

Universal Reasoning Model

Zitian Gao Lynx Chen Yihao Xiao He Xing Ran Tao

Haoming Luo Joey Zhou Bryan Dai *

Ubiquant

{ztgao02,ylchen,yhxiao,xyyang,rtao02,hmluo,jzhou,cbdai}
@ubiquant.com

Abstract

Universal transformers (UTs) have been widely used for complex reasoning tasks such as ARC-AGI and Sudoku, yet the specific sources of their performance gains remain underexplored. In this work, we systematically analyze UTs variants and show that improvements on ARC-AGI primarily arise from the recurrent inductive bias and strong nonlinear components of Transformer, rather than from elaborate architectural designs. Motivated by this finding, we propose the Universal Reasoning Model (URM), which enhances the UT with short convolution and truncated backpropagation. Our approach substantially improves reasoning performance, achieving state-of-the-art* 53.8% pass@1 on ARC-AGI 1 and 16.0% pass@1 on ARC-AGI 2. Our code is available at <https://github.com/zitian-gao/URM>.

1 Introduction

Recent advances in recurrent models [7, 11, 20] have demonstrated the effectiveness of Universal Transformers (UTs) [5] in addressing complex reasoning tasks, such as ARC-AGI and Sudoku [2, 3]. UT-based small models, despite being trained from scratch on these tasks without internet-scale pre-training, consistently outperform most standard Transformer-based Large Language models (LLMs) by a significant margin [20].

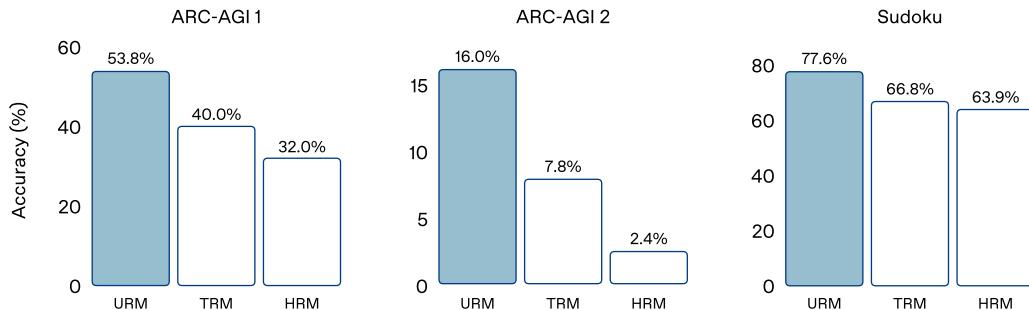


Figure 1: Performance comparison of UT-based models on the ARC-AGI and Sudoku benchmarks. ARC-AGI 1 and 2 scores are taken from the official ARC-AGI leaderboard for reliability.

* Corresponding author.

⁰This comparison focuses on pass@1 score of single small models trained from scratch under the same data setting as HRM and TRM, excluding test-time scaling, ensembling, and visual methods such as VARC [10].

While this contrast highlights the potential of UTs for depth-intensive iterative reasoning, the function and impact of gating mechanisms remain insufficiently explored beyond their initial intuition.

Prior studies often attribute improvements to high-level architectural innovations [7, 11, 20], yet our analysis reveals that the core performance gain actually arises from the often-overlooked recurrent inductive bias intrinsic to the Universal Transformer. In particular, nonlinear depth-wise computation plays a much larger role than previously acknowledged, suggesting that architectural modifications that enhance recurrent processing can yield substantial downstream improvements. Motivated by this insight, we further investigate and strengthen this inductive bias via a simplified yet effective enhancement to the UT framework, enabling stronger abstraction capabilities while preserving parameter efficiency.

Our main contributions are as follows:

- Through extensive ablation studies, we show that the performance of models on ARC-AGI-style complex reasoning tasks primarily stems from their nonlinearity. Moreover, we reveal that the true source of reasoning capability beyond standard Transformers comes from the recurrent mechanism of Universal Transformers rather than overly elaborate design in prior work.
- By introducing short convolutions and truncated backpropagation into the Universal Transformer, we achieve a state-of-the-art 53.8% pass@1 accuracy on ARC-AGI 1 and 16.0% on ARC-AGI 2.

