Learning How Deep to Go: Self-Scaling Deep Reinforcement Learning

Michelangelo Vegliò
University of Chieti-Pescara

Marco Fantozzi
University of Parma

Antonio Di Cecco
University of Chieti-Pescara

Carlo Metta
ISTI, CNR Pisa

Flora Angileri
Tor Vergata University of Rome

Simone Treccani
University of Parma

Adrienne C. R. Macazar
University of Parma

Silvia G. Galfrè
University of Pisa

Maurizio Parton
University of Chieti-Pescara

Francesco Morandin
University of Pisa

Abstract

Deep Reinforcement Learning (DRL) has achieved remarkable results in complex sequential decision-making tasks, often using very deep neural networks. However, these architectures incur substantial computational and energy costs, and selecting the optimal network depth in advance remains an open challenge. In this paper, we introduce SCALE-RL, a self-scaling DRL framework that dynamically adjusts its architectural depth during training, allowing the network to automatically adapt its depth to the task. Integrated into an AlphaZero-style pipeline for Othello, our approach matches the playing strength of the baseline agent while reducing network depth by 50%. This process not only translates into substantial savings in computation and energy but also enhances model interpretability through the additive decomposition of decision-making across layers. Our results suggest that enabling DRL models to discover the complexity they require, rather than relying on fixed, over-parameterized architectures, makes it possible to develop more efficient, interpretable, and sustainable DRL agents.

INTRODUCTION

The success of Deep Reinforcement Learning (DRL) relies heavily on the representational power of deep neural networks. This has been true from foundational breakthroughs in Atari games and Go (Schrittwieser et al., 2020; Silver et al., 2016) to the modern alignment of large language models (Christiano et al., 2017; Rafailov et al., 2023). Across these domains, increasing network scale has consistently been a key driver of performance improvements. However, this trend toward deeper architectures comes with significant computational costs and raises questions about efficiency. Traditional approaches rely on fixed depths that must handle the most complex cases, resulting in excessive capacity for simpler sub-tasks or early training phases. Moreover, selecting this optimal depth typically requires costly hyperparameter or architecture searches.

In this paper, we propose a fundamentally different approach: **SCALE-RL (Self-Scaling Architecture for Learning Efficiency in Reinforcement Learning)**, where the network architecture dynamically adapts its own depth during training based on task requirements. Our framework builds upon globally connected neural networks (Di Cecco et al., 2024, GloNets), which offer the dual benefits of principled self-regulation and inherent interpretability due to their architectural design. Coupled with a dynamic depth control algorithm, our framework enables dynamic architectural scaling: the policy and value networks can grow by adding layers to handle rising complexity and prune unneeded ones to increase efficiency.

We validate SCALE-RL on a nontrivial testbed: an AlphaZero-style agent for the game of Othello. Although not as computationally intensive as Go or Chess, Othello remains a combinatorially complex domain with a rich body of work on learning agents. Moreover, AlphaZero is a widely adopted and well-established DRL algorithm. We selected this setting as an accessible benchmark: Othello strikes a balance between tractable complexity and nontrivial strategic depth.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

Our results show three main contributions. First, SCALE-RL is **efficient**, automatically reducing network depth by about 50% compared to a fixed baseline. Second, it is **effective**, maintaining comparable playing strength despite this reduction. Third, it is **interpretable**, providing stable gradient-based explanations revealing a hierarchy of learned features from local tactics to global strategy.

We emphasize that Othello/AlphaZero is used here only as an empirical testbed; our goal is not to build a superhuman Othello agent, but rather to present a general framework enhancing the efficiency and interpretability of any reinforcement learning pipeline with deep neural networks for policy-value approximation.

RELATED WORK AND OUR APPROACH

To address the significant computational and data demands of modern DRL, the literature has explored numerous strategies for improving architectural efficiency. These efforts can be broadly categorized into three main paradigms: static compression/scaling, automated architecture search, and dynamic adaptation.

Static Compression and Scaling This paradigm focuses on creating fixed, efficient architectures prior to deployment. Methods like network pruning and knowledge distillation reduce model size, while scaling approaches such as Elastic Networks and EfficientNet (Tan and Le, 2019) systematically scale network width, depth, and resolution. While effective at producing compact models, these methods yield static architectures based on predetermined coefficients. This inherent rigidity prevents them from adapting online to the changing task complexity during DRL training.

Automated Architecture Search (NAS) NAS methods automate the discovery of optimal network topologies. Early approaches (Zoph and Le, 2017) were computationally prohibitive. More recent variants like ENAS (Pham et al., 2018) and DARTS (Liu et al., 2019) reduced search costs significantly by using weight-sharing supernets. However, the core limitation of NAS remains: it is an offline search process that produces a single, fixed architecture. Consequently, it cannot dynamically adjust the model's structure during online DRL training in response to evolving task demands.

Dynamic Adaptation This class of methods adjusts a model's computational graph during inference. Progressive Networks (Rusu et al., 2016), for example, grow monotonically by adding new network columns for each task, preventing catastrophic forgetting but leading to unbounded parameter growth and requiring task labels. Early Exit Networks (Teerapittayanon et al., 2016) allow inputs to exit at intermediate classifiers, reducing average inference cost. However, their base architecture remains fixed, limiting their adaptability to the model structure itself.

In summary, while existing approaches provide either static efficiency or limited forms of dynamism (e.g., monotonic growth or variable inference paths), a framework that enables *flexible, bidirectional structural adaptation during DRL training remains an open challenge*.

