Exploring Sparse Reward Environments with Weight Agnostic Neural Networks

Tobias Lingenberg and Markus Reuter
Department of Computer Science, Technical University of Darmstadt, Germany

***Abstract*—Sparse reward environments present a significant challenge in reinforcement learning (RL), as agents receive little to no feedback for extended periods, making effective learning difficult. Traditional RL algorithms struggle in these settings without human-engineered feedback to guide training. In this work, we explore a novel direction using Weight Agnostic Neural Networks (WANNs), which leverage evolutionary search to discover proper network architectures that, even without trained weights, are capable of performing such tasks. We evaluate WANNs on modified versions of the MountainCar and LunarLander environments, where rewards are only given upon successful task completion. Our results demonstrate that WANNs can successfully learn compact, interpretable policies in these sparse settings, whereas conventional RL methods fail without auxiliary rewards.**

I. Introduction

In reinforcement learning (RL), agents learn to maximize rewards by interacting with an environment. However, in certain tasks, reward feedback is only available after a specific objective is achieved (e.g., winning a game or reaching a goal), leaving the agent without guidance for most of its training. This is known as a sparse reward environment, and it poses significant challenges for RL algorithms due to:
(1) Delayed Feedback: Agents may receive little to no feedback for long periods. Once a reward is finally collected, it becomes difficult to assign credit to the specific actions that led to it [1].
(2) Exploration Difficulty: Traditional networks often struggle to find the rare trajectories that yield rewards, as they rely heavily on gradient information to guide learning.

As a result, standard RL approaches often require extensive manual engineering to achieve acceptable performance. Common techniques include reward shaping [2], [3], where additional intermediate rewards are manually designed to guide the agent, and staged training [4], where objectives are decomposed into easier subtasks. In fact, the motivation for this project stems exactly from our experience facing these challenges. We encountered hard-to-train algorithms, the need for domain knowledge, and significant trial-and-error in designing custom reward structures or task-specific training processes.

To address these challenges, our project aims to explore an alternative approach inspired by evolutionary algorithms (EAs) and their ability to navigate optimization landscapes that are mostly flat and sparse. EAs have demonstrated promising behavior for such "needle in the haystack" problems in applications such as software validation to find bugs, where direct gradient-based optimization is impractical.

Specifically, we focus on Weight Agnostic Neural Networks (WANNs), introduced by Gaier and Ha [5], which leverage architecture search based on EAs to discover simple, interpretable neural networks that can perform various tasks without relying on tuned weights. We hypothesize that WANNs, with their reliance on evolutionary principles, can serve as a robust alternative to standard RL approaches in challenging sparse reward environments. To evaluate this novel approach, we applied WANNs to modified versions of two common RL benchmark tasks: MountainCar and LunarLander, where agents receive a reward only upon completing the task.

The key contributions of this work can be summarized as follows: *(1)* We comprehensively evaluate WANNs in sparse reward environments and compare their performance to widely-used RL algorithms, including Q-Learning [6], DQN [7] and PPO [8]. *(2)* We provide a detailed examination of WANNs, highlighting their tendency to favor simpler architectures that enhance interpretability. By visualizing learned policies and the Pareto fronts, we illustrate the behavior of WANN solutions and how they balance performance and complexity.

II. Background

In biology, precocial species show behaviors and abilities at birth that allow them to survive immediately without extensive learning. This concept inspired Gaier and Ha [5] to develop Weight Agnostic Neural Networks (WANNs), which are neural networks that, thanks to their topology, can perform tasks effectively even when their weights are randomly initialized. Instead of optimizing weights, WANNs focus on optimizing network architectures that are robust.

WANNs are based on principles of Neural Architecture Search (NAS) methods, in particular the NEAT algorithm [9], which evolves network topologies and their weights simultaneously. While NEAT requires weight optimization, WANNs avoid this step entirely by evaluating network performance with randomly assigned weights over multiple rollouts. This ensures that performance is a product of the network topology alone.

