Iterative Nonlinear Computation Underlying Abstract Reasoning

Zitian Gao Yilong Chen Yihao Xiao Xinyu Yang
Ran Tao Haoming Luo Joey Zhou Bryan Dai
IQuest Research

Abstract

Universal Transformers (UTs) show strong performance on abstract reasoning tasks such as ARC-AGI, yet the specific sources of their performance gains remain underexplored. We conduct systematic ablations over UT variants and find that performance improvements on abstract reasoning are driven primarily by (i) the recurrent inductive bias induced by parameter sharing across depth and (ii) strong nonlinearity inside the Transformer block, rather than by elaborate architectural heuristics. Motivated by this finding, we propose the Universal Reasoning Model (URM). First, we propose ConvSwiGLU, which augments the feed-forward block with a channel-wise short convolution to effectively enhance nonlinearity of UT. Second, we adopt Truncated Backpropagation Through Loops (TBPTL) to eliminate noise from early loop when training with many loops, by freezing the gradients of early loops. Our approach substantially improves abstract reasoning performance, achieving 57.5% pass@1 on ARC-AGI 1 and 16.0% pass@1 on ARC-AGI 2. More importantly, we pretrain a 8-billion-parameter large language model based on the URM architecture from scratch, and the results show that URM still substantially outperforms the baseline across all reasoning tasks.

1 Introduction

Recent progress in abstract reasoning has revealed the surprising effectiveness of small language models when equipped with appropriate inductive biases [12, 18, 31, 35]. Universal Transformers (UTs), which share parameters across depth, have emerged as strong baselines for tasks such as ARC-AGI, Sudoku, and Maze. They often outperform substantially larger standard Transformers trained under comparable settings.

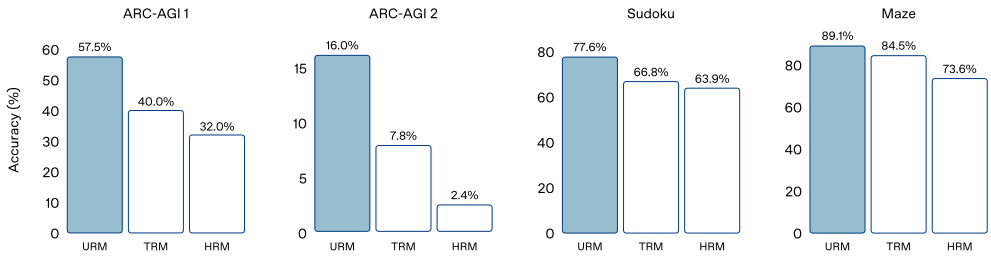

Figure 1: Performance comparison of UT-based models on the ARC-AGI, Sudoku and Maze tasks. ARC-AGI 1 and 2 scores are taken from the official ARC-AGI leaderboard for reliability.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Despite these empirical successes, the mechanisms underlying UTs' reasoning advantages remain poorly understood. Existing approaches to improve reasoning performance typically rely on scale (larger models, more test-time samples), architectural complexity (specialized modules or heuristics), or test-time scaling and ensembling methods[16, 18, 31, 35]. While effective, these strategies obscure a fundamental question: what minimal structural properties are actually required for abstract reasoning in neural networks? In particular, it remains unclear whether UTs benefit primarily from their recurrent bias, their nonlinear transformations, or from incidental architectural choices.

In this work, we take a step toward answering this question through systematic ablation and controlled experimentation. We analyze a range of UT variants under a strictly constrained setting: single small models trained from scratch, without test-time scaling, ensembling, or visual priors. Our analysis reveals two consistent findings. We find that both depth-wise parameter sharing and strong nonlinearities inside Transformer blocks are essential for abstract reasoning performance, weakening these leads to substantial degradation.

Motivated by this insight, we introduce the Universal Reasoning Model (URM), a Universal Transformer designed to strengthen nonlinear iterative computation with efficient and stable training. URM incorporates two modifications. First, ConvSwiGLU augments the standard feed-forward block with a short channel-wise convolution, enhancing nonlinear expressivity with minimal overhead. Second, we adopt Truncated Backpropagation Through Loops (TBPTL), which backpropagates gradients only through later recurrent iterations while freeze the gradients of early loops, enabling stable training with many loops. URM achieves state-of-the-art performance among single small models, reaching 57.5% pass@1 on ARC-AGI 1, 16.0% on ARC-AGI 2, 77.6% accuracy on Sudoku and 89.1% accuracy on Maze. These results demonstrate that stronger iterative nonlinearity enables small models to outperform much larger models with up to $32\times$ greater parameter efficiency. Importantly, we invested substantial computational resources to pretrain an Large Language Model (LLM) from scratch. Appendix B shows that the URM architectural principle also transfers to pretrained decoder-only LLMs, improving GSM8K from 8.5 to 16.5, MATH500 from 3.1 to 9.3, Big-Bench Hard from 38.6 to 44.1, ARC-Easy from 59.5 to 63.5 and ARC-Easy from 30.0 to 34.6.