Our Approach. To address this gap, we propose SCALE-RL, a novel framework for dynamic adaptation in DRL that allows a model to both grow and shrink its depth online. Our method is inspired by GloNet layers (Di Cecco et al., 2024), which operate by aggregating feature maps from multiple depths to form the final output. This mechanism, whose implementation is detailed in the Methods section, allows our framework to *adjust its architecture during training in response to task complexity* with minimal overhead. Our approach offers several key advantages over the methods described above.

Computational Efficiency: Unlike Progressive Networks, which grow unbounded as they accumulate parameters, SCALE-RL dynamically expands or prunes network layers in response to task demands. In contrast to NAS, our framework adapts online during DRL training.

Dynamic Bidirectional Adaptation: In contrast to the fixed architectures from NAS or the unbounded growth of Progressive Networks, our framework enables bidirectional depth adjustment—expanding when task complexity increases and pruning itself when greater efficiency is sufficient.

Task-Agnostic Operation: Unlike Progressive Networks, which require task labels to trigger structural changes, SCALE-RL operates seamlessly within a single DRL task, adapting to evolving strategic complexity without external supervision.

Explainability: By aggregating features from multiple depths and by eliminating the need for batch normalization (Di Cecco et al., 2024, Section 4), SCALE-RL naturally enhances DRL explainability, because intermediate representations can be inspected directly and their effect on the final decision can be interpreted.

METHODS: GENERAL FRAMEWORK

The SCALE-RL framework is based on the simple yet powerful concept of the GloNet layer. Below, we briefly describe its definition and theoretical foundations, while referring the reader to the original paper (Di Cecco et al., 2024) for a full technical treatment.

In a standard feedforward network, features are processed sequentially, and the final prediction head only receives the

output of the last block (where "block" denotes a modular unit such as a residual block or other composite architectural element), potentially losing valuable low-level information—a limitation addressed in the literature by various architectural innovations (He et al., 2016a,b; Huang et al., 2017; Metta et al., 2024, 2025a,b; Di Cecco et al., 2025b,a). A *GloNet layer* mitigates this by constructing a "global" feature vector that aggregates (via summation) the intermediate representations produced by selected blocks throughout the network. This vector, which captures information from multiple semantic levels of abstraction, is then passed to the prediction head. As a result, the model can exploit both low- and high-level features simultaneously, enabling more effective feature utilization, dynamic depth adaptation, and improved explainability.

As established in the Introduction, the goal of this paper is to develop a general and adaptive DRL framework that is both computationally efficient and interpretable. Our proposal consists of two core components:

1. **GloNet-Augmented Policy-Value Network:** The principle of modifying a standard policy-value network to incorporate the GloNet architecture.
2. **Dynamic Depth Controller:** An algorithm that monitors the augmented network and adapts its depth on the fly.

GloNet-Augmented Policy-Value Network Any policy and value network can be modified to include a GloNet layer immediately before the prediction heads. This approach is *architecture-agnostic*: in principle, a GloNet layer can be incorporated into any network architecture. However, this modification requires additional changes, such as removing batch normalization and skip connections when present, as detailed in (Di Cecco et al., 2024, Section 4).

Dynamic Depth Controller When adapting the policy and value networks, the guiding principle is not simply to prune layers, but to maintain their capacity in a state of equilibrium, ensuring sufficient redundancy to handle increases in task complexity without becoming inefficient. We achieve this via a *redundancy buffer*: a target number of blocks at the end of the network that can be removed without causing a significant performance drop. Our algorithm periodically assesses the current size of this buffer and performs one of two actions:

Pruning: If the current buffer is larger than our target, the network is unnecessarily deep, and we remove the excess blocks.

Growing: If the current buffer is smaller than our target, the network may lack the capacity for future challenges, so we add new, randomly initialized blocks.

Algorithm 1 formalizes the procedure for dynamic depth adjustment. First, the network's deepest blocks are iteratively shut down. After each shutdown, the `EvaluateNetwork` helper function assesses the performance of the now-shortened network on a validation dataset, $\mathcal{D}_{\text{val}}$. This process continues until the performance degradation exceeds a predefined threshold, τ. The number of blocks successfully removed defines the measured buffer size, which is then compared to a target, B_{target}, and the network's final depth is adjusted accordingly by either pruning or adding blocks.

Algorithm 1 Dynamic Depth Controller

Require: Current network with L blocks aggregated in the GloNet layer, validation dataset $\mathcal{D}_{\text{val}}$, performance threshold τ, target buffer size B_{target}
Ensure: Updated network with adjusted depth
{1. Determine the current size of the redundancy buffer}
$\text{loss}_{\text{base}} \leftarrow \text{EvaluateNetwork}(L, \mathcal{D}_{\text{val}})$
$\text{buffer}_{\text{current}} \leftarrow 0$
for $k = 1$ **to** $L - 1$ **do**
  $L_{\text{temp}} \leftarrow L - k$
  $\text{loss}_k \leftarrow \text{EvaluateNetwork}(L_{\text{temp}}, \mathcal{D}_{\text{val}})$
  if $\frac{(\text{loss}_k - \text{loss}_{\text{base}})}{\text{loss}_{\text{base}}}) > \tau$ **then**
    break {Performance drop exceeds threshold}
  end if
  $\text{buffer}_{\text{current}} \leftarrow k$
end for
{2. Adjust network depth to match the target buffer size}
$\text{delta} \leftarrow \text{buffer}_{\text{current}} - B_{\text{target}}$
if delta > 0 **then**
  Remove delta deepest blocks from the network
else if delta < 0 **then**
  Add $(-\text{delta})$ new blocks to the network
end if
return Updated network

EXPERIMENTS AND RESULTS

We validate SCALE-RL with an AlphaZero-style agent for Othello and use its standard terminology, such as Monte Carlo Tree Search (MCTS) and self-play. For a full review of the AlphaZero algorithm, we refer interested readers to the original seminal paper (Silver et al., 2018) and to our open-source code for implementation details. All experiments were conducted using two NVIDIA RTX Ada Generation GPUs, and the Othello engine is based on the Leela Zero engine (Pascutto and contributors, 2023).

