universitätfreiburg

SCALABILITY AND ROBUSTNESS OF KANs APPLIED TO REINFORCEMENT LEARNING

Members: Beatriz Fernández Larrubia, Ayisha Ryhana Dawood, Pojen Shih *Supervisor: Baohe Zhang*

1. Introduction

Kolmogorov-Arnold Networks (KANs) utilize the Kolmogorov-Arnold representation theorem unlike MLPs, which base their structure on the universal approximation theorem. KANs are a combination of splines (optimal for low-dimensional functions) and MLPs (optimal for feature learning).

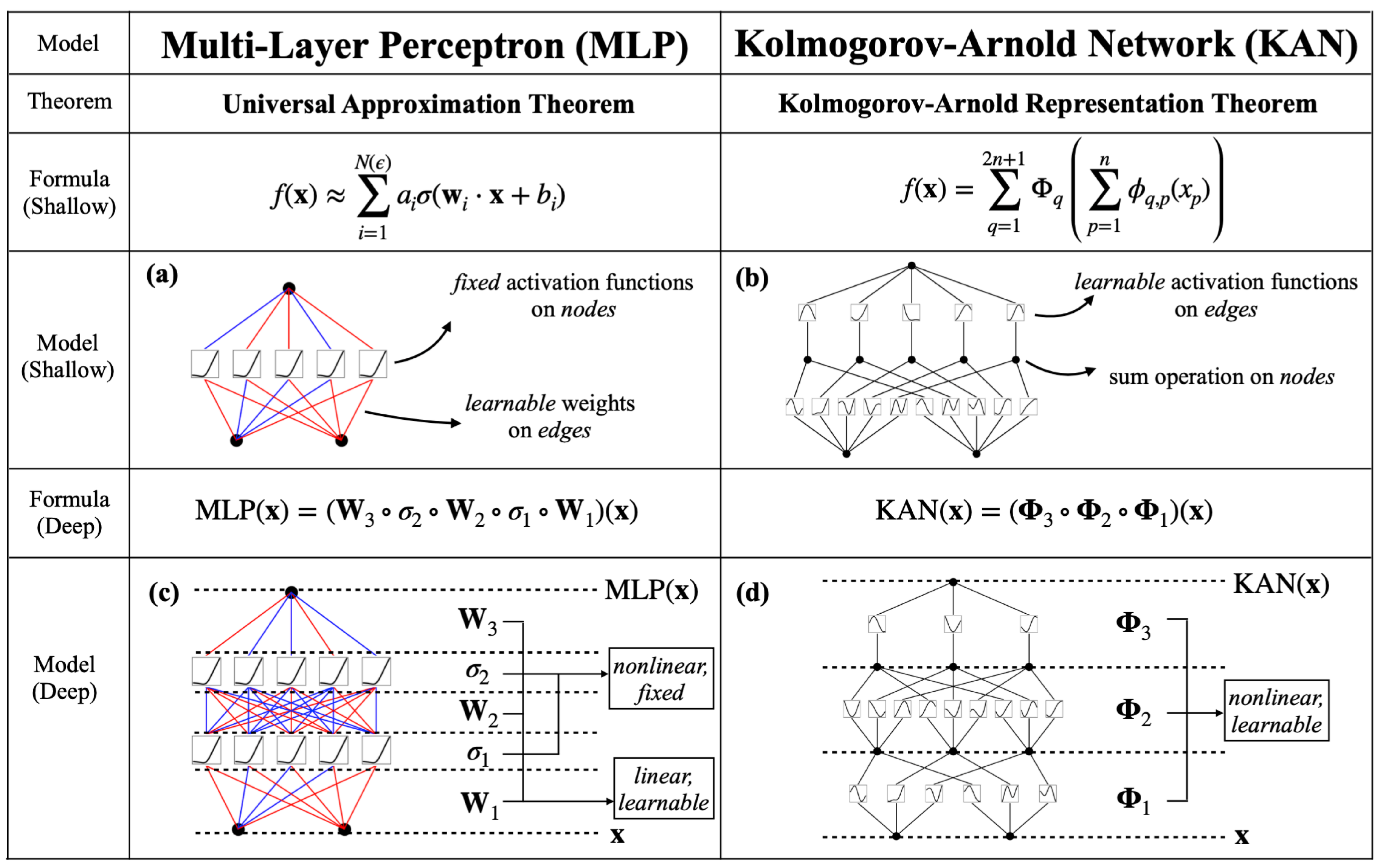

Fig1. Multi-Layer Perceptrons (MLPs) vs. Kolmogorov-Arnold Networks (KANs) [1]