Our main contributions are as follows:

- Through systematic ablations under strictly controlled settings, we identify the recurrent inductive bias induced by parameter sharing and strong nonlinearity as the primary drivers of Universal Transformers' performance on abstract reasoning, clarifying the mechanisms behind their reasoning gains beyond architectural heuristics.
- We propose ConvSwiGLU, a lightweight modification to the SwiGLU feed-forward block that injects local token interactions via a short channel-wise convolution, significantly enhancing nonlinear expressivity within each recurrent iteration at negligible computational and parameter cost.
- We introduce TBPTL, a simple yet effective training strategy that stabilizes optimization for Universal Transformers with many loop iterations by freezing early loops in a forward-only mode, it substantially enhances reasoning performance.
- We further show in Appendix B that URM scales to an 8-billion-parameter LLM pretrained from scratch, yielding consistent improvements across all downstream reasoning benchmarks including GSM8K, MATH500, Big-Bench Hard, ARC-Easy, and ARC-Challenge.

2 Preliminaries

2.1 Standard Transformer

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

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

$$\mathcal{T}_\theta(\mathbf{H}) = \text{FFN}(\text{LN}(\mathbf{H}' + \mathbf{H})),$$
$$\text{where } \mathbf{H}' = \text{MHA}(\text{LN}(\mathbf{H}))$$

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

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

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) [7] extends the standard Transformer [33] by introducing *recurrent computation over depth*. Instead of stacking L distinct layers, the UT repeatedly applies a single shared attention–FFN transition block. For an input sequence $\mathbf{x}$ with embedding matrix $\mathbf{H}^0 \in \mathbb{R}^{N\times d}$, the recurrent update is

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

followed by a shared position-wise feed-forward block (FFN):

$$\mathbf{H}^{t+1} \leftarrow \text{LN}\big(\mathbf{H}^{t+1} + \text{FFN}\big(\mathbf{H}^{t+1}\big)\big)\,,$$
$$t = 0, \ldots, T-1.$$

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

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

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

3 Universal Reasoning Model

The base architecture of our Universal Reasoning Model (URM) closely follows that of the Universal Transformer [7]. Our work differs from previous models [18, 31, 35] by introducing the **ConvSwiGLU** module and a **Truncated Backpropagation Through Loops (TBPTL)** mechanism. Pseudocode for these two components is provided in Appendix, Algorithms 1 and 2. While the we mainly focuses on controlled abstract reasoning tasks, we also evaluate the same design in a pretrained decoder-only LLM setting; the full setup and results are deferred to Appendix B.

3.1 ConvSwiGLU

Our ablation studies in Section 4.6 demonstrate that the primary source of abstract reasoning capability stems from the model's nonlinear representational capacity, which is consistent with the long-standing consensus in deep learning that the expressive power of deep networks fundamentally arises from the stacking of nonlinear activations [23].

Since the nonlinear components of Transformer models are mainly concentrated in the feed-forward blocks [28, 40], we focus our investigation on enhancing nonlinearity within the feed-forward block.

Gated Linear Units have been widely adopted in modern large language models [26]. Inspired by GLUs, we explore nonlinear enhancement mechanisms beyond conventional activation functions and accordingly propose ConvSwiGLU, which augments the standard SwiGLU feed-forward block with

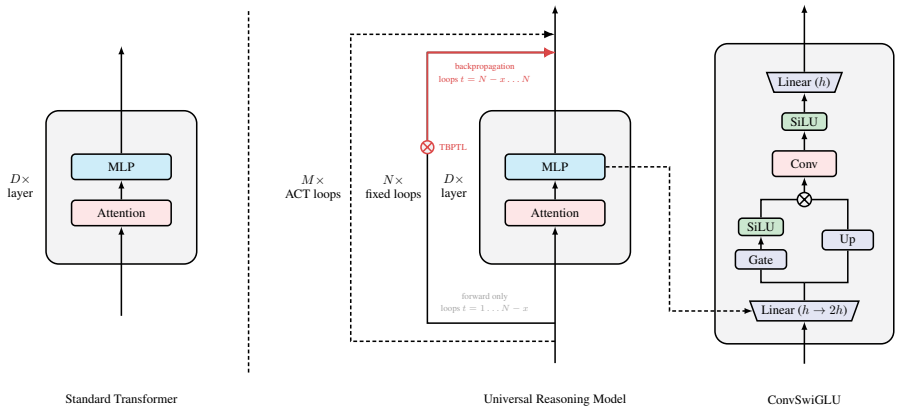

Figure 2: Illustration of our Universal Reasoning Model (URM) architecture. The left shows a standard Transformer layer stack, while the right illustrates the URM with fixed loops, ACT loops, and the ConvSwiGLU module. For illustrative purposes, components such as embeddings, residual connections, RMSNorm, 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.

a channel-wise short convolution. Unlike the conventional point-wise SwiGLU [26], 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 [2, 38].