Baseline and Datasets We first trained a standard AlphaZero agent for Othello, using the widely adopted configuration of a shared 40-layer convolutional ResNet backbone for the policy and value networks. This agent

served as our fixed-depth baseline throughout all experiments. The self-play process produced a large dataset of games, providing labeled positions of progressively increasing difficulty. For final evaluation, we also employed WTHOR (Fédération Française d'Othello, 2024), a canonical dataset containing 7,544,559 positions from 130,000 professional Othello games.

Evaluation Metrics To validate the claim that SCALE-RL enables more efficient and sustainable DRL agents, we measured five metrics:

- **Architectural Efficiency:** The number of active layers during training and the L_1 norms of layer outputs, to confirm that pruned layers contributed negligibly.
- **Game-Playing Strength:** Performance in head-to-head matches against Edax (Coudrain, 2024), a strong Othello engine, and accuracy in predicting game outcomes from WTHOR professional games.
- **Computational Cost:** The total number of floating-point operations (FLOPs) saved by our framework compared to the baseline.
- **Environmental Impact:** Estimated energy consumption and associated CO_2 emissions, derived from measured FLOPs and established energy-per-FLOP benchmarks.
- **Interpretability:** Exploratory analysis of layer activations to assess the interpretability of the learned representations.

Experiment 1: Supervised Off-Policy Validation Our first experiment evaluates SCALE-RL in a controlled, off-policy supervised setting. We modified the baseline 40-layer ResNet tower by removing all skip connections and batch normalization layers and inserting a GloNet layer before the policy and value heads. The model receives, for each player, an 8-timestep history of the Othello board as input, with the policy head outputting a probability distribution over 65 moves and the value head predicting a scalar in $[-1, 1]$ indicating the winner. The network was trained on the dataset of self-play games generated by the baseline AlphaZero agent, with the convolutional tower's depth fixed at 40 layers.

After training, we evaluated performance while progressively deactivating layers from the top of the tower, measuring policy loss, value loss, and the L_1 norm of intermediate representations. The results are shown in Figure 1. This post-hoc ablation assesses how many layers the trained agent effectively relies upon. SCALE-RL effectively utilizes a substantially smaller subset of the available layers: the first 18-20 layers are sufficient for accurate predictions, while deeper layers provide minimal or even negative contributions. This supports our claim that SCALE-RL can self-adapt its effective depth.

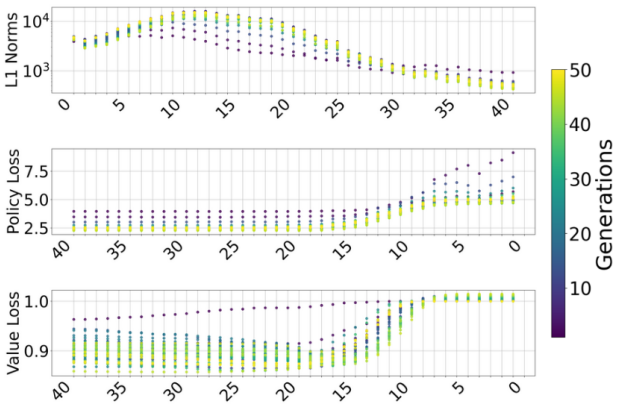
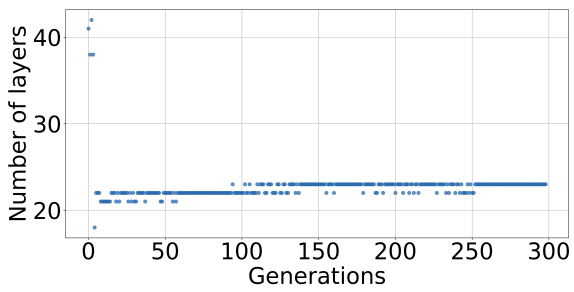


Figure 1: **Supervised Learning Ablation Study.** Layer utilization after training on Othello self-play data. **Top:** The L_1 norm of the output for each of the 40 layers, showing diminishing contributions from deeper layers. **Middle:** Policy loss as layers are cumulatively deactivated from the top (layer 40) down. The loss plateaus after the first 18 layers, indicating the remaining 22 layers are redundant for the policy head. **Bottom:** Value loss, calculated similarly. Performance is optimal with the first 18 layers, while including deeper layers increases the loss.

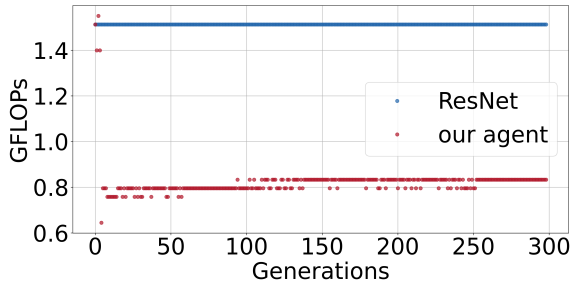
Experiment 2: On-Policy Self-Play Training Building on these results, we assessed SCALE-RL in a full on-policy DRL setting where the agent improves by generating its own experience. The agent was trained from scratch via AlphaZero-style self-play, iterating between generating new games (3,200 per generation) and performing supervised updates on this data, with MCTS guiding both processes. The Dynamic Depth Controller (Algorithm 1) with parameters $\tau = 1\%$ and $B_{\text{target}} = 5$ was active throughout, allowing the network to both prune and grow its layers. $\tau = 1\%$ and $B_{\text{target}} = 5$ were selected through a preliminary grid search as the best-performing configuration. This setup challenges the agent to adapt its policy and architecture simultaneously.