2. Methods

KANs [2] are integrated into the reinforcement learning framework by replacing the MLP component of Deep Q-Networks (DQN), resulting in the Kolmogorov-Arnold Q-Network (**KAQN**) [3].

The three points of interest were:

1. Comparing the performance of KANs and MLPs in environments with **varying numbers of distractors**.
2. Evaluating how KANs compare against MLPs in environments with **high complexity**.
3. Investigating how KANs fare compared to MLPs in **non-physics-driven environments**, given that KANs proved effective in physics-driven problems as shown in the original paper [1].

The code of the KANrl repository [3] was modified to implement continuous-space environments and **DDPG** [4].

To mitigate the lengthy processing durations of PyKAN [2], **EfficientKAN** [5] was adopted to ensure more time-efficient execution.

3. Experiments

Varying distractors

The Ant-v4 environment [6] was used to experiment with a varying number of distractors. The distractors are noise inputs that do not contribute to the Ant's current state.

The KAQN has *(x+n, 256, 1)* architecture where *x* is the shape of the original observation space and *n* distractors are sampled from Gaussian noise.

High Complexity

An arbitrary number of legs are added to the Ant using the MorphingAnt [8] environment to increase complexity. The legs are distributed symmetrically around the torso. The shape of the observation space scales with the number of legs *n* as: *observations = 11 + (6 * n)*