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:

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

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

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

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

The convolution operation is a linear transformation. Specifically, for any feature map $\mathbf{X}$, the channel-wise convolution can be written as a linear operator $\mathcal{C}$ such that

$$\mathcal{C}(\mathbf{X}) = \mathbf{W}_{\text{dwconv}} * \mathbf{X}.$$

If the nonlinear activation $\sigma(\cdot)$ between successive convolutional or projection layers is removed, the composition of transformations becomes

$$\mathbf{Y} = \mathcal{L}_2\big(\mathcal{L}_1(\mathbf{X})\big),$$

where both $\mathcal{L}_1$ and $\mathcal{L}_2$ are linear operators. By closure under composition, there exists a single linear operator $\mathcal{L}$ such that

$$\mathcal{L}_2 \circ \mathcal{L}_1 = \mathcal{L}.$$

Consequently, the entire feed-forward block collapses into an effectively linear mapping, regardless of its depth [8, 19]. In this regime, stacking multiple layers does not increase the model's expressive power, and the benefits of deep iterative computation largely vanish [8, 19]. For this reason, we explicitly append a nonlinear activation $\sigma(\cdot)$ after the short convolution. Empirically, removing this activation results in a substantial performance degradation on abstract reasoning tasks, as shown in Table 1, confirming that the post-convolution nonlinearity is essential for preserving the model's representational capacity.

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

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

3.2 Truncated Backpropagation Through Loops (TBPTL)

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 $d \in \{1, \ldots, D\}$ at iteration $t \in \{1, \ldots, M\}$. The recurrent transition is defined as:

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

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 $N < M$:

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

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

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

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

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

4 Experiment

4.1 Experiment settings

Our experimental setup largely follows HRM and TRM [18, 35]. 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. All models are trained with the AdamAtan2 optimizer [9]. For ARC-AGI 1 and 2, the 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 and Maze, 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 and Maze, 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 executed in a forward-only mode, while the outer loop employs Adaptive Computation Time (ACT) [15] with a maximum of 16 steps. ACT is primarily used to stabilize training; it yields little to no performance improvement and incurs negligible additional computational overhead during training [31]. The convergence of ACT step counts throughout training is shown in Appendix F, as shown in the figure, ACT introduces little additional computational overhead and primarily serves to stabilize training, which is consistent with findings from prior work [31]. In addition to these absract reasoning tasks, we report a complementary pretrained LLM setting in Appendix B, where the same URM design is inserted into a matched Qwen3-8B backbone and pretrained on 300B sampled tokens.

4.2 Main results

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%

Table 1: The performance of URM, TRM, and HRM on 4 complex reasoning tasks: ARC-AGI 1, ARC-AGI 2, Sudoku and Maze. 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.

	ARC-AGI 1			**ARC-AGI 2**			**Sudoku**	**Maze**
Model	pass@1	pass@10	pass@100	pass@1	pass@10	pass@100	pass@1	pass@1
HRM	34.4	46.4	55.0	5.4	9.6	14.3	63.9	73.6
TRM	40.0	51.3	59.8	4.6	7.4	11.7	66.8	84.5
URM	**57.5**	**74.4**	**82.4**	**16.0**	**26.9**	**34.3**	**77.6**	**89.1**
Cumulative URM Ablation								
- SiLU after Conv.	50.2	64.5	75.0	12.5	21.9	33.1	73.2	84.2
- ConvSwiGLU	45.3	62.6	72.0	11.0	19.9	27.4	70.3	80.0
- TBPTL	40.0	54.4	64.5	10.1	18.8	26.0	68.2	74.7

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@100), indicating that iterative refinement enables richer candidate generation rather than brittle one-step predictions.

This advantage is not limited to the four from-scratch reasoning benchmarks in the main text. Appendix B reports the same trend in a pretrained language-model regime: under a matched Qwen3-0.6B backbone and equal-scale 300B-token pretraining, URM improves GSM8K from 3.7 to 6.5, Big-Bench Hard from 28.9 to 31.1, and ARC-Easy from 32.0 to 34.2 (Table 8). This complementary result strengthens our central claim that iterative nonlinear computation is a general architectural advantage rather than an artifact of grid-based task formatting.

4.3 Why Universal Transformer?

Table 2 shows that Universal Transformers (UTs) outperform vanilla Transformers on ARC-AGI 1 primarily through greater parameter efficiency rather than larger scale or compute. With only $4\times$ parameters, a UT attains a pass@1 score of 40.0, far surpassing vanilla Transformers using up to $32\times$ more parameters. Scaling depth or width alone yields diminishing returns and may even degrade performance, revealing inefficient parameter utilization for multi-step reasoning.

This advantage remains under equal computation budgets. At $32\times$ FLOPs, reallocating compute from deep non-shared layers to recurrent refinement raises pass@1 from 23.75 in vanilla Transformers to 40.0 in UTs. This aligns with prior work [25], which suggests reasoning tasks benefit more from iterative computation than from additional independent layers. In vanilla Transformers, extra FLOPs often produce redundant refinement, whereas recurrent computation converts the same budget into greater effective depth [25, 39].

The efficiency gain stems from the recurrent inductive bias induced by parameter sharing across depth. Repeated application of shared transformations enables iterative refinement better suited to abstract reasoning without increasing parameter count. As a result, under both fixed parameter and fixed FLOPs budgets, UTs consistently outperform vanilla Transformers on reasoning tasks, making them especially effective for reasoning-intensive settings such as ARC-AGI.