As shown in Figure 2a, SCALE-RL quickly recognized that a much shallower architecture was sufficient, reducing the initial 40-layer network to a stable range of 20–23 layers for most of the 300-generation run. The slightly higher number of active layers compared to Experiment 1 (20–23 vs. 18-20) likely reflects the increased complexity of the on-policy learning task. This confirms that SCALE-RL can automatically discover a problem's intrinsic complexity and reduce its computational footprint accordingly. The training and validation learning curves for the policy and value heads are shown in Figure 3.

Figure 4 shows how SCALE-RL responds to layer pruning



(a)



(b)

Figure 2: **a.** Number of active layers in the tower for each generation. **b.** Baseline vs our agent GFLOPs.

changes over training. For any given generation, both the policy and value losses increase as more layers are removed from the top of the tower. This indicates that SCALE-RL concentrates most of the important feature extraction in the earlier layers, making the deeper layers less critical. In contrast, the L_1 norms of the layer outputs at the end of training display a non-monotonic pattern. This suggests that, while the initial layers are crucial for learning, certain deeper layers become important as well, possibly because they capture more global, board-wide patterns that are essential for Othello.

Game-Playing Strength To ensure that improved architectural efficiency did not come at the expense of playing strength, we evaluated our agent against the fixed-size ResNet baseline on two complementary benchmarks. The first is the winner-prediction accuracy on the WTHOR dataset: in Figure 5a we report the MSE between the agent's value prediction and the true game outcome, normalized to the $[0, 1]$ range (lower is better). The second is head-to-head performance against Edax, a strong Othello engine configured to a search depth of 5: in Figure 5b we report the winrate averaged over 100 games.

Results show that, despite the gap in the figure, both agents achieve a superhuman level of play. Thus, a substantial gain in efficiency is achieved at the cost of a small, measurable decrease in asymptotic performance.

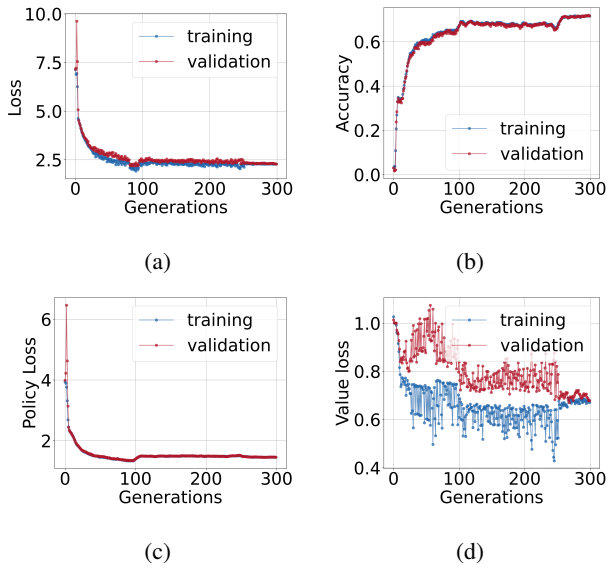


Figure 3: **Self-Play Training Performance over 300 Generations. (a)** Total loss on the training and validation sets, combining the policy, value, and regularization terms. **(b)** Accuracy of the policy head against the MCTS target move. **(c)** Policy loss for training and validation. The curves closely track each other, indicating robust generalization for the move prediction task. **(d)** Value loss for training and validation. The persistent gap suggests that generalizing game outcome predictions is more difficult.

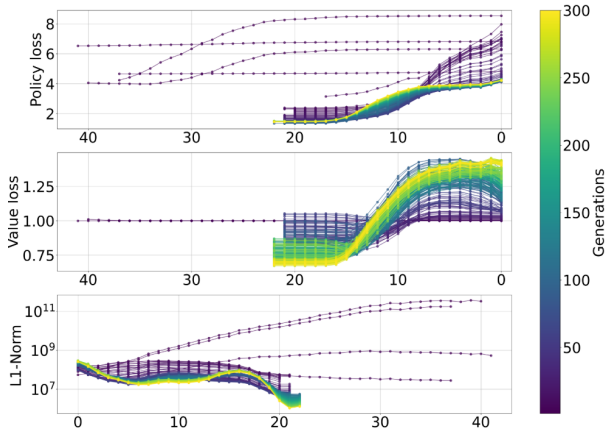
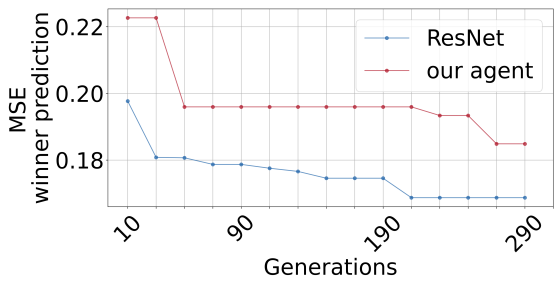
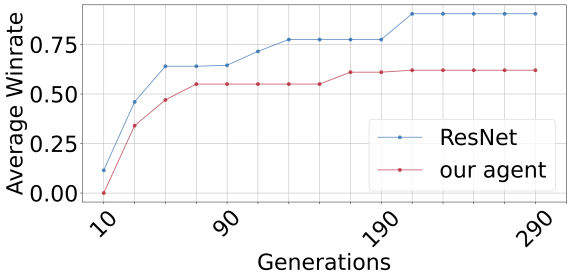


Figure 4: **Layer Adaptation Metrics Over Generations. Top** and **middle:** Policy and value loss (y-axis) as a function of network depth. On the x-axis the number of active layers after Algorithm 1 truncates the tower. **Bottom:** L_1 norm of each layer's output.