4. Results

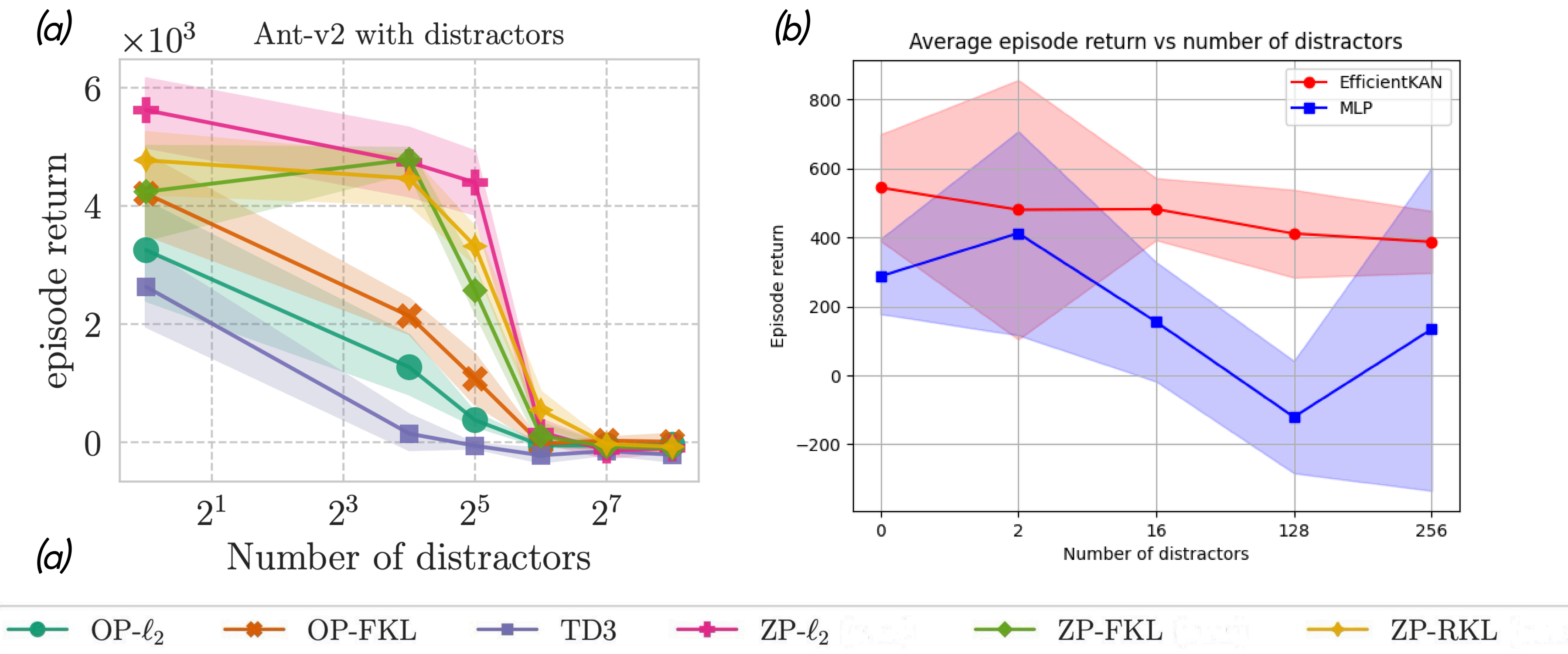

Fig2. Experiments with distractors. (a) Images from Ni, Tianwei et al. [7], (b) Own experiments with KAN and MLP, averaged over 3 seeds (0, 1, 2).
KANs appear to be superior to the other baselines and MLP.

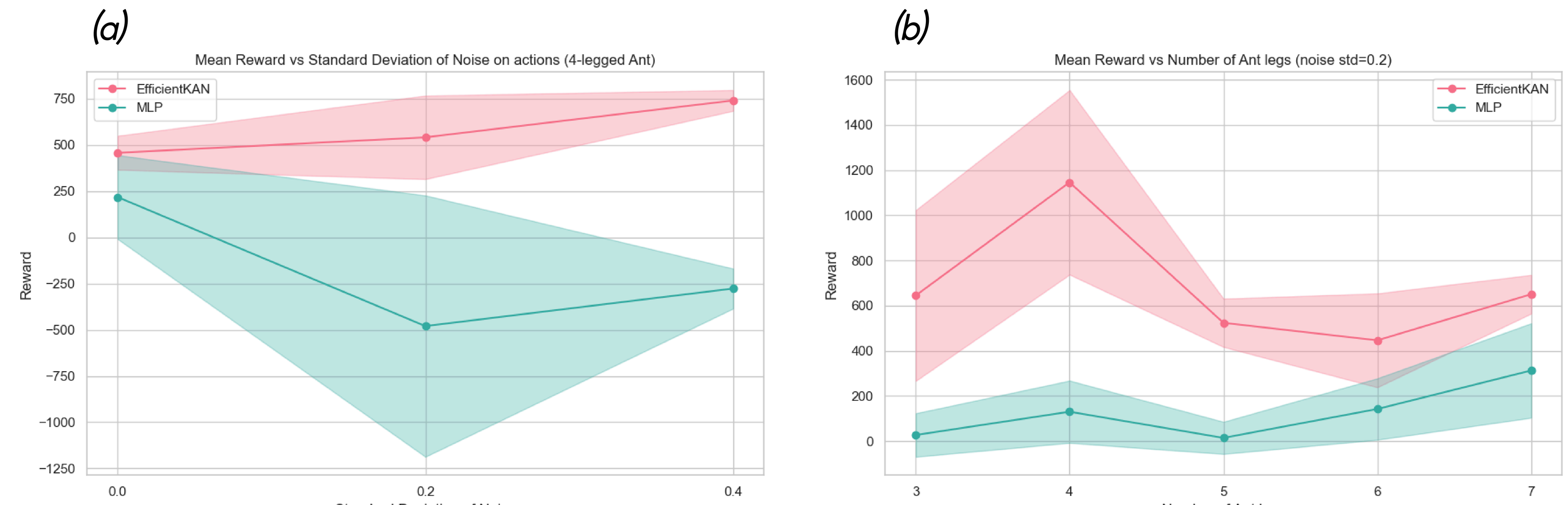

Fig 3. MorphingAnt experiments [8] tested with a fixed seed (0).
(a) Impact of noise applied to actions on 4-legged Ant training; EfficientKAN improves with noise.
(b) Training with different leg counts and fixed noise (std=0.2), using healthy_reward=0.1 to promote movement. All tests use healthy_reward=1.0 to ensure comparability.