Table 2: Comparison between vanilla Transformers and Universal Transformers under different numbers of layers, hidden sizes, parameters, training FLOPs, and loop iterations. We report pass@n results on ARC-AGI-1. The table shows that Universal Transformers not only substantially outperform vanilla Transformers under matched parameters, but also exhibit a significant advantage under matched compute.

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**

4.4 Short Convolution

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 output projection; and (f) after MLP expansion.

As shown in Figure 3, inserting the W_{dwconv} module into the attention pathway at positions (a)–(d) fails to improve 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 ben-

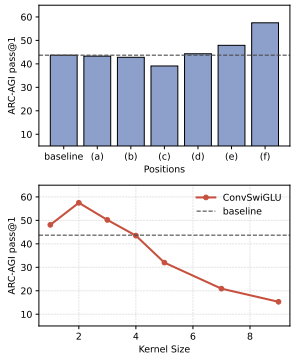

Figure 3: ARC-AGI pass@1 results for inserting the short convolution module at different positions within the UT transition (left), and varying the kernel size of ConvSwiGLU applied after the MLP expansion (right).

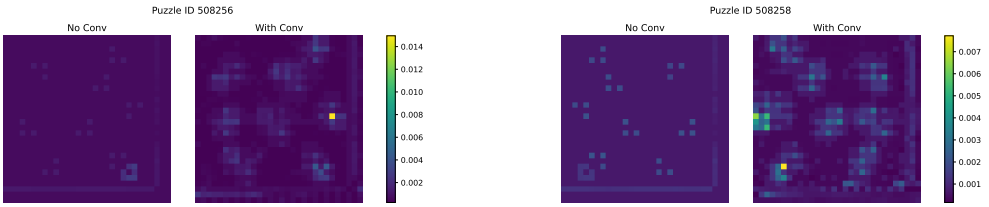

Figure 4: Visualization of the attention matrices after adding Short Convolution. The left figure shows the standard Universal Transformer, while the right figure shows the Universal Transformer with ConvSwiGLU applied.

eficial 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 [28, 40]; 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

As shown in Table 3, when the total number of loops is fixed to 8, truncating gradients for the first loop—i.e., running the first loop in forward-only mode—achieves the best average performance. Under this truncation setting, while shorter or longer truncation lead to degraded average performance. In contrast, when the total number of loops is fixed to 16, truncating gradients for the first four loops yields a substantially larger performance gain than truncating only the first loop when the total number of loops is 8, pass@1, pass@10 and pass@100 all reach their peak values. These results reveal two consistent trends: as the total number of loops increases, the optimal gradient truncation horizon also grows; moreover, under deeper loop unrolling, moderately increasing the number of truncated loops leads to more pronounced performance improvements.

Table 3: Effect of Truncated Backpropagation Through Loops (TBPTL) across different total 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.

Loop w/ grad.	Loop w/o grad.	pass@1	pass@10	pass@100
Total 8 Loops				
8	0	**50.50**	66.63	77.38
7	1	50.00	**68.63**	**78.01**
6	2	47.63	66.62	73.63
5	3	50.00	67.63	77.25
4	4	48.74	64.88	75.00
3	5	44.25	61.00	72.80
2	6	44.88	59.75	70.63
1	7	34.50	49.00	57.85
Total 16 Loops				
16	0	54.63	73.13	81.37
14	2	56.23	75.38	82.10
12	4	**57.50**	**76.75**	**84.50**
10	6	55.87	74.50	83.75
8	8	54.68	74.25	82.75
6	10	54.17	73.50	82.88
4	12	50.93	68.81	77.86
2	14	45.22	62.25	70.50

This trend closely resembles truncated backpropagation through time (TBPTT) in recurrent neural networks [22, 29, 30], 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 [24, 29].

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.

4.6 Nonlinearity of Transformers

Transformers derive their nonlinear representational capacity almost exclusively from the feed-forward blocks, while the attention mechanism itself remains largely linear aside from softmax and normalization [28, 40]. As shown in Table 4, as the nonlinearity in the feed-forward blocks is progressively weakened, the model's performance on ARC-AGI 1 exhibits a monotonic decline.

Table 4: ARC-AGI 1 results after replacing ConvSwiGLU with weaker feed-forward nonlinearities.

Model	pass@1	pass@10	pass@100
ConvSwiGLU	**57.50**	**76.75**	**84.50**
SwiGLU	45.25	62.63	78.25
SiLU	29.75	42.13	54.50
ReLU	28.63	43.38	53.29
LinearGLU	24.22	36.50	47.05
LinearLU	8.25	17.00	21.81

Table 5 further clarifies that the *location* of nonlinearity is more important than the specific module in which it is implemented. Removing either the attention module or the feed-forward block in isolation leads to catastrophic failure, confirming nonlinear transformation are indispensable. However, when the feed-forward block is removed and its nonlinearity is instead relocated into the attention pathway—specifically between multi-head concatenation and the output projection—the model recovers almost the full performance of the baseline Transformer. PyTorch-like pseudocode for the activation variants and SiLU Attention, together with activation visualizations, is provided in Appendix B.