Relative FLOPs Savings via Depth Adaptation Since both models were trained under identical protocols, any computational-efficiency gap depends directly on network depth: it is linear in the baseline ResNet depth and in the dynamically adapting depth of our agent. Thus, by reducing



(a)



(b)

Figure 5: **Game-Playing Strength.** On the x-axis, the generation. **(a)** Best (minimum) MSE up to each generation between the agent's value prediction and the true outcome from the WTHOR dataset. The raw MSE is normalized by 0.25 to yield a more interpretable [0, 1] metric (lower is better). **(b)** Best (maximum) win rate up to each generation against Edax (search depth 5), averaged over 100 games.

its depth from 40 to 20–23 layers, our agent achieves an approximately 50% reduction in GFLOPs, see Figure 2b. We stress that this relative saving is contingent on the baseline's depth (here, 40 layers, as in AlphaZero's architecture). Had the baseline been shallower (e.g., 20 layers), our dynamic agent would likely have expanded its architecture to reach a similar size, possibly outperforming a static model thanks to its ability to self-adjust. This supports the claim that SCALE-RL can mitigate suboptimal architectural choices.

Computational Overhead The dynamic nature of SCALE-RL makes a precise *a priori* estimation of computational overhead challenging. Nevertheless, its dominant costs can be identified by examining the supervised training phase. The primary source of overhead arises from the initial layer-depth evaluation procedure (Algorithm 1, lines 4–11), whereas the primary savings stem from reduced inference during large-scale self-play.

This procedure iteratively assesses the performance impact of removing successive layers. All reported measurements correspond to a single training generation (supervised training plus self-plays). For an initial network with L_{init} layers, it computes the validation loss, over n_{val} samples, after removing one, two, and so on, up to L_{removed} layers, until the loss increases by at least τ. The computational complexity of this search is $O(n_{\text{val}} \cdot L_{\text{init}} \cdot L_{\text{removed}} \cdot C_{\text{layer}})$, where C_{layer} denotes the GFLOPs associated with a single layer. Once the optimal depth is identified, a buffer of B_{target} layers is preserved to define the final architecture.

Compared with the inference cost required to generate self-play trajectories in the AlphaZero algorithm, this overhead is relatively minor. In AlphaZero, the network must perform inference on the order of $O(n_{\text{selfplay}} \cdot n_{\text{visits}} \cdot n_{\text{moves}})$, where, in our case, $n_{\text{selfplay}} = 3200$, $n_{\text{visits}} = 400$, and n_{moves} is variable but typically around 60.

Consequently, the $\sim 7.68 \times 10^7$ forward passes per generation required by AlphaZero make the depth evaluation overhead practically negligible.

Energy Efficiency and Environmental Impact The dynamic depth adaptation achieved by SCALE-RL translates into substantial energy savings during training. Empirical measurements reveal that each training generation requires approximately 1.5 GFLOPs for the ResNet baseline compared to 0.8 GFLOPs for SCALE-RL, representing a 46.7% reduction in computational load per generation.

To quantify the environmental implications of this efficiency gain, we analyze the energy consumption across the complete 300-generation training cycle. The total computational savings amount to $(1.5 - 0.8) \times 300 = 210$ GFLOPs over the entire training process. When accounting for forward and backward passes during training, the effective computational reduction increases to approximately 630 GFLOPs total. Modern GPU hardware achieves energy efficiency ratings in the range of 10^{-11} to 10^{-12} Joules per FLOP (Strubell et al., 2019). Using a conservative estimate of 1.5×10^{-11} J/FLOP for contemporary training infrastructure (Patterson et al., 2021), the energy savings from our approach translate to:

$$\Delta E = 0.63 \times 10^{12} \times 1.5 \times 10^{-11} = 9.45\,\text{J} \qquad (1)$$

While this represents the energy consumption for our specific Othello benchmark, the scaling properties of SCALE-RL suggest proportionally larger savings for more computationally intensive RL applications. For training regimens comparable to those employed in modern large-scale DRL systems, where computational requirements can exceed 10^{20} FLOPs (Brown et al., 2020), our 46.7% efficiency improvement would yield energy reductions measured in megawatt-hours.

The carbon footprint reduction depends on the electrical grid's carbon intensity, which varies significantly by geographic region from 0.1 to 1.0 kg CO_2-eq/kWh (Moro and Lonza, 2018). For training conducted on typical computational infrastructure with a carbon intensity of 0.5 kg CO_2-

eq/kWh, SCALE-RL efficiency gains directly translate to reduced greenhouse gas emissions proportional to the computational savings achieved.

Beyond the immediate environmental benefits, the consistent computational efficiency demonstrated across our 300-generation training cycle suggests that SCALE-RL offers a promising pathway toward sustainable DRL. The ability to automatically discover and maintain optimal network complexity eliminates the computational waste inherent in fixed over-parameterized architectures, addressing growing concerns about the environmental cost of AI research (Schwartz et al., 2020) while maintaining competitive performance through principled architectural adaptation.

