
Sample-Efficiency of Kolmogorov–Arnold Networks in DeepRL

Kevin Riehl

ETH Zürich, IVT &
Agentic Systems Lab (ASL)
kriehl@ethz.ch

Shaimaa K. El-Baklish

ETH Zürich, IVT
selbaklish@ethz.ch

Fan Wu

ETH Zürich, Agentic Systems Lab (ASL)
fanwu@ethz.ch

Anastasios Kouvelas

ETH Zürich, IVT
kouvelas@ethz.ch

Abstract

Deep reinforcement learning has achieved substantial performance gains over classical control approaches. Yet, a central challenge to learning in real-world applications is acquiring costly samples. Kolmogorov-Arnold Networks are a recently proposed architecture that can learn physical relationships in control problems effectively, with significantly higher parameter efficiency and interpretability when compared to Multi-Layer-Perceptron architectures. In this work, we systematically study sample-efficiency using computational experiments, covering the Feynman dataset and the Gymnasium RL benchmark. The results show that similar performance can be achieved with 40% fewer samples using the Kolmogorov-Arnold architecture, and that relative performance improvements up to 50% occur during the training process. The observed gains are robust to varying levels of noise in rewards. These results highlight the potential of the Kolmogorov-Arnold architectures for more sample-efficient reinforcement learning.

Code: https://github.com/DerKevinRiehl/neurips26_kan_training

1 Introduction

Reinforcement learning (RL) has become a powerful paradigm for learning feedback controllers directly from system interactions [1]. Classical control methods, including model-based and model predictive control, can be highly effective when accurate model dynamics and tractable objectives are available, but these assumptions are often difficult to satisfy in complex, nonlinear, and noisy environments [2]. Value-based RL methods such as Q-learning instead learn an action-value function that estimates the long-term return of taking an action in a given state [3]. Deep RL extends this idea by replacing tabular value functions or policies with neural networks, enabling controllers to approximate high-dimensional and nonlinear state-action relationships [4, 5]. This flexibility has made neural RL attractive for control tasks in robotics, networking, and other cyber-physical systems where the controller must cope with sensor noise, nonlinear dynamics, delayed effects, and changing operating conditions.

For RL-based control, the dominant practical challenge is sample efficiency: learning a useful controller often requires many environment interactions [6]. Unlike labeled examples in supervised learning, each control sample is a transition generated by running a policy in a simulator, digital twin, or physical system [7]. This makes exploration costly, since the agent must discover better actions while maintaining stable behavior under a changing data distribution [8, 9]. In safety-critical

settings, this challenge is amplified because neural policies are difficult to inspect, verify, or translate into controller equations [10]. These constraints motivate function approximators that learn strong policies from fewer interactions while retaining enough structure for analysis and interpretation [11].

Kolmogorov-Arnold Networks (KANs) are a recently proposed class of neural network architectures inspired by the Kolmogorov-Arnold representation theorem, which states that any continuous multivariate function can be expressed as a finite superposition of univariate functions and summation operators [12, 13]. Unlike traditional neural networks such as multilayer perceptrons (MLPs), which rely on linear transformations followed by fixed nonlinear activation functions, KANs replace scalar weights with learnable univariate functions defined along the edges of the network [14]. This formulation allows the model to directly learn functional relationships between variables rather than approximating them only through weighted sums of fixed activation functions. As a result, KANs offer improved interpretability and significantly higher parameter efficiency while retaining strong approximation capabilities. Recently, KANs have gained considerable attention, particularly in areas where understanding the underlying functional structure of the data is important [15, 16, 17]. KANs are therefore particularly well suited for applications such as symbolic regression, scientific discovery, and physics-informed modeling, where both data efficiency and model transparency are essential [15, 16]. These properties have also motivated the use of KANs in RL-based control, including PPO-based continuous control, robotic policy learning, load balancing, and explainable RL [18, 19, 20, 21, 22].

Existing evidence on the efficiency of KANs is promising. Prior work reports that smaller KANs can match or outperform larger MLPs in several data-fitting and scientific-computing examples, suggesting favorable neural scaling behavior and parameter efficiency [14]. In structured settings, KANs have been shown to reduce the need for expensive high-fidelity data and to perform well in low-data industrial fault detection [23, 24, 25]. Theoretical work also gives sample-complexity guarantees for residual KANs under smoothness and noiseless-sample assumptions [26]. Yet, in more systematic low-data comparisons, MLPs with individualized learnable activations can outperform KANs when only a few hundred samples are available [27]. These mixed findings, together with the absence of a systematic assessment of sample efficiency in RL-based control applications, motivate this work: it remains unclear when KANs improve sample efficiency over MLPs under comparable experimental budgets.