2 Preliminaries

2.1 Standard Transformer

Let \mathcal{V} denote the vocabulary of size V , and let $\mathbf{x} = (x_1, \dots, x_N) \in \mathcal{V}^N$ be an input sequence of length N . We define the token embedding function as $\phi : \mathcal{V}^N \rightarrow \mathbb{R}^{N \times d}$, mapping discrete tokens to a d -dimensional continuous representation. Conversely, the unembedding function (or language modeling head) is denoted by $\psi : \mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{N \times V}$, which projects hidden states back to the vocabulary logit space.

A single Transformer layer, parameterized by θ , is defined as a function $\mathcal{T}_\theta : \mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{N \times d}$. This function typically composes a Multi-Head Self-Attention (MHSA) module and a Position-wise Feed-Forward Network (FFN), each wrapped with residual connections and layer normalization:

$$\begin{aligned} \mathcal{T}_\theta(H) &= \text{FFN}(\text{LN}(H' + H)), \\ \text{where } H' &= \text{MHSA}(\text{LN}(H)) \end{aligned}$$

A standard, non-recursive Transformer model \mathcal{M}_{std} of depth L is constructed by stacking L layers with distinct parameters $\Theta = \{\theta_1, \dots, \theta_L\}$. The forward pass is the composition of these layers:

$$\mathcal{M}_{\text{std}}(\mathbf{x}) = \psi \circ \mathcal{T}_{\theta_L} \circ \dots \circ \mathcal{T}_{\theta_1} \circ \phi(\mathbf{x})$$

Here, the operator \circ denotes function composition. The computational cost and parameter count both scale linearly with L , creating a rigid coupling between model capacity and inference compute.

2.2 Universal Transformer

The Universal Transformer (UT) [5] extends the standard Transformer [18] by introducing *recurrent computation over depth*. Instead of stacking L distinct layers, the UT applies a single transition block repeatedly to refine token representations. For an input sequence \mathbf{x} with embedding matrix $\mathbf{H}^0 \in \mathbb{R}^{n \times d}$, the UT updates states as

$$\mathbf{H}^{t+1} = \text{LayerNorm}(\mathbf{H}^t + \text{MHA}(\mathbf{H}^t)),$$

followed by a shared position-wise transition function

$$\mathbf{H}^{t+1} \leftarrow \text{LayerNorm}(\mathbf{H}^{t+1} + \text{Transition}(\mathbf{H}^{t+1})), \quad t = 0, \dots, T-1,$$

where Transition is either a feed-forward network or separable convolution. To encode both position and refinement depth, UT adds 2-D sinusoidal embeddings at each step.

2.2.1 Parameter Sharing

A key design of UT is *weight tying* across depth. The attention and transition parameters

$$\Theta_{\text{UT}} = \{\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V, \mathbf{W}_h^O, \Theta_{\text{Transition}}\}$$

are reused for all t . Thus, the model performs iterative representation refinement with a flexible number of steps T , enabling (i) depth adaptation at inference and (ii) higher theoretical expressivity than fixed-depth Transformers.

2.2.2 Adaptive Computation Time (ACT)

With ACT [9], different tokens may halt at different recurrent steps. At step t , each position predicts a halting probability

$$p_{t,i} = \sigma(\mathbf{w}^\top \mathbf{h}_{t,i} + b),$$

accumulated until reaching threshold $1 - \epsilon$. The final token representation is a weighted mixture

$$\mathbf{h}_i^{\text{final}} = \sum_t \Delta_{t,i} \mathbf{h}_{t,i},$$

where $\Delta_{t,i}$ is the truncated allocation. ACT allows UT to allocate more computation to complex tokens and less to simpler ones.

3 Universal Reasoning Model

The base architecture of our Universal Reasoning Model (URM) closely follows that of the Universal Transformer [5], with the difference being its decoder-only design. This aspect is consistent with previous works such as HRM [20] and TRM [11]. Our work differs from previous models [11, 20] by introducing the following **ConvSwiGLU** module and a **Truncated Backpropagation Through Loops** mechanism.

3.1 ConvSwiGLU

To strengthen the non-linearity of Universal Transformer, we introduce a **ConvSwiGLU** (motivation see Section 4.6), which augments the standard SwiGLU feed-forward block with a depthwise short convolution. Unlike the conventional point-wise SwiGLU [16], which treats each token independently, our design explicitly injects *local contextual interactions* into the gating mechanism, introducing lightweight channel mixing in token space without increasing sequence-level complexity [1, 22].

