observed in the standing task.

VIII. Conclusions

This study presented the TruKAN-RPO framework by integrating TruKAN variant into the RPO architecture. We replaced traditional MLPs with TruKAN models in actor and critic networks of RPO to exploit the function approximation properties of TruKAN in CRL tasks. We conducted an empirical evaluation of two TruKAN-RPO variants (fixed vs. learnable knot locations) in the complex "MuJoCo-MJX Humanoid" environment and compared them with KAN-RPO and SineKAN-RPO models. The results indicate that TruKAN improves the learning performance of on-policy RL methods by facilitating more efficient function approximation. Specifically, the results suggest that the TruKAN model better handles the trade-off between exploration and exploitation compared to other models.

References

[1] Bayeh, A., Mouhoub, M., Sadaoui, S.: Enhancing Off-Policy Method SAC with KAN for Continuous Reinforcement Learning. In: Proc. of International Conference on Deep Learning Theory and Applications DeLTA. pp. 229–239. Springer (2025)

[2] Bayeh, A., Mouhoub, M., Sadaoui, S.: KAN-based On-policy PPO Method for Continuous Reinforcement Learning. In: In: Dorronsoro, B., Talbi, EG., Weraikat, D., Bouamama, S. (eds) Optimization and Learning. OLA 2025. Communications in Computer and Information Science, vol 2808. (2026)

[3] Bayeh, A., Sadaoui, S., Mouhoub, M.: SineKAN-based SAC Approaches for Continuous Reinforcement Learning Environments. In: 10th International Conference on Machine Learning and Soft Computing, ICMLSC (2026)

[4] Bayeh, A., Sadaoui, S., Mouhoub, M.: TruKAN: Towards More Efficient Kolmogorov-Arnold Networks Using Truncated Power Functions. In: arXiv pre-print, arXiv.2602.03879 (2026), https://arxiv.org/abs/2602.03879

[5] de Boor, C.: A Practical Guide to Splines, Applied Mathematical Sciences, vol. 27. Springer-Verlag, New York (1978)

[6] Bozorgasl, Z., Chen, H.: Wav-KAN: Wavelet Kolmogorov-Arnold Networks. arXiv 2405.12832, cs.LG (2024)

[7] Bresson, R., Nikolentzos, G., Panagopoulos, G., Chatzianastasis, M., Pang, J., Vazirgiannis, M.: KAGNNs: Kolmogorov-Arnold Networks meet Graph Learning. Transactions on Machine Learning Research (2024)

[8] Freeman, C.D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., Bachem, O.: Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation (2021), http://github.com/google/brax

[9] Hafner, D., Pasukonis, J., Ba, J., Lillicrap, T.: Mastering diverse control tasks through world models. Nature 640 pp. 1–7 (2025)

[10] Liu, Z., Ma, P., Wang, Y., Matusik, W., Tegmark, M.: Kolmogorov-arnold networks meet science. Physical Review X **15**(4), 041051 (2025). https://doi.org/10.1103/4t7t-v19l

[11] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T.Y., Tegmark, M.: KAN: Kolmogorov–Arnold Networks. In: The Thirteenth International Conference on Learning Representations, ICLR (2025)

[12] Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization. In: 7th International Conference on Learning Representations, ICLR (2019)

[13] Rahman, M.M., Xue, Y.: Robust policy optimization in deep reinforcement learning (2023), https://openreview.net/forum?id=HnLFY8F9uS

[14] Reinhardt, E., Ramakrishnan, D., Gleyzer, S.: SineKAN: Kolmogorov-Arnold Networks using Sinusoidal Activation Functions. Frontiers in Artificial Intelligence **7**, 1462952 (2025)

[15] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. In: Proceedings of the 34th International Conference on Machine Learning (ICML). pp. 1374–1382 (2017)

[16] Shi, C., Luan, X.: KAN-Dreamer: Benchmarking Kolmogorov-Arnold Networks as Function Approximators in World Models. arXiv 2512.07437 (2025), https://arxiv.org/abs/2512.07437

[17] Sidharth, S., Keerthana, A., Gokul, R., Anas, K.: Chebyshev Polynomial-Based Kolmogorov-Arnold Networks: An Efficient Architecture for Nonlinear Function Approximation. arXiv preprint arXiv:2405.07200 (2024), https://arxiv.org/abs/2405.07200

[18] Zakka, K., Tabanpour, B., Liao, Q., Haiderbhai, M., Holt, S., Luo, J.Y., Allshire, A., Frey, E., Sreenath, K., Kahrs, L.A., Sferrazza, C., Tassa, Y., Abbeel, P.: Demonstrating mujoco playground. In: Robotics: Science and Systems (RSS) (2025)