The evolutionary process of Gaier and Ha [5] follows these steps: *(1)* An initial population of minimal neural network topologies is created, with partial connections between input and output nodes. *(2)* Each network is evaluated over multiple rollouts in the environment, with a different shared weight value assigned at each rollout. *(3)* Networks are ranked according to their performance and complexity resulting in

a multi-objective optimization problem handled by Pareto dominance. *(4)* The top-ranked networks undergo variation through operations similar to NEAT [9], namely adding nodes, creating new connections, or changing activation functions.

This process mirrors principles observed in biology, where evolution shapes systems to efficiently perform tasks critical to survival. The reliance on architecture, rather than detailed parameter tuning, aligns with the critique offered by Zador on how learning is currently implemented in most artificial neural networks [10]:
"Animal brains are not the blank slates, equipped with a general purpose learning algorithm ready to learn anything, as envisioned by some AI researchers; there is strong selection pressure for animals to restrict their learning to just what is needed for their survival."

III. Methodology

We evaluated WANNs in sparse reward environments using two well-established tasks: *Mountain Car* and *Lunar Lander* [11]. These environments were chosen because they highlight key RL challenges, such as delayed feedback and exploration inefficiency, making them ideal for testing alternative learning approaches.

a) Sparse Mountain Car: This task involves an underpowered car building momentum to ascend a hill and reach a target. Sparse rewards are awarded only upon reaching the goal. The agent's state includes position and velocity in x direction, and the action space can be either discrete (accelerate left, no acceleration, accelerate right) or continuous in $[-1, 1]$, where negative values accelerate left and positive values accelerate right.

b) Sparse Lunar Lander: This more complex task requires a spacecraft to safely land at a designated point with sparse rewards granted solely for successful landings. The 8-dimensional state includes position, velocity, angle, angle velocity and ground contact of the two legs, while the agent can use 4 discrete thrust controls (do nothing, fire main, left or right engine). This task poses unique challenges due to its higher-dimensional state space, reliance on precise control, and increased likelihood of failure, making it a strong candidate to test the generalization and robustness of WANNs under sparse feedback conditions.

To amplify the sparsity of rewards, we redefine the reward r at each step as:

$$r_b(t) = \begin{cases} 100 \cdot b^t & \text{if goal is reached,} \\ 0 & \text{otherwise.} \end{cases}$$

Here t, is the time steps taken to reach the goal, and b is a decay factor ($b = 0.99$ for Mountain Car, $b = 0.999$ for Lunar Lander). This formulation ensures that faster solutions receive higher rewards, encouraging efficient exploration.

Performance was measured by cumulative reward, indicating both task success and the efficiency of reaching the goal. If an algorithm failed to solve Sparse Mountain Car, we introduced reward shaping, adding an auxiliary reward for the models current velocity $r_+ = |\dot{x}|$ at each step to encourage momentum, as an essential factor in reaching the goal.

To ensure a meaningful comparison, we benchmarked WANN against three standard RL algorithms that represent distinct learning paradigms: Q-Learning [6] (a tabular, non-deep learning method), Deep Q-Networks (DQN) [7] (an off-policy algorithm), and Proximal Policy Optimization (PPO) [8] (an on-policy approach). We used the implementation from Stable-Baselines3 [12] and conducted a small hyperparameter tuning for each algorithm. This selection ensures a diverse range of benchmarks to assess WANNs' relative performance across both simple and complex RL methodologies.

IV. Experimental Results

The performance of the Weight Agnostic Neural Networks (WANN) algorithm in comparison to standard reinforcement learning (RL) approaches is summarized in Table I. Notably, none of the conventional RL algorithms successfully identified solutions for the sparse reward versions of both tasks. This limitation is likely attributed to the highly sparse reward landscape, where feedback is provided only upon successful task completion. This interpretation is further supported by experiments in which auxiliary rewards were introduced. Under these conditions, nearly all RL algorithms were able to complete the task, highlighting their dependence on frequent reward signals to guide learning. In contrast, the WANN algorithm demonstrated robust performance across all evaluated tasks.

TABLE I: Comparison of WANNs and standard RL algorithms on Sparse Mountain Car (SMC) with discrete and continuous action spaces, and Sparse Lunar Lander with a discrete action space. The values represent the average number of time steps required to reach the goal, based on the best-performing network or model over 3 to 10 trials. Values in parentheses indicate performance when intermediate reward feedback (reward shaping) was applied. ∞ denotes cases where the goal was never reached, while ✗ indicates that the method is not applicable.

Method	SMC discrete	SMC continous	Lunar Lander
WANN	**123.92**	**136.73**	**1135.37**
Q-Learn.	∞ (110.53)	∞ (∞)	∞
PPO	∞ (133.27)	∞ (224.25)	∞
DQN	∞ (322.79)	✗	∞

To gain deeper insights beyond final fitness values, we provide visual representations of the selected model, its policy, and the Pareto front of its population (see Fig. 1). These visualizations reveal key characteristics of the discovered solutions.

For the Sparse Mountain Car task with a discrete action space, the resulting network exhibits a remarkably simple structure, yet effectively captures the underlying task dynamics. The necessary strategy for reaching the goal involves leveraging momentum by swinging back and forth to generate sufficient velocity to ascend the hill. This behavior is inherently encoded within the network's architecture. Specifically, the network utilizes a constant bias of 1, which gets inverted to -1. This ensures that the "no acceleration" action is never selected, as the corresponding output node consistently receives a negative value. When the car's velocity is positive,

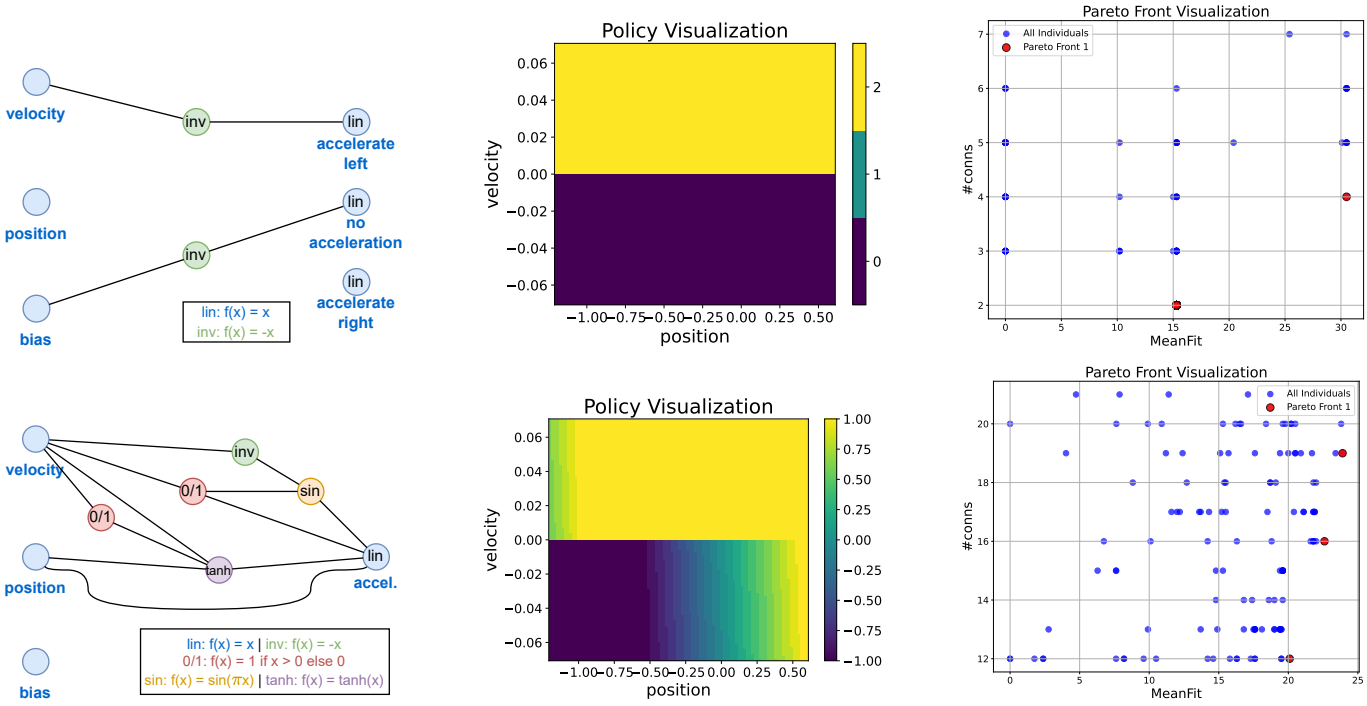

Fig. 1: Visualization of solutions for the Sparse Mountain Car task with discrete action space (top row) and continuous action space (bottom row). The left column displays a well-performing, minimal network architecture evolved by the algorithm. The middle column illustrates policy visualizations across all states $(x, \dot{x})$, where positive values represent actions accelerating right and negative values indicate left acceleration. The right column shows the evolved population from the evolutionary algorithm, plotted against the two objectives: mean fitness (maximize) and number of connections (minimize). The Pareto front is highlighted in red, demonstrating the trade-off between performance and network complexity

indicating rightward movement, the network inverts this value, leading to a negative activation for the "accelerate left" output node. Consequently, the argmax function selects "accelerate right." Conversely, when the velocity is negative (leftward movement), the same mechanism dictates that "accelerate left" is chosen as the optimal action. This results in a very simple yet effective behavior strategy to solve the task.