ANALYSIS OF INTERPRETABILITY

In addition to computational efficiency and performance, SCALE-RL offers inherent interpretability. This property arises directly from the GloNet architecture's connection to Generalized Additive Models (GAMs) (Hastie and Tibshirani, 1990) and Neural Additive Models (NAMs) (Agarwal et al., 2021), enabling a deeper understanding of the agent's decision-making process without the need for complex post-hoc analysis tools. In this section, we provide both theoretical and empirical perspectives, showing how the additive structure allows for both attention visualization and counterfactual analysis.

Theoretical Analysis: NAMs for Reinforcement Learning The explainability of SCALE-RL is mathematically grounded in the structure of GloNet, which can be viewed as a specialized Neural Additive Model (NAM) adapted for sequential decision making. In contrast to traditional deep networks, where decisions result from complex nonlinear compositions, $f(s) = f_L \circ f_{L-1} \circ \ldots \circ f_1(s)$, the GloNet architecture aggregates intermediate representations of the input s through summation. This additive aggregation effectively decomposes the value function into interpretable components, each corresponding to a distinct depth in the network:

$$v(s) = H\left(\sum_{\ell=1}^{L} G_\ell(s)\right) = H\left(\sum_{\ell=1}^{L} \phi_\ell(s)\right) \qquad (2)$$

where each $G_\ell(s) = \phi_\ell(s)$ represents a learned basis function operating at depth ℓ, analogous to the feature networks in NAMs but adapted for hierarchical spatial reasoning, and H is the value head.

This connection to GAMs provides theoretical guarantees about the model's interpretability. The additive structure ensures that each layer's contribution can be analyzed independently while maintaining the global decision context, enabling two complementary forms of analysis. First, we can examine the **attention patterns** by visualizing the magnitude of activations $|G_\ell(s)|$ at each layer, revealing what the network focuses on at different levels of abstraction. Second, we can perform **counterfactual analysis** through gradient computation, showing how policy probabilities would change with respect to input modifications.

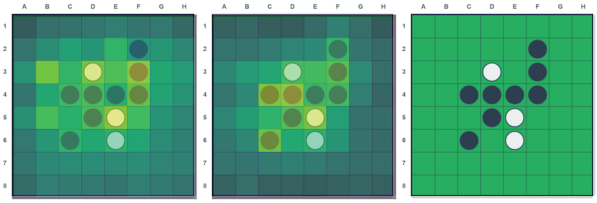
The gradient of the value with respect to the input state provides direct insight into the decision-making process:

$$\frac{\partial v(s)}{\partial s} = \frac{\partial H}{\partial x_{L+1}} \cdot \sum_{\ell=1}^{L} \frac{\partial G_\ell(s)}{\partial s} \qquad (3)$$

Here, $x_{L+1} = \sum_{\ell=1}^{L} G_\ell(s)$ is the sum of the GloNet block outputs before the prediction head. Thus, the gradient decomposes into a weighted sum of the contributions from each block, illustrating how each network depth influences the value's sensitivity to changes in the input state.

Empirical Analysis Following established practices in gradient-based attribution (Sundararajan et al., 2017), we apply min-max scaling to map gradient values to the range $[0, 1]$, where darker blue regions indicate positions the model considers disadvantageous for the current player. This provides a counterfactual interpretation: "how would the policy change if we could modify this board position?".

Our analysis reveals distinct patterns in how the network processes strategic information across different layers. Figure 6 demonstrates this hierarchical structure through three complementary visualizations, while Figure 7 shows the gradient-based counterfactual analysis revealing the model's learned strategic principles.



(a) Conv8 Layer (b) Global Attention (c) Game Position

Figure 6: **Interpretability Analysis of Decision Making. (a)** Early layer attention patterns showing focus on board edges and corners, demonstrating the network's awareness of positionally important regions even at shallow depths. **(b)** Global attention patterns aggregated across all layers, revealing the network's overall strategic focus. **(c)** Specific game position where the agent (playing as white) incorrectly chooses b5 instead of stronger moves like c3, e3, c5, or d6, illustrating how gradient analysis can identify the source of strategic errors.

The attention visualizations reveal a clear hierarchical pattern consistent with the additive model framework:

Early Layers (1-8) exhibit strong attention to board edges and corners, consistent with fundamental Othello strategy

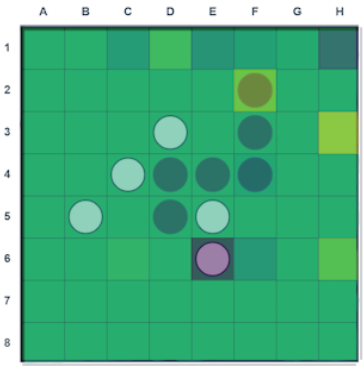


Figure 7: **Gradient Analysis Reveals Strategic Understanding.** Gradient magnitude visualization showing the model's learned preference to avoid positions adjacent to the h1 corner (marked in darker blue). The white player's actual move b5 is overlaid on the gradient map showing the value head's what-if analysis. Positions g1 and h2 show negligible gradients since these squares are strategically unfavorable for both players according to fundamental Othello principles. This counterfactual analysis demonstrates that the network has internalized corner-adjacency avoidance and confirms alignment between learned strategy and established game theory.

where controlling these positions provides significant advantages. This demonstrates that even shallow layers capture essential strategic concepts.

Middle and Deep Layers show more diffuse, global attention patterns that integrate information across the entire board state. These layers appear responsible for evaluating complex positional relationships and long-term strategic implications.