Given an input sequence $X \in \mathbb{R}^{T \times d}$, we first project it into an expanded intermediate representation:

$$[\mathbf{G}, \mathbf{U}] = XW_{\text{up}} \in \mathbb{R}^{T \times 2m}.$$

The SwiGLU activation produces a gated representation:

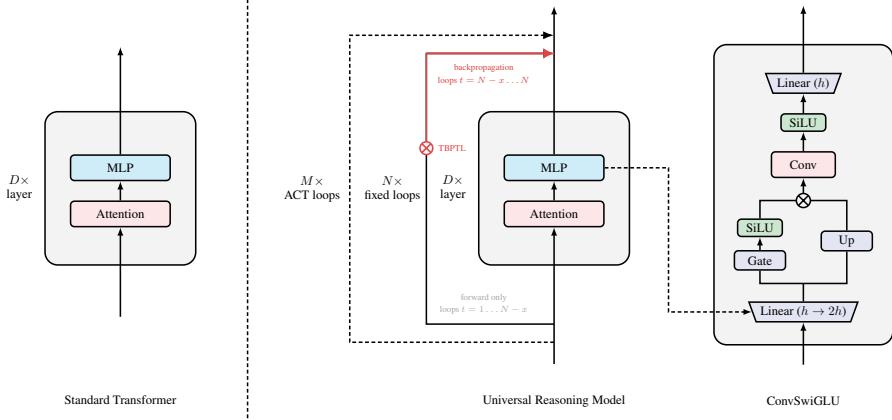


Figure 2: Illustration of our Universal Reasoning Model (URM) architecture. The left shows a standard Transformer layer stack, while the right illustrates the URM with fixed loops, ACT loops, and the ConvSwiGLU module. For illustrative purposes, components such as embeddings, residual connections, RMSNorm, positional encodings, and other modules are omitted, x in right figure represents the first x loops of the inner loop in forward-only mode, TBPTL represents our proposed Truncated Backpropagation Through Loops.

$$\mathbf{H}_{\text{ffn}} = \text{SiLU}(\mathbf{G}) \odot \mathbf{U}.$$

To integrate short-range token interactions, we apply a *depthwise 1D convolution* over the gated features:

$$\mathbf{H}_{\text{conv}} = \sigma(\mathbf{W}_{\text{dwconv}} * \mathbf{H}_{\text{ffn}}),$$

where $\mathbf{W}_{\text{dwconv}} \in \mathbb{R}^{m \times 1 \times k}$ is a depthwise convolution kernel of size $k = 2$.

Finally, the output is projected back to the hidden dimension:

$$\boxed{\mathbf{Y} = [\sigma(\mathbf{W}_{\text{dwconv}} * (\text{SiLU}(\mathbf{G}) \odot \mathbf{U}))] W_{\text{down}}}.$$

3.2 Truncated Backpropagation Through Loops

When the number of recurrent reasoning loops becomes large, the gradients propagated from early loops may hinder optimization due to noise accumulation and instability (see empirical evidence in Section 4.5). To alleviate this issue, we employ **Truncated Backpropagation Through Loops (TBPTL)** and only compute gradients for the later loops.

Consider a D -layer Universal Reasoning Model unrolled for M iterative loops during training. Let $\mathbf{h}_t^{(d)}$ denote the hidden representation of layer $d \in \{1, \dots, D\}$ at iteration $t \in \{1, \dots, M\}$. The recurrent transition is defined as:

$$\mathbf{h}_t^{(d)} = F_\theta^{(d)}(\mathbf{h}_t^{(d-1)}, \mathbf{h}_{t-1}^{(d)}),$$

where $F_\theta^{(d)}$ denotes the parameterized transformation at layer d with trainable parameters θ .

Instead of backpropagating through all M loops, we partition the rollout into *forward-only* and *trainable* segments. Specifically, for a truncation index $N < M$:

$$\underbrace{\{1, 2, \dots, N\}}_{\text{no backward pass}}, \quad \underbrace{\{N+1, \dots, M\}}_{\text{forward + backward}}.$$

During training, we compute gradients only on the loss accumulated in the latter $(M - N)$ loops:

$$\mathcal{L}_{\text{TBPTL}}(\theta) = \sum_{t=N+1}^M \mathcal{L}(\mathbf{h}_t^{(D)}, y),$$

where $\mathcal{L}(\cdot)$ is cross-entropy loss function. The gradients with respect to θ are thus:

$$\nabla_{\theta} \mathcal{L}_{\text{TBPTL}} = \sum_{t=N+1}^M \frac{\partial \mathcal{L}}{\partial \mathbf{h}_t^{(D)}} \frac{\partial \mathbf{h}_t^{(D)}}{\partial \theta}.$$

Example. For a configuration with $D = 4$ layers and $M = 8$ inner loops, we choose $N = 2$ forward-only loops. Thus, only the last 6 loops (*i.e.*, $t = 3$ to $t = 8$) contribute to gradient computation.

4 Experiment

4.1 Experiment Settings

Our experimental setup largely follows HRM and TRM [11, 20]. We use the same datasets and augmented data as in prior work, and apply an exponential moving average (EMA) to model parameters to improve training stability, following [11]. All models are trained with the AdamAtan2 optimizer [6]. For ARC-AGI 1 and ARC-AGI 2, the main model learning rates are set to 1×10^{-4} and 3×10^{-4} , respectively, while the puzzle embedding uses a learning rate of 1×10^{-2} ; for Sudoku, the puzzle embedding learning rate is 1×10^{-4} . Weight decay is set to 0.1 for both the main model and puzzle embedding on ARC-AGI 1 and ARC-AGI 2, and to 1.0 for Sudoku, consistent with prior work. The model has 4 layers with hidden size 512 and 8 attention heads. The inner loop runs for 8 steps, with the first two steps being forward-only, while the outer loop employs Adaptive Computation Time (ACT) [9] with a maximum of 16 steps.

4.2 Main Results

	ARC-AGI 1				ARC-AGI 2				Sudoku pass@1
	pass@1	pass@10	pass@100	pass@1000	pass@1	pass@10	pass@100	pass@1000	
HRM	34.4	46.4	55.0	60.5	5.4	9.6	14.3	18.6	63.9
TRM	40.0	51.3	59.8	64.4	4.6	7.4	11.7	13.6	66.8
URM	53.8	71.3	80.4	85.1	16.0	26.9	34.3	41.3	77.6
w/o Short Conv.	45.3	62.6	72.0	78.3	-	-	-	-	-
w/o Trunc. Backprop.	40.0	54.4	64.5	70.5	-	-	-	-	-

Table 1: The performance of URM, TRM, and HRM on three complex reasoning tasks: ARC-AGI 1, ARC-AGI 2, and Sudoku. pass@n denotes the pass rate when sampling n answers from the model; a sample is considered correct if at least one of the n answers is correct. The scores of TRM and HRM in this table may differ from those shown in the teaser. This is because the teaser scores are taken directly from the official ARC-AGI leaderboard for rigor, whereas the scores in this table are reproduced from the official TRM and HRM repositories following their official evaluation procedures. Minor discrepancies may occur due to randomness.

As shown in Table 1, the Universal Reasoning Model (URM) achieves substantial improvements over prior UT-based approaches across all benchmarks. On ARC-AGI 1, URM reaches 53.8% pass@1, outperforming TRM (40.0%) and HRM (34.4%) by large margins. On ARC-AGI 2, URM obtains 16.0% pass@1, nearly tripling HRM and more than doubling TRM. A similar advantage appears on Sudoku, where URM achieves 77.6% accuracy, surpassing both TRM and HRM.

Notably, URM’s gains further widen under larger sampling budgets (e.g., pass@1000), indicating that iterative refinement enables richer candidate generation rather than brittle one-step predictions.

4.3 Why Universal Transformer?

Layer	Loop	Hidden Size	Params	FLOPs	pass@1	pass@10	pass@100	pass@1000
Vanilla Transformers								
2	1	256	1×	1×	0.75	3.75	5.75	7.00
2	1	384	1.5×	1.5×	2.75	4.13	6.75	9.13
2	1	512	2×	2×	3.63	6.00	8.88	11.00
2	1	768	3×	3×	2.75	5.00	7.38	9.13
4	1	256	2×	2×	4.25	8.25	10.50	13.88
4	1	384	3×	3×	2.88	5.88	8.38	10.13
4	1	512	4×	4×	5.13	9.00	10.50	12.63
4	1	768	6×	6×	5.63	9.25	10.75	12.25
6	1	256	3×	3×	4.63	8.75	11.75	13.38
6	1	384	4.5×	4.5×	5.00	9.38	11.25	13.25
6	1	512	6×	6×	7.88	11.13	13.75	15.63
6	1	768	9×	9×	8.13	12.13	16.13	17.88
8	1	256	4×	4×	6.88	11.25	13.63	15.63
8	1	384	6×	6×	7.00	11.38	13.13	14.63
8	1	512	8×	8×	8.50	12.75	15.75	17.13
8	1	768	12×	12×	10.63	17.38	21.50	23.25
16	1	1024	32×	32×	0.00	6.50	8.75	9.75
32	1	512	32×	32×	23.75	34.13	38.88	43.38
64	1	256	32×	32×	18.25	31.75	38.25	41.38
Universal Transformers								
2	8	512	2×	16×	36.25	50.75	61.25	66.88
4	8	512	4×	32×	40.00	54.38	64.50	70.50

Table 2: Comparison between vanilla Transformers and Universal Transformers under different model depths, hidden sizes, and loops. We report pass@ n results on ARC-AGI 1.

Table 2 demonstrates that the performance gains of Universal Transformers (UTs) on ARC-AGI 1 arise from substantially higher parameter efficiency rather than increased model scale or computational budget. With only 4× parameters, a UT achieves a pass@1 score of 40.0, dramatically outperforming vanilla Transformers that employ up to 32× more parameters yet remain markedly weaker. Simply scaling depth or width in vanilla Transformers yields diminishing returns and can even lead to performance degradation, highlighting a fundamental inefficiency in how parameters are used to support multi-step reasoning.

Crucially, this advantage persists even when computation is held constant. At 32× FLOPs, reallocating computation from deep, non-shared layers to recurrent refinement improves pass@1 from 23.75 for vanilla Transformers to 40.0 for UTs. This behavior is consistent with analyses of previous works [15], which argue that many reasoning tasks benefit more from iterative computation than from increasing the number of independent layers. In standard Transformers, additional FLOPs are often

spent on redundant refinement in higher layers, whereas recurrent computation converts the same budget into increased effective depth [15, 23].

This superior efficiency is driven by the recurrent inductive bias introduced by parameter sharing across depth. Through repeated application of a shared transformation, UTs realize iterative refinement that better aligns with the structure of algorithmic reasoning, while avoiding any increase in parameter count. Consequently, under both fixed parameter and fixed FLOPs budgets, UTs consistently outperform vanilla Transformers on reasoning tasks, making them particularly well suited for reasoning-intensive settings such as ARC-AGI, where multi-step abstraction is more critical than sheer scale.

4.4 Short Convolution

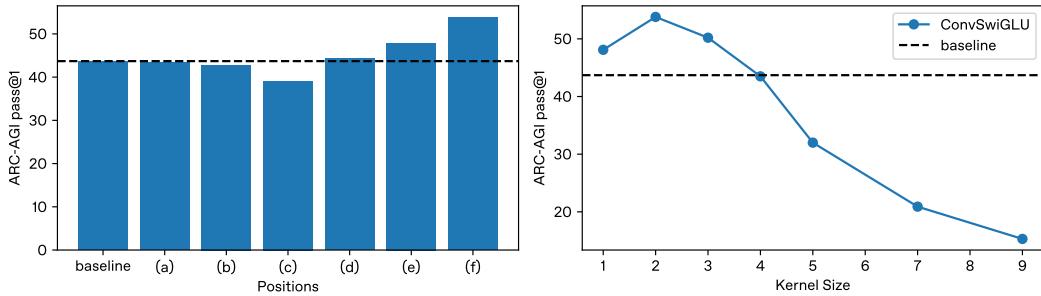


Figure 3: ARC-AGI pass@1 results for inserting the short convolution module at different positions within the UT transition (left figure), and varying the kernel size of the ConvSwiGLU module applied after the MLP expansion (right figure).

To strengthen the nonlinear inductive bias of the Universal Transformer, we introduce a depthwise short convolution module parameterized by W_{dwconv} (see Section 3.1 for details), which provides token-local mixing while preserving the per-step computational budget. Since ARC-AGI performance correlates strongly with nonlinear capacity (Section 4.6), we evaluate how inserting this module at different locations affects the recurrent transition.

We examine six insertion points: (a) after the SDPA output; (b) after the value projection; (c) after the key projection; (d) after the query projection; (e) between multi-head concatenation and the output projection; and (f) after the MLP expansion.

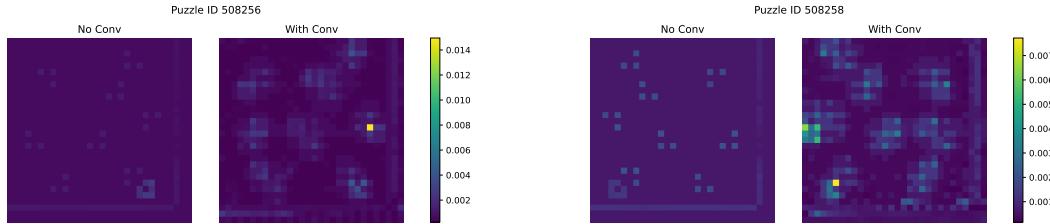


Figure 4: Visualization of the attention matrices after adding Short Convolution. The left figure shows the standard Universal Transformer, while the right figure shows the Universal Transformer with ConvSwiGLU applied.

As shown in Figure 3, inserting the W_{dwconv} module inside the attention pathway, positions (a)–(d), does not yield improvements and often degrades performance, suggesting that local perturbations interfere with the geometric structure of attention’s linear projections. A mild gain appears at position (e), where the perturbation acts only on aggregated multi-head features.

The dominant effect arises at position (f), after the MLP expansion, indicating that short-range mixing is most beneficial when applied within an already nonlinear subspace. This supports a functional interpretation in which the MLP—not attention—constitutes the model’s primary source of expressive nonlinearity; augmenting it with W_{dwconv} substantially enhances the model’s nonlinear representational capacity.

As shown in Fig. 4, the incorporation of short convolution into the MLP significantly enhances channel mixing. While the standard Universal Transformer exhibits relatively sparse and homogeneous attention patterns, the model with ConvSwiGLU produces attention matrices with more diverse and structured distributions. This suggests that short convolution facilitates more effective inter-channel information flow, thereby improving the expressiveness of the attention mechanism.

4.5 Truncated Backpropagation Through Loops

Loop w/ grad.	Loop w/o grad.	pass@1	pass@10	pass@100	pass@1000
8	0	36.25	50.75	61.25	66.88
7	1	37.75	49.13	59.50	65.88
6	2	39.13	53.50	61.88	66.88
5	3	39.50	51.63	60.88	65.25
4	4	38.75	50.50	61.50	65.88
3	5	36.88	49.00	57.75	63.88
2	6	34.25	46.25	55.75	61.75
1	7	22.50	37.00	45.38	52.38

Table 3: Effect of Truncated Backpropagation Through Loops (TBPTL) across inner loops on ARC-AGI 1. “Loop w/o grad.” denotes the number of forward-only inner-loop iterations, while “Loop w/ grad.” indicates the number of inner loops involved in backpropagation.

As shown in Table 5, when the total number of inner loops is fixed to 8, truncating gradients for the first two loops—i.e., running the initial two inner-loop iterations in forward-only mode—achieves the best performance. Both pass@1 and pass@1000 peak at this truncation setting, while shorter or longer truncation horizons result in inferior outcomes.

This trend closely resembles truncated backpropagation through time (TBPTT) in recurrent neural networks, where the underlying motivation is largely the same. In full backpropagation through time, gradients are propagated through the entire sequence, which incurs high computational and memory costs and often yields ineffective long-range gradients due to vanishing or exploding behaviors. As a result, practical implementations typically restrict gradient propagation to a fixed recent window, e.g., by backpropagating errors only through the last L time steps and updating the network parameters accordingly [14, 17].

Similarly, in universal transformers, propagating gradients across all inner-loop iterations can lead to unstable optimization, while overly aggressive truncation limits the model’s ability to coordinate multi-step refinement. Moderately truncating gradient propagation therefore provides a favorable balance between optimization stability and effective long-horizon learning.

We note that all results in this experiment are obtained using a two-layer URM without the short convolution module, which differs from the full URM model reported earlier.

4.6 Nonlinearity of Transformers

Model	pass@1	pass@10	pass@100	pass@1000
Full Universal Reasoning Model	53.75	71.25	80.38	85.13
w/o Short Conv.	45.25	62.63	72.00	78.25
SwiGLU → SiLU	29.75	42.13	50.00	54.50
SiLU → ReLU	28.63	43.38	50.63	54.88
w/o Attention Softmax	2.00	6.75	10.25	15.00

Table 4: Ablation study on nonlinearity architectural components of the Universal Reasoning Model. We report pass@ n results on ARC-AGI 1. All experiments are conducted under exactly the same settings as in Section 4.1.

As shown in Table 4, the performance on ARC-AGI 1 decreases monotonically as nonlinear components are progressively removed from the model. Among these components, the activation function in the MLP plays a particularly critical role: replacing SwiGLU with simpler nonlinearities such as SiLU or ReLU leads to substantial degradation, while completely removing the attention softmax results in a dramatic collapse in performance. This clear monotonic trend highlights the importance of strong nonlinear transformations for solving complex abstract reasoning tasks.

These results suggest that the expressive power required for ARC-AGI primarily arises from rich nonlinear mappings. Weakening the nonlinearity may systematically limit the model’s ability to represent complex reasoning skills.

We note that the model still retains certain forms of nonlinearity that are not ablated in this study, such as the RMSNorm applied after each layer and the dot-product interaction between queries and keys in attention. However, these components are either difficult to remove without causing training instability or represent relatively weak nonlinear effects compared to explicit activation functions. As ablating them typically leads to training failure, they fall outside the scope of the present analysis.

4.7 Muon Optimizer

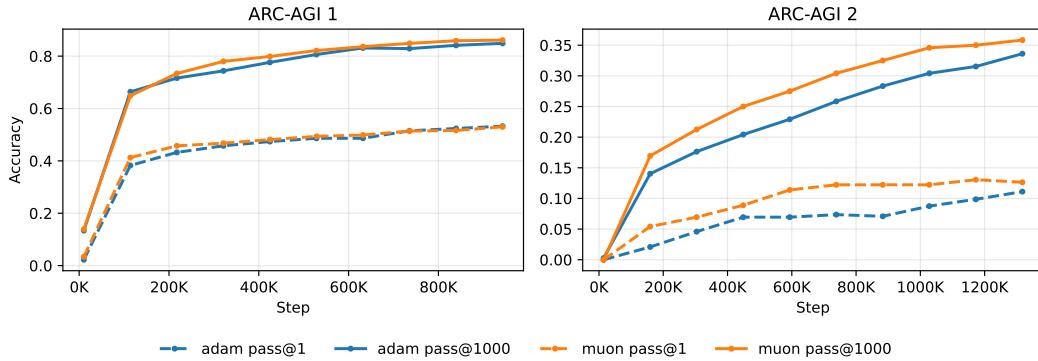


Figure 5: ARC-AGI pass@1 and pass@1000 performance of Adam and Muon optimizers on ARC-AGI 1 and ARC-AGI 2 benchmarks. Solid lines denote pass@1000, dashed lines denote pass@1, and colors indicate different optimizers. Training steps are shown in thousands (K).

To evaluate the training efficiency of the Universal Reasoning Model (URM), we compare the Muon (Momentum Updated Orthogonal Newton) optimizer [12] with a standard adaptive baseline,

Adamantan2 [6]. Muon approximates second-order curvature to apply orthogonal updates to better handle the complex loss landscapes [8] induced by deep recurrent structures. Both models are trained from scratch under identical experimental settings, including batch size, learning rate schedules, and data augmentation, ensuring that any observed differences arise solely from the choice of optimizer.

Across the ARC-AGI 1 and ARC-AGI 2 benchmarks, Muon demonstrates substantially faster convergence. On ARC-AGI 2, the Muon-optimized model reaches a pass@1 accuracy of 11.5% in approximately 600,000 training steps, whereas the Adamantan2 baseline requires over 1,300,000 steps to achieve the same performance, corresponding to nearly a twofold speedup in optimization. Despite this advantage in early training, both methods converge to similar final accuracies (approximately 53.8% on ARC-AGI 1 and 16.0% on ARC-AGI 2), indicating comparable asymptotic performance.

These results suggest a separation between optimization efficiency and architectural capacity in the URM. While Muon preconditions the challenging spectral properties of recurrent weight matrices [13] and reduces training cost, it does not lead to improved final generalization.

5 Related Work

5.1 ARC-AGI

Prior work on the ARC-AGI benchmark [2, 3] spans vision-based formulations, large language model (LLM) adaptation, and recurrent reasoning architectures. Vision-centric approaches such as Vision ARC [10] reformulate ARC as an image-to-image transformation problem and show that standard visual inductive biases can achieve competitive performance, particularly with ensembling and test-time scaling. LLM-based methods explore fine-tuning and test-time training, demonstrating that transient parameter updates outperform static in-context learning on ARC-like tasks. Beyond language and vision models, recurrent architectures emphasize iterative computation as a core mechanism for abstraction. The Hierarchical Reasoning Model (HRM) [7, 20] introduces multi-timescale recurrence and achieves strong ARC-AGI results, while subsequent analyses suggest that its gains may largely stem from recurrence rather than explicit hierarchy. The Tiny Recursive Model (TRM) [11] further simplifies this paradigm, showing that a single lightweight network applied recursively can match or exceed more complex hierarchical designs.

5.2 Universal Transformers (Looped Transformers)

The Universal Transformer (UT), also known as the Looped Transformer, was introduced by Dehghani et al. [5] as an extension of the standard Transformer with recurrent computation and adaptive computation time. Subsequent work has shown that UTs exhibit significantly stronger multi-step reasoning abilities than vanilla Transformers, as the recurrent refinement mechanism helps overcome architectural limitations in multi-hop reasoning tasks [4, 19]. In addition, UTs demonstrate improved algorithmic learning capabilities, enabling more effective modeling of iterative and rule-based computations [21]. By reusing parameters across refinement steps, UTs also achieve higher parameter efficiency, allowing more expressive computation without increasing model size [15].

6 Conclusion

We systematically investigate the sources of performance gains in Universal Transformer models on complex reasoning tasks. Extensive ablation studies reveal that these gains stem primarily from the recurrent inductive bias and strong nonlinear components of Transformer, rather than from overly complex architectural designs. Motivated by this insight, we propose the Universal Reasoning Model (URM), which enhances nonlinear depth-wise computation via short convolutional gating and improves optimization stability through truncated backpropagation through loops. URM achieves state-of-the-art performance on ARC-AGI 1 and 2.

References

- [1] Zeyuan Allen-Zhu. Physics of language models: Part 4.1, architecture design and the magic of canon layers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.
- [2] Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. Arc prize 2024: Technical report, 2025.
- [3] Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-ag1-2: A new challenge for frontier ai reasoning systems, 2025.
- [4] Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Punta Cana, Dominican Republic, November 2021.
- [5] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2019.
- [6] Katie Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.
- [7] Renee Ge, Qianli Liao, and Tomaso Poggio. Hierarchical reasoning models: Perspectives and misconceptions, 2025.
- [8] Zixuan Gong, Jiaye Teng, and Yong Liu. What makes looped transformers perform better than non-recursive ones (provably), 2025.
- [9] Alex Graves. Adaptive computation time for recurrent neural networks, 2017.
- [10] Keya Hu, Ali Cy, Linlu Qiu, Xiaoman Delores Ding, Runqian Wang, Yeyin Eva Zhu, Jacob Andreas, and Kaiming He. Arc is a vision problem!, 2025.
- [11] Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025.
- [12] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.
- [13] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025.
- [14] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.
- [15] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [16] Noam Shazeer. Glu variants improve transformer, 2020.
- [17] Corentin Tallec and Yann Ollivier. Unbiasing truncated backpropagation through time, 2017.
- [18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [19] Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [20] Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025.

- [21] Liu Yang, Kangwook Lee, Robert D Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations*, 2024.
- [22] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10819–10829, June 2022.
- [23] Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. Investigating layer importance in large language models. In *The 7th BlackboxNLP Workshop*, 2024.