Using the Feynman dataset (a collection of physical equations) and the Gymnasium RL benchmark (a collection of RL environments) [28] we systematically assess sample efficiency of KANs as function approximators when compared to MLPs. Our results show that 40% fewer samples are required to achieve similar performance, and that 50% reward gains can be achieved during the training process using KANs.

2 Related Works

Kolmogorov–Arnold networks and learning guarantees. KANs were introduced as an alternative to MLPs in which learnable univariate functions are placed on edges rather than fixed nonlinearities on nodes [14]. This architecture has motivated rapid follow-up work because it connects neural approximation to interpretable function decompositions and can use fewer parameters in some scientific-computing and symbolic-regression settings [15, 16]. Theoretical work has begun to formalize these intuitions by proving approximation and sample-complexity guarantees for residual KANs on smooth function classes (including Besov spaces) [26]. These guarantees are important, but they are typically stated under assumptions that differ from RL control, where targets are noisy, bootstrapped, and induced by a changing policy.

Low-data and fixed-sample empirical comparisons. Empirical evidence for KAN sample efficiency remains mixed. The original KAN experiments show favorable scaling and strong performance with compact models, but they do not isolate environment-sample efficiency under fixed compute and fixed interaction budgets [14]. Follow-up works directly study low-data regimes and find that MLPs with individualized learnable activations can outperform KANs with substantially fewer parameters, especially when only a few hundred samples are available [27]. Further studies show that KANs and MLPs trade advantages across regular, irregular, discontinuous, and noisy functions, with KANs converging quickly in some cases but failing to dominate across stabilized test loss [29]. These comparisons suggest KANs should not be treated as universally more sample-efficient than MLPs.

Structured and expensive-data settings. More positive results appear when the problem structure matches the inductive bias of KAN edge functions. Some works propose multifidelity KANs that combine abundant low-fidelity data with limited high-fidelity data, reducing the amount of expensive high-fidelity data needed in scientific machine learning [23]. Furthermore, studies evaluate KAN autoencoders for fault detection across varying training-set sizes and find that several KAN variants match or surpass an MLP-based orthogonal autoencoder with substantially less data [24]. These results are highly relevant for control domains with costly simulation or measurement, but they are supervised or unsupervised learning studies rather than online RL studies.

KANs in RL and control. Several recent works integrate KANs into RL pipelines. Prior works replace MLP actor and critic components in PPO with KANs and report comparable performance with fewer parameters on DeepMind Control Proprio robotics benchmarks [18]. Similarly, studies propose KAN-PPO and MultKAN-PPO for continuous control, finding robust performance across several configurations while noting sensitivity to dropout design [19]. Other studies focus on KAN-based PPO agents for CartPole and radio-access-network control, showing that KAN agents can match MLP performance with far fewer parameters and can be converted into sparse symbolic policies [22]. Moreover, works propose a KAN actor in PPO for load balancing and emphasize the ability to extract interpretable controller equations [21]. Together, these works support the feasibility of KAN-based RL, but they primarily emphasize parameter efficiency and interpretability rather than controlled sample-efficiency comparisons.

KANs for robotic policies and world models. KANs have also been used in policy-learning systems outside the standard actor-critic replacement setting. KAN Policy can be used for diffusion-based visuomotor policy learning, and works doing so report smoother trajectories, shorter execution times, and improved success rates relative to diffusion-policy baselines [20]. *QuadKAN* for vision-guided quadruped locomotion reports faster convergence, improved returns, lower action jitter, and better robustness in structured gait-control settings [25]. *KAN-Dreamer* [30] replaces selected components with KAN or FastKAN [31] modules, finding that FastKAN is promising for reward and continue prediction but that direct replacement of visual encoders or actor-critic modules can harm sample efficiency. This component-level evidence reinforces the need to distinguish where KANs are useful in a RL stack rather than assuming that every MLP can be replaced beneficially.

3 Experiments

3.1 Feynman Sample-Efficiency Benchmark

The first experiment tests the core approximation question in a controlled supervised setting, before moving to full RL control. If KANs are more sample-efficient because their edge functions provide a useful inductive bias, this advantage should already be visible when learning smooth physical laws from few samples. For this purpose, we follow Liu et al. and use the Feynman benchmark, which contains closed-form physics equations derived from the Feynman Lectures and related symbolic-regression benchmarks [14, 32, 33]. Each benchmark task is a regression problem, where each Feynman Lecture equation is independently sampled to generate training sets (varying size) and test sets (fixed size). The primary evaluation metric is normalized root mean squared error (NRMSE).