These results indicate that the expressive power required for ARC-AGI strongly depends on rich nonlinear mappings. Weakening the nonlinearity systematically limits the model's capacity to represent abstract reasoning abilities. The pretrained LLM results in Appendix B are consistent with the same interpretation: after large-scale language-model pretraining, adding URM-style iterative nonlinear computation still yields uniform gains on linguistic reasoning benchmarks, suggesting that stronger nonlinearity under recurrence remains beneficial beyond ARC-style tasks.

Table 5: Ablation study on Vanilla Transformer for ARC-AGI 1.

Model	pass@1	pass@10	pass@100
Vanilla	**23.75**	**34.13**	**38.88**
w/o MLP	0.00	0.38	0.75
w/o Attention	0.00	0.00	0.00
w/o MLP + SiLU Attention	22.35	33.52	38.43

5 Conclusion

We systematically investigate the sources of performance in Universal Transformers on abstract reasoning tasks. Ablations show that their gains arise primarily from recurrent inductive bias induced by parameter sharing and strong nonlinear transformations, rather than architectural complexity. Motivated by this insight, we propose the Universal Reasoning Model (URM), which strengthens nonlinear iterative computation through convolutional gating and truncated backpropagation through loops. Under controlled settings, URM achieves state-of-the-art performance among single small models on Sudoku, Maze, ARC-AGI 1 and 2.

We do not claim that abstract reasoning uniquely requires recurrence. Instead, abstract reasoning exemplifies a broader class of abstract reasoning tasks in which *iterative nonlinear computation* is essential, including algorithmic abstraction, symbolic rule induction, and compositional generalization. We expect these insights to generalize beyond abstract reasoning, and Appendix B provides initial evidence in this direction: under matched 300B-token pretraining with a Qwen3-0.6B architecture, URM also substantially improves all downstream reasoning performance across GSM8K, Big-Bench Hard, and ARC-Easy.

References

[1] Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=asgBo3FNdg.

[2] 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. URL https://openreview.net/forum?id=kxv0M6I7Ud.

[3] Bo Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. Bypassing the exponential dependency: Looped transformers efficiently learn in-context by multi-step gradient descent. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025. URL https://openreview.net/forum?id=4C1aRz2gRq.

[4] Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. Arc prize 2024: Technical report, 2025. URL https://arxiv.org/abs/2412.04604.

[5] Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-agi-2: A new challenge for frontier ai reasoning systems, 2025. URL https://arxiv.org/abs/2505.11831.

[6] 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.

[7] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2019. URL https://arxiv.org/abs/1807.03819.

[8] Amir Ben Dror, Niv Zehngut, Avraham Raviv, Evgeny Artyomov, Ran Vitek, and Roy Jevnisek. Layer folding: Neural network depth reduction using activation linearization, 2021. URL https://arxiv.org/abs/2106.09309.

[9] 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.

[10] Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2edigk8yoU.

[11] Khashayar Gatmiry, Nikunj Saunshi, Sashank Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[12] Renee Ge, Qianli Liao, and Tomaso Poggio. Hierarchical reasoning models: Perspectives and misconceptions, 2025. URL https://arxiv.org/abs/2510.00355.

[13] Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

[14] Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[15] Alex Graves. Adaptive computation time for recurrent neural networks, 2017. URL https://arxiv.org/abs/1603.08983.

[16] 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. URL https://arxiv.org/abs/2511.14761.

[17] Jianhao Huang, Zixuan Wang, and Jason D. Lee. Transformers learn to implement multi-step gradient descent with chain of thought. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r3DF5sOo5B.

[18] Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025. URL https://arxiv.org/abs/2510.04871.

[19] C. C. Jay Kuo. Understanding convolutional neural networks with a mathematical model, 2016. URL https://arxiv.org/abs/1609.04112.

[20] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

[21] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*, 2024.

[22] Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3082–3091. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/liao18c.html.

[23] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. 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. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/32cbf687880eb1674a07bf717761dd3a-Paper.pdf.

[24] 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. URL https://proceedings.mlr.press/v28/pascanu13.html.

[25] 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. URL https://openreview.net/forum?id=din0lGfZFd.

[26] Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

[27] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[28] Haoyuan Sun, Ali Jadbabaie, and Navid Azizan. On the role of transformer feed-forward layers in nonlinear in-context learning, 2025. URL https://arxiv.org/abs/2501.18187.

[29] Corentin Tallec and Yann Ollivier. Unbiasing truncated backpropagation through time, 2017. URL https://arxiv.org/abs/1705.08209.

[30] Corentin Tallec and Yann Ollivier. Unbiased online recurrent optimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJQDjk-0b.

[31] ARC Prize Team. The hidden drivers of hrm's performance on arc-agi, August 2025. URL https://arcprize.org/blog/hrm-analysis. Accessed: 2026-01-22.

[32] Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz 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. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[34] 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. URL https://arxiv.org/pdf/2405.15071.

[35] Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734.

[36] Bohong Wu, Mengzhao Chen, Xiang Luo, Shen Yan, Qifan Yu, Fan Xia, Tianqi Zhang, Hongrui Zhan, Zheng Zhong, Xun Zhou, Siyuan Qiao, and Xingyan Bin. Parallel loop transformer for efficient test-time computation scaling, 2025. URL https://arxiv.org/abs/2510.24824.

[37] 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. URL https://openreview.net/forum?id=HHbRxoDTxE.

[38] 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.

[39] Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. Investigating layer importance in large language models. In *The 7th BlackboxNLP Workshop*, 2024. URL https://openreview.net/forum?id=kjZIIvFtmK.

[40] Sumu Zhao, Damian Pascual, Gino Brunner, and Roger Wattenhofer. Of non-linearity and commutativity in bert. *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. URL https://api.semanticscholar.org/CorpusID:231583152.

[41] Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models, 2025. URL https://arxiv.org/abs/2510.25741.

Appendix

A Related Work

A ARC-AGI

Prior work on the ARC-AGI benchmark [4, 5] spans vision-based formulations, large language model (LLM) adaptation, and recurrent reasoning architectures. Vision-centric approaches such as Vision ARC [16] 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) [12, 35] 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) [18] further simplifies this paradigm, showing that a single lightweight network applied recursively can match or exceed more complex hierarchical designs. Test-time training (TTT), where the model performs a brief gradient update using demonstrations from each ARC task, has been shown to substantially outperform static prompting approaches [1].

B Universal Transformers (Looped Transformers)

The Universal Transformer (UT), also known as the Looped Transformer, was introduced by Dehghani et al. [7] 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 [6, 34]. In addition, UTs demonstrate improved algorithmic learning capabilities, enabling more effective modeling of iterative, rule-based, and program-like computations [14, 37]. Moreover, looped Transformers have been shown to exhibit strong length generalization, maintaining stable performance when extrapolating to significantly longer input sequences than those seen during training [10]. By reusing parameters across refinement steps, UTs also achieve higher parameter efficiency, allowing more expressive computation without increasing model size [25]. What's more, several recent works have trained Universal Transformers with tens of billions of parameters and demonstrated that, at comparable parameter scales, they significantly outperform vanilla Transformers, providing strong evidence for the scalability of Universal Transformers. [13, 20, 25, 36, 41] Related studies further suggest that recurrent refinement in looped Transformers can emulate in-context and iterative reasoning dynamics, enabling more structured computation across refinement steps [3, 11, 17].

B Pretrained Large Language Model Results

To evaluate whether the benefits of iterative nonlinear computation extend beyond grid-based abstract reasoning, we conducted additional experiments in a pretrained large language model setting. In contrast to the task-specific ARC-AGI, Sudoku, and Maze experiments, this setting evaluates URM under large-scale unsupervised language-model pretraining and downstream linguistic reasoning benchmarks.

Training setup. We use Megatron-LM [27] as the training framework. The backbone architecture follows Qwen3-8B [32], and all models are pretrained on a randomly sampled 300B-token subset of DCLM-Baseline [21]. The baseline model uses the standard Qwen3-8B architecture, while the URM variant incorporates the proposed iterative nonlinear computation mechanism into the same backbone setting. Both models are evaluated under the same downstream evaluation protocol.

Backbone architecture. Table 6 summarizes the main architectural configuration of the Qwen3-8B backbone used in our pretrained LLM experiments.

Table 6: Architecture details of the Qwen3-8B backbone used in the pretrained LLM experiments.

Configuration	Value
Number of parameters	8.2B
Number of non-embedding parameters	6.95B
Number of layers	36
Hidden size	4096
Attention type	Grouped-Query Attention (GQA)
Number of query attention heads	32
Number of key/value attention heads	8
Head dimension	128
MLP type	SwiGLU
Normalization	RMSNorm
RMSNorm epsilon	1×10^{-6}
Position encoding	RoPE
RoPE theta	1,000,000

Table 7: Training hyperparameters used in the pretrained LLM experiments.

Configuration	Value
Training framework	Megatron-LM
Learning rate	3×10^{-4}
Warmup steps	1500
Learning rate decay type	Constant
Global batch size	1024

Pretraining data. We use DCLM-Baseline as the pretraining data source. DCLM-Baseline is a 4T-token, 3B-document pretraining dataset that achieves strong performance on language model benchmarks. The dataset consists entirely of web text from Common Crawl. More specifically, it is a cleaned, deduplicated, and quality-filtered subset of DCLM-POOL. DCLM-POOL covers 5.1M Common Crawl WARC dumps from 2013–2022, with text extracted from HTML using `resiliparse`. In our experiments, we randomly sample 300B tokens from the full DCLM-Baseline dataset for pretraining.

Evaluation benchmarks. We evaluate the pretrained models on three language reasoning benchmarks: GSM8K, Big-Bench Hard (BBH), and ARC-Easy. These benchmarks test mathematical reasoning, diverse challenging reasoning skills, and science-question reasoning, respectively. This setting allows us to test whether the architectural advantages observed on grid-based abstract reasoning tasks also transfer to linguistic reasoning.

Table 8: Pretrained LLM results using the Qwen3-8B backbone trained on 300B randomly sampled tokens from DCLM-Baseline.

Model	**GSM8K**	**MATH500**	**Big-Bench Hard**	**ARC-Easy**	**ARC-Challenge**
Vanilla Qwen3	8.5	3.1	38.6	59.5	30.0
Qwen3 w/ 2 Loop	9.7	5.4	41.9	60.3	32.0
URM (2 Loop)	16.5	9.3	44.1	63.5	34.6

Results. As shown in Table 8, URM consistently improves over the vanilla Qwen3 and looped Qwen3 baseline across all five downstream reasoning benchmarks. URM improves GSM8K from 8.5 to 16.5, MATH500 from 3.1 to 9.3, Big-Bench Hard from 38.6 to 44.1, ARC-Easy from 59.5 to 63.5, and ARC-Challenge from 30.0 to 34.6.

Importantly, these results should be interpreted as evidence of transfer rather than as a claim that URM universally improves all language modeling settings. The pretrained LLM experiments show that the same design principle identified in our controlled abstract reasoning experiments—strengthening nonlinear computation inside iterative refinement—remains beneficial in a large-scale unsupervised pretraining setting.

C Algorithmic Pseudocode

Algorithm 1 ConvSwiGLU Feed-Forward Block

Input: $X \in \mathbb{R}^{B\times L\times d}$
Parameters:
$W_{gu} \in \mathbb{R}^{d\times 2d_h}$ (gate & up projection)
$K \in \mathbb{R}^{d_h\times 1\times k}$ (channel-wise convolution kernel)
$W_d \in \mathbb{R}^{d_h\times d}$ (down projection)
Output: $Y \in \mathbb{R}^{B\times L\times d}$
$(G, U) \leftarrow \text{Split}(XW_{gu})$ $G, U \in \mathbb{R}^{B\times L\times d_h}$
$H \leftarrow \text{SiLU}(G) \odot U$ (SwiGLU gating)
$\tilde{H} \leftarrow \text{DWConv1D}(H; K)$ (channel-wise 1D conv.)
$\hat{H} \leftarrow \text{SiLU}(\tilde{H})$
$Y \leftarrow \hat{H}W_d$
return Y

Algorithm 2 Universal Reasoning Model (URM)

Input: tokens x, total inner loops T, no-gradient inner loops N, parameters θ
Output: final hidden state H_T, loss $\mathcal{L}$
$H_0 \leftarrow h_{\text{init}}$
$E \leftarrow \text{Embed}(x)$
$r_{\text{halt}} \leftarrow 0$
$t \leftarrow 0$
while $t < T \wedge r < 1$ **do**
  $t \leftarrow t + 1$
  $H \leftarrow H + E$
  $H \leftarrow \text{RMSNorm}(H + \text{Attention}(H))$
  $H \leftarrow \text{RMSNorm}(H + \text{ConvSwiGLU}(H))$
  $p_{\text{halt}} \leftarrow \sigma(q_{\text{halt}}(H))$
  $r \leftarrow r + p_{\text{halt}}$
  if $t \leq N$ **then**
    NoGrad(H) (trunc. backprop. through loops)
  end if
end while
$H_T \leftarrow H$
$\mathcal{L} \leftarrow \text{LossHead}(H_T) + \lambda \cdot t$ (ACT penalty)
return $H_T,\ \mathcal{L}$

D Pytorch-like Pseudocode of the URM, Attention and FFN Used in Ablation Studies.

```
def URMBlock(hidden_states):
    hidden_states = RMSNorm(Attention(hidden_states) + hidden_states)
    hidden_states = RMSNorm(ConvSwiGLU(hidden_states) + hidden_states)

    return hidden_states

def URM_forward(hidden_states, batch):
    """
    hidden_states: [B, L, d]
    batch.inputs: token ids
    batch.puzzle_ids: optional identifiers
    """

    # === Input embedding ===
    embedding = Embed(batch.inputs)
    if use_puzzle_embedding:
        embedding = Concat(PuzzleEmbed(batch.puzzle_ids), embedding)
    embedding = scale * embedding

    # === Forward-only loops (no gradient) ===
    for _ in range(no_gradient_loops):
        with no_grad():
            hidden_states = hidden_states + embedding
            for URMBlock in URMBlocks:
                hidden_states = URMBlock(hidden_states)

    # === Main loops with gradient ===
    for _ in range(gradient_loops):
        hidden_states = hidden_states + embedding
        for URMBlock in URMBlocks:
            hidden_states = URMBlock(hidden_states)

    # === Output heads ===
    logits = LMHead(hidden_states)
    q_halt, q_continue = QHead(hidden_states)

    return hidden_states, logits, (q_halt, q_continue)
```

Figure 5: PyTorch-like pseudocode of the forward pass of the Universal Reasoning Model (URM). URM performs iterative nonlinear state transitions using shared Transformer blocks with ConvSwiGLU-enhanced feed-forward networks, followed by ACT-style halting.

```
def ConvSwiGLU(x):
    """
    x: [B, L, d]
    """
    # gated expansion
    gate_up = Linear(x)               # [B, L, 2d']
    gate, up = gate_up.chunk(2, dim=-1) # each [B, L, d']

    # gated nonlinearity
    h = SiLU(gate) * up               # [B, L, d']

    # channel mixing
    h = ChannelWiseConv1D(h)            # [B, L, d']
    h = SiLU(h)

    # projection back
    out = Linear(h)                   # [B, L, d]
    return out


def SwiGLU(x):
    gate_up = Linear(x)               # [B, L, 2d']
    gate, up = gate_up.chunk(2, dim=-1)
    h = SiLU(gate) * up
    out = Linear(h)
    return out


def LinearLU(x):
    h = Linear(x)                     # [B, L, d']
    out = Linear(h)                   # [B, L, d]
    return out


def LinearGLU(x):
    gate_up = Linear(x)               # [B, L, 2d']
    gate, up = gate_up.chunk(2, dim=-1)
    h = gate + up                     # linear gating
    out = Linear(h)
    return out


def SiLU(x):
    h = Linear(x)                     # [B, L, d']
    h = SiLU(h)
    out = Linear(h)
    return out
```

Figure 6: Torch-style pseudocode of feed-forward blocks evaluated in URM. All blocks share the same input/output dimensionality. ConvSwiGLU uniquely injects local token interactions via a channel-wise convolution after gated nonlinearity, amplifying iterative nonlinearity under recurrent application.

```
def SiLUAttention(hidden_states, cos_sin=None):
    """
    hidden_states: [B, L, d]
    """

    # === Linear projection to QKV ===
    qkv = Linear(hidden_states)
    # qkv: [B, L, (H + 2H_kv) * d_h]

    # === Split heads ===
    qkv = qkv.view(B, L, H + 2H_kv, d_h)
    Q = qkv[:, :, :H]              # [B, L, H, d_h]
    K = qkv[:, :, H : H + H_kv]    # [B, L, H_kv, d_h]
    V = qkv[:, :, H + H_kv :]      # [B, L, H_kv, d_h]

    # === Rotary positional embedding ===
    if cos_sin is not None:
        Q, K = RoPE(Q, K, cos_sin)

    # === Scaled dot-product attention ===
    A = Attention(Q, K, V, causal=causal)
    # A: [B, L, H, d_h]

    # === Merge heads ===
    A = A.view(B, L, H * d_h)

    # === Post-attention nonlinearity ===
    A = SiLU(A)

    # === Output projection ===
    out = Linear(A)                # [B, L, d]
    return out
```

Figure 7: Torch-style pseudocode of the SiLU-enhanced multi-head attention used in ablation studies.

E Activation Functions in Ablation Studies

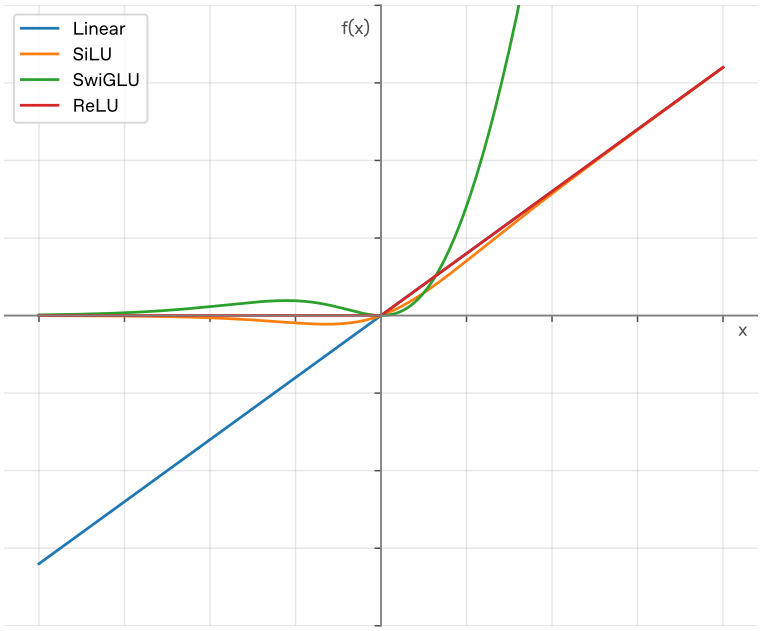

Figure 8: A visualization of the activation functions used in the feed-forward blocks for the ablation study. The visible nonlinearity progressively decreases from SwiGLU to Linear. Linear and ReLU overlap, as they coincide for $x > 0$.

F Convergence of the Number of ACT Loops During Training

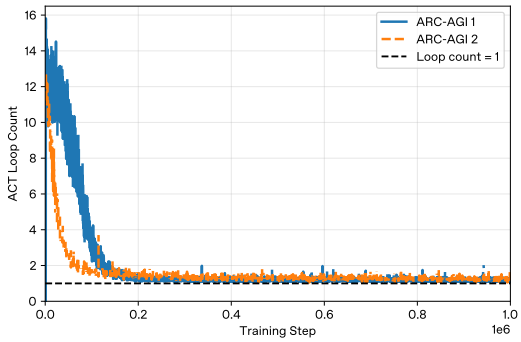

Figure 9: Convergence curve of the number of ACT loops during training. We report the training dynamics of URM on ARC-AGI 1 and ARC-AGI 2. In both settings, the ACT loop count converges to a value very close to 1 in the later stages of training, indicating that adaptive computation introduces little to no additional computational overhead.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: **[TODO]**

   Justification: **[TODO]**

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: **[TODO]**

    Justification: **[TODO]**

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.