Furthermore, the architecture exhibits weight-agnostic properties when constrained to single-value shared weights. Each path within the network contains two connections, effectively neutralizing the impact of the absolute weight sign. This characteristic ensures robustness across varying weight values.

The policy visualization in the top row of Fig. 1 confirms the described behavioral patterns. Additionally, an analysis of the population distribution across the dual objectives - mean fitness and network complexity (measured by the number of connections) - reveals a clear trend: performance degrades as connectivity decreases and does not significantly improve with excessive complexity. Based on this observation, the network depicted in the top left of Fig. 1 represents a well-balanced trade-off between simplicity and task performance.

The second row of Fig. 1 shows the results for Sparse Mountain Car with continuous action space. Compared to the discrete action space variant, the resulting network is slightly more complex, but still a sparsely connected neural network. The policy visualization indicates that the network aims to replicate the same fundamental behavior observed in the discrete version. However, due to the nature of continuous control and the constraints of a weight-agnostic architecture with a single output node, additional structural complexity is required to encode this behavior effectively.

V. Discussion of Results and Project

A. WANNs in Sparse Reward Environments

Why does WANNs work so much better on sparse reward settings than classical RL methods? We were surprised to find such a clear result: WANNs successfully solved all tasks tested, while conventional RL approaches struggled.

We attribute this success to the "simple first" search process of WANNs. Simple solutions are prioritized by design, which has two advantages: (1) the combinatorial search space of simple solutions is smaller and more manageable, and (2) simple solutions are typically more generalizable models and thus a good initial direction to continue the search. In contrast, standard RL methods explore all high-complexity solutions candidates from the start, leading to potential inefficiencies in sparse reward settings.

Another key advantage of WANNs is their ability to maintain diverse candidate solutions throughout the search process. Unlike traditional RL, which optimizes a single agent, evolutionary algorithms explore multiple strategies in parallel,

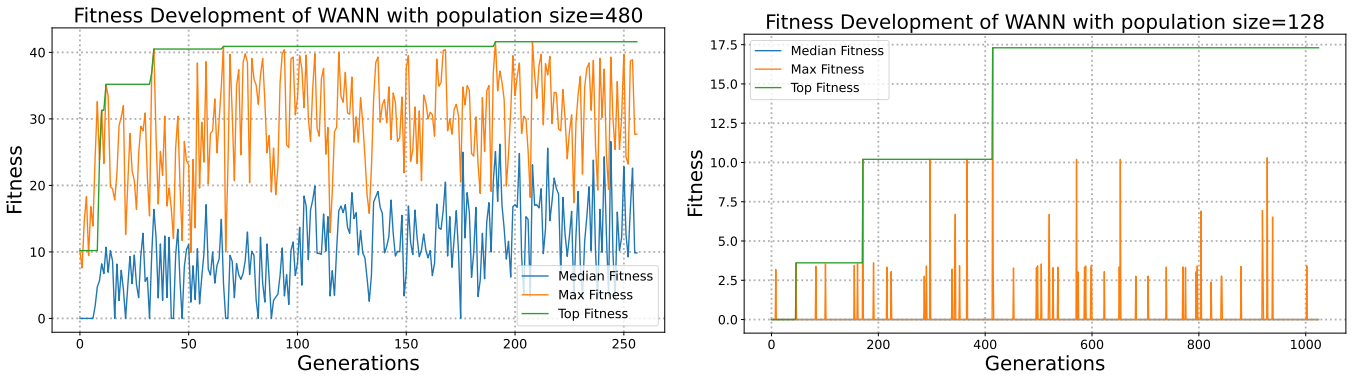

Fig. 2: Fitness development of WANNs for the Lunar Lander task with different population sizes.

increasing the likelihood of discovering rare but effective policies.

WANNs' preference for simple architectures results in interpretable models with clear decision boundaries, as seen in its policy visualizations (Fig. 1). This is consistent with a fundamental principle of machine learning: prefer simpler models because they often generalize better.

B. Limitations and Potential Improvements

While WANNs performed well, certain limitations became evident. In the continuous Sparse Mountain Car task, we observed that later generations evolved increasingly complex models. After attempting to develop our own simple policy, we hypothesized that the expressiveness of the given operators could be improved. The current activation functions may not be sufficient to effectively approximate such a simple policy as seen in the discrete case. A complex combination of various nested activation functions is required. One potential improvement is to allow the algorithm to either evolve weights (as in NEAT) or incorporate additional activation functions (e.g., $x + 1$, $2x$) that mimic weight and bias adjustments. We hypothesize that this would speed up convergence and lead to even simpler architectures.

Regarding the Pareto fronts (Fig. 1), we also observed that increased complexity does not necessarily improve fitness. This trend becomes more pronounced over longer evolutionary runs. Also, a lack of diversity emerged, with individuals clustering in solution space. This issue stems either from a WANN hyperparameter that ignores the complexity objective in 20% of cases, leading to the loss of simple architectures, or from a lack of diversity preserving mechanisms such as speciation [9].

It is worth noting that the discovered WANN architectures can be further fine-tuned by training their weights, potentially greatly improving performance, as shown by Gaier and Ha [5]. This is a simple but interesting to go for future work.

VI. Conclusion

In this project, we explored the potential of Weight Agnostic Neural Networks (WANNs) for reinforcement learning (RL) tasks with sparse rewards. Our findings demonstrate that WANNs can successfully solve the sparse MountainCar and LunarLander task. In contrast, conventional RL approaches like Q-Learning, PPO, and DQN completely fail in these environments, requiring auxiliary rewards to reach the goal—a practice that can lead to unintended biases.

By analyzing the discovered network architectures, policies, and evolved populations, we find that WANNs produce compact and interpretable solutions. Our results suggest that WANNs are indeed a promising alternative to traditional RL models in sparse settings. Their evolutionary-based optimization enables effective exploration and convergence in problem landscapes that lack gradient information.

References

[1] E. Pignatelli, J. Ferret, M. Geist, T. Mesnard, H. van Hasselt, O. Pietquin, and L. Toni, "A survey of temporal credit assignment in deep reinforcement learning," *arXiv preprint arXiv:2312.01072*, 2023.

[2] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287.

[3] E. Wiewiora, *Reward Shaping*. Boston, MA: Springer US, 2010, pp. 863–865. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_731

[4] S. Shen, P. Walsh, K. Keutzer, J. Dodge, M. Peters, and I. Beltagy, "Staged training for transformer language models," in *International Conference on Machine Learning*. PMLR, 2022, pp. 19 893–19 908.

[5] A. Gaier and D. Ha, "Weight agnostic neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[6] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[9] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[10] A. M. Zador, "A critique of pure learning and what artificial neural networks can learn from animal brains," *Nature communications*, vol. 10, no. 1, p. 3770, 2019.

[11] G. Brockman, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[12] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, and R. Traoré, "Stable-baselines3: Reliable reinforcement learning implementations," 2021. [Online]. Available: https://github.com/DLR-RM/stable-baselines3