3.2 RL Gymnasium Benchmark

The second experiment studies the sample efficiency of KANs and MLPs in the context of DeepRL for control. For this purpose, we use the RL benchmark "*Gymnasium*" [28], which contains multiple different physics-based environments to study RL. We chose the subset of "Classic Control" as state variables are used for learning, rather than image or video streams, as KANs are known to be particularly suitable to learn physical relationships. Furthermore, we chose the proximal policy optimization (PPO) learning strategy, as it is a standard, robust policy-gradient method for continuous and discrete control problems [34]. The benchmark includes the following environments: "Acrobot-v1", "CartPole-v1", "MountainCar-v0", "MountainCarContinuous-v0", and "Pendulum-v1". Each benchmark task provides a comparable learning curve with best rewards achievable (convergence limits) enabling a comparison.

3.3 RL Gymnasium Benchmark With Noised Rewards

The third experiment studies the robustness of the sample efficiency gains of KANs over MLPs in the context of DeepRL in control applications, for varying reward (Gaussian) noise levels.

3.4 Comparison and Sample-Efficiency Assessment

Different model complexities and training parameters were explored via grid-search to identify optimal performance. KAN and MLP models were matched by model complexity (number of parameters) to ensure a fair comparison. Number of training samples to achieve a certain performance were used to assess the enhanced sample-efficiency of KANs over MLPs. Specifically, we compare ReLU and Sigmoid for MLP, and B-Splines and radial basis functions (RBF) for KAN networks, to cover a multitude of different function space shapes.

4 Results

The Feynman experiment shows a clear low-sample crossover in favor of KANs after 50 training samples (Fig. 1). The best KAN family reduces mean NRMSE from 0.200 to 0.163 (at 50 samples) and from 0.127 to 0.060 (at 1000 samples), corresponding to a relative improvement from 13.9% (at 50 samples) to 49.3% (at 1000 samples). Equivalently, KANs reach several MLP error levels with substantially fewer training samples once the target error is below the early-data regime.

The Gymnasium experiment (Fig. 2) reproduces the same pattern in control applications, although the size of the effect depends strongly on the environment. Across Acrobot, CartPole, MountainCar, and Pendulum, the advantage of KANs peaks at about 41.6% (2250 episodes) and then narrows to about 7.5% (final checkpoint) on average. The sample-reduction curves show that that similar performances can be achieved with about 40% fewer samples using KANs. After many training episodes, both architectures approach similar return levels, demonstrating the faster learning ability of KANs over MLPs. The noisy-reward benchmark indicates that these gains are not an artifact of clean reward feedback. Adding Gaussian reward noise increases variance, but the KAN and MLP comparison keeps the same qualitative shape across the tested noise scales (Fig. 5).

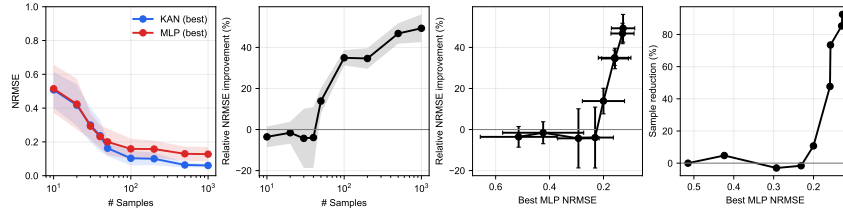


Figure 1: Experiment 1: Sample Efficiency in Feynman Benchmark.

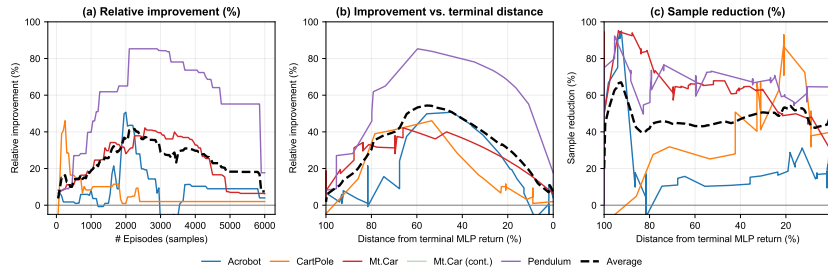


Figure 2: Experiment 2: Sample Efficiency in Gymnasium Benchmark.

5 Discussion

Overall, the experiments highlight that KAN architectures improve sample efficiency when the task contains smooth low-dimensional structure. The strongest gains occur in the early and middle part of

training, which is the regime that matters most when simulator calls, measurements, or hardware trials are expensive. The effect is not uniform across tasks, and the final returns are often close once both model architectures receive many samples. The results suggest that KANs can be useful for low-data control problems. A practical route for deployment is to train with KANs during the scarce-sample phase and then distill the learned policy into a cheaper MLP when inference cost is the bottleneck.

References

- [1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.
- [2] Bahare Kiumarsi, Kyriakos G Vamvoudakis, Hamidreza Modares, and Frank L Lewis. Optimal and autonomous control using reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2042–2062, 2017.
- [3] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
- [4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv Preprints*, 2017.
- [6] Junzi Zhang, Jongho Kim, Brendan O’Donoghue, and Stephen Boyd. Sample efficient reinforcement learning with reinforce. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10887–10895, 2021.
- [7] Fei Tao, Bin Xiao, Qinglin Qi, Jiangfeng Cheng, and Ping Ji. Digital twin modeling. *Journal of Manufacturing Systems*, 64:372–389, 2022.
- [8] Chuheng Zhang, Yuanying Cai, Longbo Huang, and Jian Li. Exploration by maximizing Rényi entropy for reward-free rl framework. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10859–10867, 2021.
- [9] Jonas Eschmann. Reward function design in reinforcement learning. *Reinforcement learning algorithms: Analysis and Applications*, pages 25–33, 2021.
- [10] Peilang Li, Umer Siddique, and Yongcan Cao. From explainability to interpretability: Interpretable policies in reinforcement learning via model explanation. *arXiv Preprints*, 2025.
- [11] Silvia Ferrari and Robert F Stengel. Smooth function approximation using neural networks. *IEEE Transactions on Neural Networks*, 16(1):24–38, 2005.
- [12] Andrey N. Kolmogorov. On the representation of continuous functions of several variables by superpositions of continuous functions of fewer variables. *Doklady Akademii Nauk SSSR*, 108:179–182, 1956.
- [13] Vladimir I. Arnold. On functions of three variables. *Doklady Akademii Nauk SSSR*, 114:679–681, 1963.
- [14] Ziming Liu, Yifan Wang, Shashank Vaidya, Fabian Ruehle, James Halverson, Marin Soljagic, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov-Arnold networks. *arXiv Preprints*, 2024.
- [15] Shriyank Somvanshi, Syed Aaqib Javed, Md Monzurul Islam, Diwas Pandit, and Subasish Das. A survey on Kolmogorov-Arnold networks. *ACM Computing Surveys*, 58(2):1–35, 2025.
- [16] Siham Essahraoui, Ismail Lamaakal, Khalid El Makkaoui, Ibrahim Ouahbi, Mouncef Filali Bouami, and Yassine Maleh. Kolmogorov-Arnold Networks: Overview of architectures and use cases. In *International Conference on Circuits, Systems and Communication*, 2025.
- [17] Shairoz Sohail. On training of Kolmogorov-Arnold networks. *arXiv Preprints*, 2024.
- [18] Victor A. Kich, Jair A. Bottega, Raul Steinmetz, Ricardo B. Grando, Ayano Yoroazu, and Akihisa Ohya. Kolmogorov-Arnold networks for online reinforcement learning. In *2024 24th International Conference on Control, Automation and Systems (ICCAS)*, pages 958–963. IEEE, 2024.