To illustrate the practical value of this interpretability, we analyzed a specific position from early training (epoch 200), see Figure 6c, where the agent makes a suboptimal move. Playing as white, the network chooses b5 despite stronger alternatives like c3, e3, c5, and d6 being available. Notably, the value head rates these positions as approximately equivalent, suggesting the error originates in the policy head's move selection rather than positional evaluation.

Our gradient analysis reveals the source of this error: while early convolutional layers correctly identify the superior moves, only at deeper layers does the weaker move b5 gain preference. This indicates that the network's long-range strategic reasoning has not yet stabilized during early training phases. Such insights demonstrate how the GloNet architecture enables practitioners to identify specific aspects of strategic reasoning that require additional training or architectural adjustment.

Counterfactual Analysis and Strategic Preferences The gradient-based visualization (Figure 7) provides evidence that our model has discovered fundamental Othello principles through self-play. The analysis shows the white player's actual move b5 overlaid on the gradient map, revealing the value head's what-if assessment of potential position changes.

The visualization demonstrates the model's learned strategic understanding: h1 appears as a high-magnitude negative gradient (the darkest blue region, representing the minimum value), while corner-adjacent squares g1 and h2 show near-zero gradients, indicating neutral strategic value. This pattern reveals that the network has internalized the fundamental Othello principle that corners are highly valuable (hence h1's negative gradient when occupied by the opponent) while adjacent squares offer no strategic advantage to either player.

The overlay of move b5 with gradient analysis proves valuable for debugging agent behavior, allowing verification that learned strategies align with established game theory. The zero gradients at g1 and h2 demonstrate that the model correctly recognizes when positions offer no strategic value to either player, while the strong negative signal at h1 confirms proper valuation of corner control.

LIMITATIONS AND FUTURE WORK

While the SCALE-RL framework demonstrates promising results in Othello, its generality beyond this setting remains to be established. Future work should extend empirical validation to other games like Go (Morandin et al., 2020, 2019), and to more diverse RL domains, including continuous control (Lillicrap et al., 2015), multi-agent (Hernández-Leal et al., 2019), pseudo-random numbers (Pasqualini and Parton, 2020a,b; Pasqualini et al., 2022), Wagner's framework (Angileri et al., 2025a,b), biomedical domains (Marchetti et al., 2025), and partially observable environments (Liu and Zhang, 2023; Parton et al., 2024).

The dynamic depth controller introduces computational overhead due to periodic validation evaluations. This may become prohibitive in large-scale settings or when validation data is expensive to generate. Distributed implementations and integration with efficiency techniques such as model distillation and quantization could mitigate this issue and improve scalability.

Our current approach relies on tuning the hyperparameters τ and B_{target} by hand, and focuses exclusively on depth adaptation. Extending the framework to simultaneously adapt network width, connectivity patterns, or attention mechanisms is a natural next step, but requires novel theoretical insights to handle more complex architectural adjustments.

Interpretability in our analysis benefits from the additive model structure but is most informative in environments where strategic concepts are interpretable. In higher-dimensional or less intuitive domains, new causal or transferable analysis tools may be necessary.

Finally, while SCALE-RL yields substantial energy savings in a modest benchmark, its impact and overhead in large-scale neural network settings remain to be fully assessed. Advances in principled task complexity metrics could further enable more targeted and efficient adaptation without costly empirical validation.

Acknowledgments

A. Di Cecco is a PhD student, National PhD in AI, 2nd round of 38th cycle, health and life sciences, UCBM.
M. Vegliò is a PhD student, National PhD in AI, 39th cycle, health and life sciences, UCBM.
M. Parton, and F. Morandin are partially funded by INdAM groups GNAMPA and GNSAGA.
Computational resources provided by the CLAI laboratory (Amato et al., 2024), University of Chieti-Pescara, Italy.
All authors are members of the CuriosAI research group and can be contacted collectively at curiosailab@gmail.com.

References

Rishabh Agarwal, Levi Melnick, Nicholas Frosst, et al. Neural additive models: Interpretable machine learning with neural nets. In *Advances in Neural Information Processing Systems 34*, pages 4699–4711, 2021.

Gianluca Amato, Alessia Amelio, Luciano Caroprese, et al. AI for Sustainability: Research at Ud'A Node. *CEUR Workshop Proceedings*, 3762:494–498, 2024.

Flora Angileri, Giulia Lombardi, Andrea Fois, et al. Analyzing RL components for Wagner's framework via Brouwer's conjecture. *Machine Learning*, 114(11):242, 2025a. doi: 10.1007/s10994-025-06890-2.

Flora Angileri, Giulia Lombardi, Andrea Fois, et al. A Systematization of the Wagner Framework: Graph Theory Conjectures and Reinforcement Learning. *Lecture Notes in Computer Science*, 15243 LNAI:325–338, 2025b. doi: 10.1007/978-3-031-78977-9_21.

Tom B. Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

Paul F. Christiano, Jan Leike, Tom B. Brown, et al. Deep reinforcement learning from human preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 4302–4310, 2017.

Léonard Coudrain. Edax: An open-source othello engine. https://github.com/abulmo/edax-reversi, 2024. Version 4.4, Accessed: 2025-07-25.

Antonio Di Cecco, Carlo Metta, Marco Fantozzi, et al. GloNets: Globally Connected Neural Networks. In *IDA 2024*, volume 14641 of *Lecture Notes in Computer Science*, pages 53–64, 2024.

Antonio Di Cecco, Andrea Papini, Carlo Metta, et al. Exploration and generalization in deep learning with SwitchPath activations. *Machine Learning*, 114, 2025a. doi: 10.1007/s10994-025-06840-y.