(a) (b) (c)

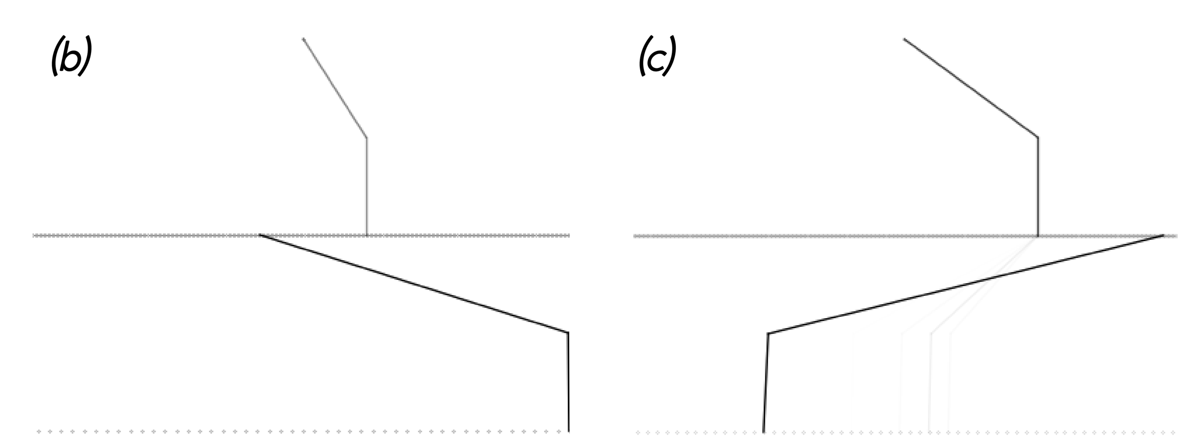

Fig 4. MorphingAnt [8] trained with noise std=0.2 and healthy_reward=0.1
(a) Video of episode renders of 4-legged Ant (b) Interpretable graph of the 4-legged Ant KAQN with (47, 256, 1) architecture. (c) Interpretable graph of the 5-legged Ant KAQN with (56, 512, 1) architecture. These graphs reveal which inputs are most useful for the calculations, making it easier to understand the results.

5. Conclusion

- **Performance with distractors:** KAQNs exhibit more stable performance over traditional MLP-based Q-networks with increasing distractors in the environment.
- **Performance in complex environments:** KAQNs consistently outperform MLPs in increasingly complex environments, with both networks using identical configurations in each case.
- **Robustness:** KANs perform well in continuous environments, making them versatile for various reinforcement learning applications.

Limitations:

- **Computational Complexity:** Higher complexity leads to greater computational demands and longer training times. Further research is needed for large, dynamic environments.
- **Adaptation Difficulties:** The complex base code complicates adaptation to other environments, which prevented the completion of experiments in non-physics environments.

References

[1] Liu, Wang, et al. "Kan: Kolmogorov-arnold networks." arXiv:2404.19756 (2024). [2] KAN Github: https://github.com/KindXiaoming/pykan [3] KANrl github: https://github.com/riiswa/kanrl [4] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015). [5] EfficientKAN github: https://github.com/Blealtan/efficient-kan [6] Ant-v4: https://gymnasium.farama.org/environments/mujoco/ant/ [7] Ni, Tianwei, et al. "Bridging State and History Representations: Understanding Self-Predictive RL." arXiv preprint arXiv:2401.08898 (2024). [8] MorphingAnt github: https://github.com/brandontrabucco/morphing-agents/blob/master/morphing_agents/mujoco/ant/env.py [9] Our github link: https://github.com/BeatrizFernandezLarrubia/kanrl/tree/main