- [19] Ali Bayeh, Malek Mouhoub, and Samira Sadaoui. KAN-based on-policy ppo method for continuous reinforcement learning. In *Optimization and Learning: 8th International Conference, OLA 2025, Revised Selected Papers*, pages 3–15. Springer Nature Switzerland, 2026.
- [20] Zikang Chen, Fei Gao, Ziya Yu, and Peng Li. KAN policy: Learning efficient and smooth robotic trajectories via Kolmogorov-Arnold networks. *IEEE Robotics and Automation Letters*, 10(11):11164–11171, 2025.
- [21] Kamal Singh, Sami Marouani, Ahmad Al Sheikh, Pham Tran Anh Quang, and Amaury Habrard. Interpretable reinforcement learning for load balancing using Kolmogorov-Arnold networks. In *2026 IEEE 23rd Consumer Communications and Networking Conference (CCNC)*. IEEE, 2026.
- [22] Michele Antonio D’Alto. *Kolmogorov-Arnold Networks for Explainable Reinforcement Learning: A Symbolic Approach to Function Approximation in Reinforcement Learning*. PhD thesis, KTH Royal Institute of Technology, 2025.
- [23] Amanda A. Howard, Bruno Jacob, and Panos Stinis. Multifidelity Kolmogorov-Arnold networks. *arXiv Preprints*, 2024.
- [24] Enrique Luna Villagomez and Vladimir Mahalec. Comparative evaluation of Kolmogorov-Arnold autoencoders and orthogonal autoencoders for fault detection with varying training set sizes. *arXiv Preprints*, 2025.
- [25] Allen Wang and Gavin Tao. QuadKAN: KAN-enhanced quadruped motion control via end-to-end reinforcement learning. *arXiv Preprints*, 2025.
- [26] Anastasis Kratsios, Bum Jun Kim, and Takashi Furuya. Approximation rates in besov norms and sample-complexity of Kolmogorov-Arnold networks with residual connections. *Neural Networks*, page 108797, 2026.
- [27] Farhad Pourkamali-Anaraki. Kolmogorov-Arnold networks in low-data regimes: A comparative study with multilayer perceptrons. In *2025 International Conference on Machine Learning and Applications (ICMLA)*, pages 923–928. IEEE, 2025.
- [28] Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun Kg, Rodrigo Perez-Vicente, et al. Gymnasium: A standard interface for reinforcement learning environments. *Advances in Neural Information Processing Systems*, 38, 2026.
- [29] Chen Zeng, Jiahui Wang, Haoran Shen, and Qiao Wang. KAN versus MLP on irregular or noisy functions. *arXiv Preprints*, 2024.
- [30] Chenwei Shi and Xueyu Luan. KAN-Dreamer: Benchmarking Kolmogorov-Arnold networks as function approximators in world models. *arXiv Preprints*, 2025.
- [31] Ziyao Li. Kolmogorov-Arnold networks are radial basis function networks. *arXiv Preprints*, 2024.
- [32] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.
- [33] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. In *Advances in Neural Information Processing Systems*, volume 33, pages 4860–4871, 2020.
- [34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv Preprints*, 2017.
- [35] Ziming Liu, Pingchuan Ma, Yixuan Wang, Wojciech Matusik, and Max Tegmark. KAN 2.0: Kolmogorov-Arnold networks meet science. *arXiv Preprints*, 2024.
- [36] Hoang-Thang Ta. BSRBF-KAN: A combination of B-splines and radial basis functions in Kolmogorov-Arnold networks. In *International Symposium on Information and Communication Technology*, pages 3–15. Springer, 2024.

- [37] Hoang-Thang Ta, Duy-Quy Thai, Abu Bakar Siddiquir Rahman, Grigori Sidorov, and Alexander Gelbukh. FC-KAN: Function combinations in Kolmogorov-Arnold networks. *arXiv Preprints*, 2024.
- [38] Hoang-Thang Ta, Duy-Quy Thai, Anh Tran, Grigori Sidorov, and Alexander Gelbukh. PRKAN: Parameter-reduced kolmogorov-Arnold networks. *arXiv Preprints*, 2025.
- [39] Hoang-Thang Ta and Anh Tran. AF-KAN: Activation function-based Kolmogorov-Arnold networks for efficient representation learning. *arXiv Preprints*, 2025.

A Experiment Setup Details

A.1 Feynman Benchmark Details

A.1.1 Dataset Construction

The Feynman benchmark consists of analytic equations with named input variables and prescribed sampling ranges [32, 33]. Following Liu et al., we evaluate on 27 implemented ‘Feynman_no_units’ equations for which finite train and test samples can be generated [14]. This includes univariate equations, which are reported separately from multivariate equations in the aggregate analysis. We use the unit-free subset, because univariate equations provide a sanity check for spline-based edge learning, while multivariate equations test whether KANs retain any sample-efficiency advantage when the model must learn interactions between variables. For each equation $f_j : \mathbb{R}^{d_j} \rightarrow \mathbb{R}$, we generate supervised examples by sampling $x \sim \mathcal{U}(\mathcal{D}_j)$ and evaluating $y = f_j(x)$. The domain $\mathcal{D}_j$ is the Cartesian product of the variable intervals specified by the benchmark. For each equation, one fixed train pool and one fixed test set are sampled, and model initializations are varied across random seeds. If an equation is undefined at a sampled point or returns a non-finite value, we reject the point and resample until the desired split size is reached. This rejection step is applied before normalization and is identical for all models.

A.1.2 Sample-Efficiency Protocol

For each equation, we fix $N_{\text{test}} = 1000$. We vary the training size over

$$N_{\text{train}} \in \{10, 20, 30, 40, 50, 100, 200, 500, 1000\}. \quad (1)$$

. For each N_{train} , equation, and model family, we repeat the experiment over 20 random seeds. Each seed determines the training draw, the test draw, and the model initialization. Inputs are standardized using the mean and standard deviation of the training set. Targets may be standardized during optimization, but all reported metrics are computed after transforming predictions back to the original target scale. No information from the test set is used for model selection or normalization.

A.1.3 Model Comparisons

The main comparison includes KAN and MLP regressors trained with the same optimizer, loss, maximum number of update steps, batch size, and random seeds. For KANs, we use spline-based edge functions as in the original KAN implementation, with spline order and grid size fixed across equations unless otherwise stated. For MLPs, we compare ReLU and sigmoid activations, to avoid giving KANs an artificial advantage over nonsmooth baselines on smooth physics equations. We report trainable parameter counts and wall-clock training time for every trained model. The primary setting uses a fixed architecture recipe for each model family. A secondary parameter-matched setting chooses MLP widths so that the number of trainable parameters is as close as possible to the KAN parameter count. When exact matching is impossible, we report the parameter ratio alongside the result. This protocol avoids treating parameter count as interchangeable across architectures while still checking whether apparent sample-efficiency gains are explained by model size.

A.1.4 KAN Implementations

To validate KAN baselines, we use the public All-KAN implementation repository as the implementation source for several KAN variants. The implementation collection is organized around the original KAN model and the KAN 2.0 scientific-discovery framework [14, 35]. Our experiment wrapper currently exposes EfficientKAN for B-spline KANs, FastKAN for Gaussian RBF edge functions, and BSRBF-KAN for hybrid B-spline/RBF edge functions [31, 36]. The All-KAN repository also includes recent variants such as FC-KAN, PRKAN, and AF-KAN, which provide implementation references for future ablations [37, 38, 39]. All implementation choices are fixed across sample budgets within an experiment, and each reported result specifies the backend used to instantiate the KAN model.

A.1.5 Metrics

Let $\{(x_i, y_i)\}_{i=1}^{N_{\text{test}}}$ be the test set for one equation and let $\hat{y}_i$ be the model prediction. We define the test-set mean and standard deviation as

$$\bar{y} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} y_i, \quad s_y = \sqrt{\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (y_i - \bar{y})^2}. \quad (2)$$

The root mean squared error is

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (\hat{y}_i - y_i)^2}. \quad (3)$$

The normalized root mean squared error is

$$\text{NRMSE} = \frac{\text{RMSE}}{s_y + \epsilon}, \quad (4)$$

where $\epsilon = 10^{-12}$ prevents numerical instability for nearly constant targets. The mean absolute error is

$$\text{MAE} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} |\hat{y}_i - y_i|. \quad (5)$$

The relative ℓ_2 error is

$$\text{Rel-}\ell_2 = \frac{\sqrt{\sum_{i=1}^{N_{\text{test}}} (\hat{y}_i - y_i)^2}}{\sqrt{\sum_{i=1}^{N_{\text{test}}} y_i^2 + \epsilon}}. \quad (6)$$

The coefficient of determination is

$$R^2 = 1 - \frac{\sum_{i=1}^{N_{\text{test}}} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{N_{\text{test}}} (y_i - \bar{y})^2 + \epsilon}. \quad (7)$$

RMSE and MAE preserve the physical scale of each equation. NRMSE and relative ℓ_2 error allow comparisons across equations with different output magnitudes. R^2 reports explained variance and is easier to interpret when communicating whether a model has learned the dominant structure of the function.

A.1.6 Aggregate Reporting

For each equation and training size, we report the mean and standard error over seeds. Across equations, the plotted summaries average errors within each seed and report mean and standard deviation across seeds, since a small number of high-scale or ill-conditioned equations can dominate aggregate errors. We report these aggregate statistics for all equations, for univariate equations only, and for multivariate equations only. The full sample-efficiency curve plots NRMSE against N_{train} on a logarithmic x-axis. As scalar summaries, we report relative improvement and sample reduction against the best MLP curve. The area under the error-sample curve is computed by trapezoidal integration of NRMSE over $\log N_{\text{train}}$, where lower values indicate better sample efficiency. The threshold N_τ is the smallest training-set size for which the median NRMSE falls below a chosen target τ . If no method reaches τ , we report $N_\tau > 1000$.

A.2 RL Gymnasium Benchmark Details

A.2.1 Benchmark and Agents

We use the Gymnasium Classic Control environments CartPole-v1, Acrobot-v1, MountainCar-v0, MountainCarContinuous-v0, and Pendulum-v1. The observations are low-dimensional state vectors, which matches the intended use of KAN edge functions for smooth physical relationships. The runner supports both discrete and continuous action spaces, using categorical policies for discrete actions and diagonal Gaussian policies with tanh squashing for continuous actions. For continuous actions, the trainable log standard deviation is initialized to 0.

A.2.2 Model Sweep

The main comparison uses full actor-critic agents, so the actor and critic are both KANs for KAN runs and both MLPs for MLP runs. The KAN sweep includes B-spline KANs and Gaussian RBF KANs with hidden dimensions [2], [4], [8], [4, 4], [8, 8], and [16, 16]. The MLP sweep includes ReLU and sigmoid MLPs with hidden dimensions [16], [32], [64], [32, 32], and [64, 64]. The KAN settings use grid size 5, spline order 3, grid range $[-1.5, 1.5]$, and SiLU as the base activation. The runner records actor, critic, and total trainable parameter counts for every checkpoint.

A.2.3 PPO Training and Evaluation

Each training job collects complete episodes, computes generalized advantage estimates with $\gamma = 0.99$ and $\lambda = 0.95$, and applies clipped PPO updates with clip parameter 0.20. The optimizer is Adam with learning rate 3×10^{-4} , no weight decay, entropy coefficient 0.01, value coefficient 0.50, four PPO epochs per update, and gradient clipping at 0.50. Updates are performed after batches of five episodes in the full protocol. Training seeds are paired across model families, and evaluation uses deterministic actions with five evaluation episodes per checkpoint. Discrete policies are evaluated with the action of maximum logit, while continuous policies are evaluated with the mean action. The result file used for the paper contains ten paired random seeds and learning curves up to 6000 training episodes. During training, episodes, cumulative environment steps, recent training returns, deterministic evaluation returns, episode lengths, PPO losses, runtime, and parameter counts are logged.

A.3 RL Gymnasium Noise Benchmark Details

A.3.1 Reward Perturbation

The noisy benchmark keeps the same Gymnasium environments, PPO update, model sweep, and deterministic clean evaluation protocol as the clean Gymnasium benchmark. During training only, each clean reward r_t is replaced by $r_t^{\text{noise}} = r_t + \alpha \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, 1)$. The main sweep uses $\alpha \in \{0, 0.25, 0.5, 1, 2, 4, 7, 10\}$ in raw reward units. Because noise only enters the rewards used for advantage estimation, the experiment tests robustness to corrupted training feedback rather than robustness to stochastic evaluation.

B Detailed Results

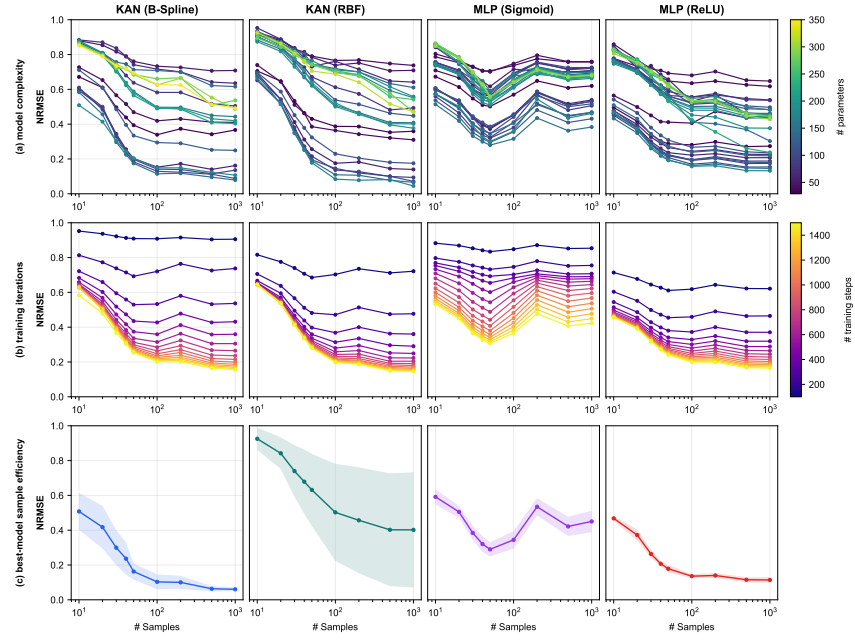


Figure 3: Experiment 1: Feynman Benchmark Experiment Design.

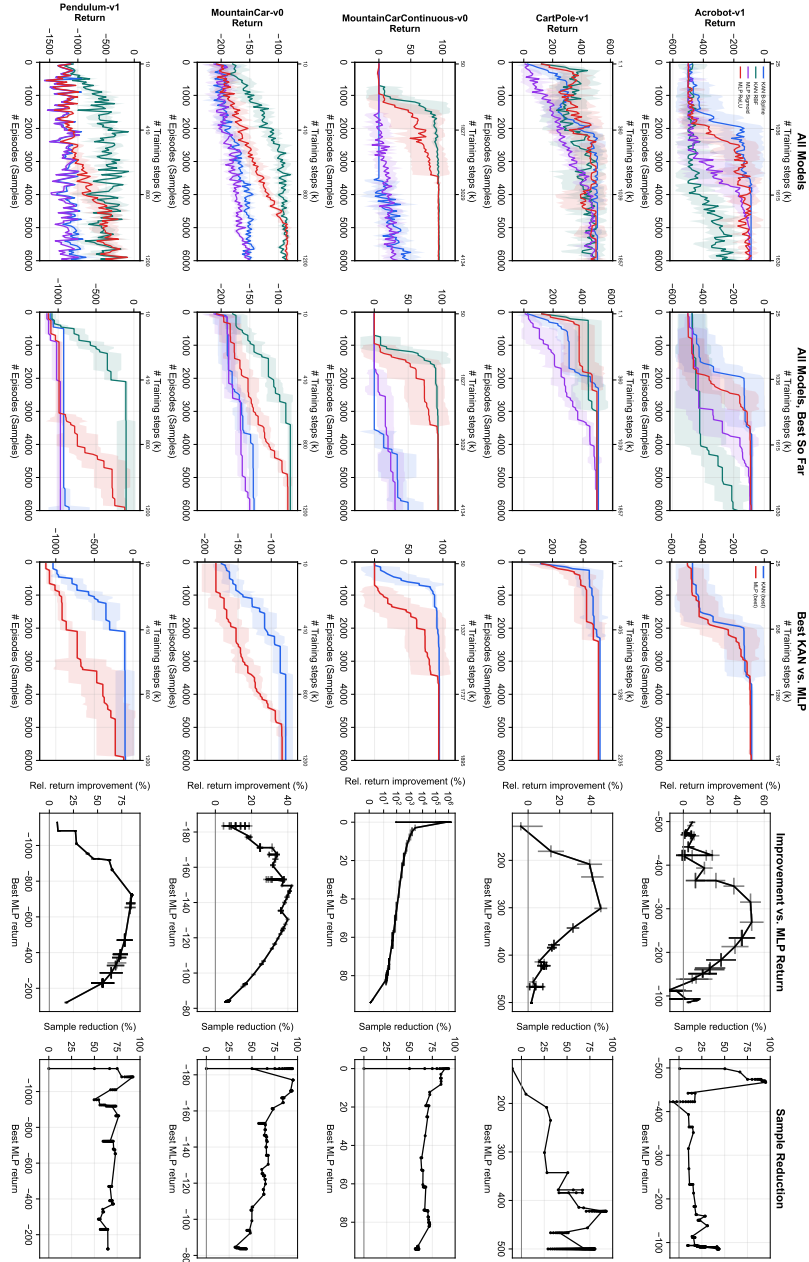


Figure 4: Experiment 2: DeepRL Gymnasium Experiment Design.

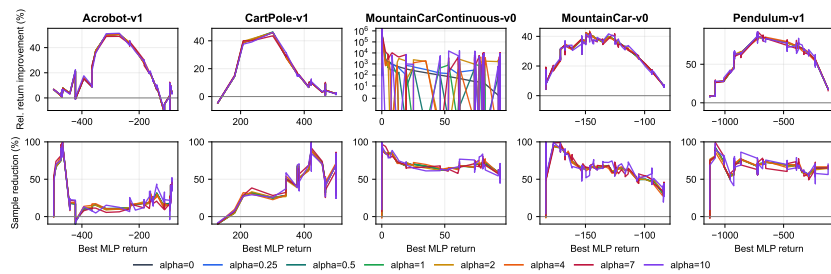


Figure 5: Experiment 3: Sample Efficiency in Noised Gymnasium Benchmark.