Antonio Di Cecco, Andrea Papini, Carlo Metta, et al. SwitchPath: Enhancing Exploration in Neural Networks Learning Dynamics. *Lecture Notes in Computer Science*, 15243 LNAI:275–291, 2025b. doi: 10.1007/978-3-031-78977-9_18.

Fédération Française d'Othello. Wthor: Othello database. https://www.ffothello.org/informatique/la-base-wthor, 2024. Accessed: 2025-07-25.

Trevor J Hastie and Robert J Tibshirani. *Generalized additive models*. Chapman & Hall/CRC, 1990.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. Identity mappings in deep residual networks. In *Computer Vision - ECCV 2016*, volume 9908 of *Lecture Notes in Computer Science*, pages 630–645, 2016b. doi: 10.1007/978-3-319-46493-0_38.

Pablo Hernández-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

G. Huang, Z. Liu, L. Van Der Maaten, et al. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR*, 2019.

Xiangyu Liu and Kaiqing Zhang. Partially observable multi-agent reinforcement learning with information sharing. *arXiv preprint arXiv:2308.08705*, 2023.

Alessandro Marchetti, Daniele Sasso, Federico D'Antoni, et al. Deep reinforcement learning for Type 1 Diabetes: Dual PPO controller for personalized insulin management. *Computers in Biology and Medicine*, 191, 2025. doi: 10.1016/j.compbiomed.2025.110147.

Carlo Metta, Marco Fantozzi, Andrea Papini, et al. Increasing biases can be more efficient than increasing weights. In *IEEE/CVF, Winter Conference on Applications of Computer Vision WACV*, pages 2798–2807, 2024. doi: 10.1109/WACV57701.2024.00279.

Carlo Metta, Marco Fantozzi, Andrea Papini, et al. Increasing biases can be more efficient than increasing weights. *Adv Data Anal Classif*, 19:437–468, 2025a. doi: 10.1007/s11634-025-00649-2.

Carlo Metta, Marco Fantozzi, Andrea Papini, et al. Improving performance in neural networks by dendrite-activated connection. *Studies in Classification, Data Analysis, and Knowledge Organization*, pages 133–141, 2025b. doi: 10.1007/978-3-031-84702-8_15.

Francesco Morandin, Gianluca Amato, Rosa Gini, et al. SAI: A sensible Artificial Intelligence that plays Go. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, 2019. doi: 10.1109/IJCNN.2019.8852266.

Francesco Morandin, Gianluca Amato, Marco Fantozzi, et al. SAI: A sensible artificial intelligence that plays with handicap and targets high scores in 9×9 go. In *ECAI 2020*, volume 325, pages 403–410, 2020. doi: 10.3233/FAIA200119.

Alberto Moro and Laura Lonza. Electricity carbon intensity in european member states: Impacts on GHG emissions of electric vehicles. *Transportation Research Part D: Transport and Environment*, 64:5–14, 2018. doi: 10.1016/j.trd.2017.07.012.

Maurizio Parton, Andrea Fois, Michelangelo Vegliò, et al. Predicting the Failure of Component X in the Scania Dataset with Graph Neural Networks. In *IDA 2024*, volume 14642 of *Lecture Notes in Computer Science*, pages 251–259, 2024. doi: 10.1007/978-3-031-58553-1_20.

Gian-Carlo Pascutto and contributors. leela-zero: Go engine with no human-provided knowledge. https://github.com/leela-zero/leela-zero, 2023.

Luca Pasqualini and Maurizio Parton. Pseudo random number generation through reinforcement learning and recurrent neural networks. *Algorithms*, 13(11)(307):1–14, 2020a. doi: 10.3390/A13110307.

Luca Pasqualini and Maurizio Parton. Pseudo random number generation: a reinforcement learning approach. *Procedia Computer Science*, 170:1122–1127, 2020b. doi: 10.1016/j.procs.2020.03.057.

Luca Pasqualini, Maurizio Parton, Francesco Morandin, et al. Score vs. winrate in score-based games: which reward for reinforcement learning? In *21st IEEE International Conference on Machine Learning and Applications, ICMLA 2022*, pages 573–578, 2022. doi: 10.1109/ICMLA55696.2022.00099.

David A. Patterson, Joseph Gonzalez, Quoc V. Le, et al. Carbon emissions and large neural network training. *arXiv:2104.10350*, 2021.

Hieu Pham, Melody Y. Guan, Barret Zoph, et al. Efficient neural architecture search via parameter sharing. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4092–4101, 2018.

Rafael Rafailov, Archit Sharma, Eric Mitchell, et al. Direct preference optimization: your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, et al. Progressive neural networks. *arXiv:1606.04671*, 2016.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588:604–609, 2020. doi: 10.1038/s41586-020-03051-4.

Roy Schwartz, Jesse Dodge, Noah A. Smith, et al. Green ai. *Communications of the ACM*, 63(12):54–63, 2020. doi: 10.1145/3381831.

David Silver, Aja Huang, Chris J. Maddison, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961.

David Silver, Thomas Hubert, Julian Schrittwieser, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362 (6419):1140–1144, 2018.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019. doi: 10.18653/v1/P19-1355.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3319–3328, 2017.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6105–6114, 2019.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition, ICPR 2016*, pages 2464–2469, 2016. doi: 10.1109/ICPR.2016.7900006.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Not Applicable
   (b) Complete proofs of all theoretical results. Not Applicable
   (c) Clear explanations of any assumptions. Not Applicable

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes
   (b) The license information of the assets, if applicable. Yes
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable
   (d) Information about consent from data providers/curators. Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable