LoopViT: Scaling Visual ARC with Looped Transformers

Wen-Jie Shu[1] **Xuerui Qiu**[2] **Rui-Jie Zhu**[3] **Harold Haodong Chen**[4]
Yexin Liu[4] **Harry Yang**[4]
[1]ZODA [2]CASIA [3]UC Santa Cruz [4]HKUST
wenjieshu2003@gmail.com

Abstract

Recent ARC-AGI systems have shown that pure vision can solve abstract grid reasoning without language backbones, but most strong models still rely on fixed-depth feed-forward computation. We study whether recurrent computation is a better scaling axis for this setting. We propose **Loop-ViT**, a vision-native recurrent model that reuses a weight-tied Hybrid Block across multiple steps, with local convolutions for image-token refinement and attention for global rule induction. We also use an entropy-based exit rule to reduce unnecessary iterations at inference time. On ARC-AGI-1, Loop-ViT improves the parameter-efficiency trade-off over fixed-depth vision baselines and reaches 65.8% accuracy with an 18M model. Overall, the results suggest that repeated computation is a useful inductive bias for ARC-style visual reasoning. The code is available at https://github.com/WenjieShu/LoopViT.

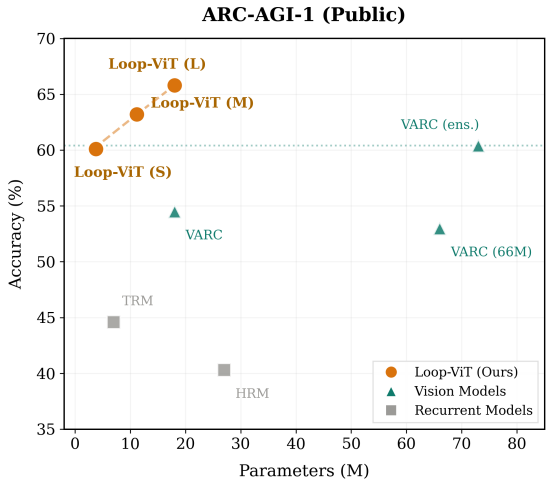


Figure 1: Accuracy and parameter trade-offs on ARC-AGI-1. Loop-ViT improves the parameter-efficiency curve relative to prior vision-based baselines.

1 Introduction

A core facet of ARC-style reasoning is not recognition alone, but the execution of a rule inferred from a handful of examples. ARC (Chollet, 2019; Chollet et al., 2024) operationalizes this setting through compact demonstration pairs and exact grid-level outputs, so the model must repeatedly revise local structure while keeping track of a global transformation. ARC-AGI combines compositional query-time reasoning with a small supervision budget, which makes success depend on step-wise state updates rather than static pattern matching. Unlike conventional vision benchmarks that reward dataset-scale statistical learning of textures or semantics (Deng et al., 2009; Lin et al., 2014; Cordts et al., 2016), ARC emphasizes procedural transformation over recognition.

Humans often solve these tasks through iterative hypothesis testing and revision (Johnson et al., 2021; LeGris et al., 2024), but current learning systems still struggle when they lack an explicit mechanism for multi-step deliberation (Pfister and Jud, 2025; Moskvichev et al., 2023). That gap matters because ARC is not just a visual classification problem: it asks the model to infer a latent program and then execute it consistently under tight sample constraints.

The need for iteration is not new. Prior recurrent ARC systems such as TRM and HRM already showed that iterative computation can help with abstract reasoning (Wang et al., 2025; Jolicoeur-Martineau, 2025). Their value is conceptual as much as empirical: they show that ARC benefits from multi-step inference. At the same time, those models operate on tokenized or serialized inputs, where spatial structure is flattened before reasoning begins. In contrast, the VARC line of work restored a purely visual formulation for ARC (Hu et al., 2025), but most strong vision solvers still use fixed-depth feed-forward computation. This leaves a gap between *iteration* and *spatial representation*: can recurrence be made

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 21271–21280
October 24-29, 2026 ©2026 Association for Computational Linguistics

vision-native without losing the visual structure that ARC depends on? This question is especially relevant because many successful ARC pipelines still rely on serialization, search, or test-time adaptation, which makes it hard to tell whether gains come from the representation, the computation, or both (Wang et al., 2024; Berman, 2024b; Tang et al., 2024; Berman, 2024a; Li et al., 2025; Berman, 2025; Macfarlane and Bonnet, 2024). In that sense, our goal is not to replace those systems wholesale, but to isolate what recurrence contributes once the input is kept fully visual.

We study this gap with **Loop-ViT**, a recurrent Vision Transformer that reuses a weight-tied Hybrid Block across multiple iterations. The block combines depth-wise convolution and self-attention so that each step can refine local structure while preserving global context. In this design, recurrence is not an add-on; it is the update rule the model repeatedly applies over a visual state. We also use an entropy-based exit rule at inference time so that easier instances stop early and harder ones continue for more steps. We treat this as a practical compute-control mechanism, not the main contribution.

Empirically, Loop-ViT improves the parameter-efficiency trade-off on ARC-AGI-1, as shown in Figure 1. A 3.8M-parameter model exceeds the 18M VARC baseline, and the 18M variant reaches 65.8% accuracy. The qualitative analyses in Sec. 4 further show that predictions stabilize over iterations and that the exit rule correlates with task difficulty. Taken together, these results suggest that recurrence remains a useful scaling axis for pure visual ARC, provided it is paired with a spatially grounded representation.

Our contributions are summarized as follows:

- Vision-native recurrence for ARC: We study ARC-style solving as recurrent state refinement over spatial tokens, while keeping the input representation fully visual.
- Spatially grounded recurrent core: We instantiate this formulation with a weight-tied **Hybrid Block** that combines local convolution for image-token updates and attention for global rule induction.
- Time-vs-space scaling evidence: We show increasing recurrent computation improves the parameter-efficiency trade-off on ARC-AGI-1 relative to fixed-depth vision baselines, and we provide lightweight behavioral analysis of the recurrent refinement process.

2 Related Work

We situate Loop-ViT in three lines of work: ARC reasoning with recurrent models, pure-vision ARC systems, and adaptive computation. Figure 2 summarizes the resulting design space.

ARC reasoning with recurrence. Recurrent computation is not new to ARC. Prior models such as TRM and HRM showed that iterative state updates can be effective for abstract reasoning (Wang et al., 2025; Jolicoeur-Martineau, 2025). These methods are important because they demonstrate that ARC benefits from multi-step inference rather than one-shot prediction. However, they typically operate on tokenized or serialized representations, where the grid structure is flattened before reasoning begins. Loop-ViT takes the same lesson, but asks whether recurrence can operate directly on a vision-native spatial state.

Pure-vision ARC models. The VARC framework (Hu et al., 2025) moved ARC back into the image domain by treating it as a vision problem rather than a text generation problem. This is a shift, because the task is fundamentally grid-based and depends on spatial relationships that are awkward to preserve in serialized form. Most strong vision systems in this line, however, use fixed-depth feed-forward computation. Loop-ViT builds on VARC's visual front end, but replaces the static computation graph with a weight-tied recurrent update rule applied to the same spatial state.

Adaptive computation. Recurrent models naturally decouple parameters from computation, which makes them a good fit for instance-dependent reasoning time. This broader idea also appears in weight-tied and iterative architectures that reuse computation across depth. Classical soft-halting methods such as ACT and PonderNet (Graves, 2017; Banino et al., 2021) learn when to stop, while more recent hard-halting or depth-allocation methods adjust computation with learned policies (Xu et al., 2023; Raposo et al., 2024). Related work has also explored entropy-based objectives and latent-step allocation for reasoning (Zhu et al., 2025; Chen et al., 2025; Bae et al., 2025). In Loop-ViT, we use a simple entropy-based exit at inference time as a practical heuristic: it is not the main novelty, but a way to let easier tasks stop early while preserving more

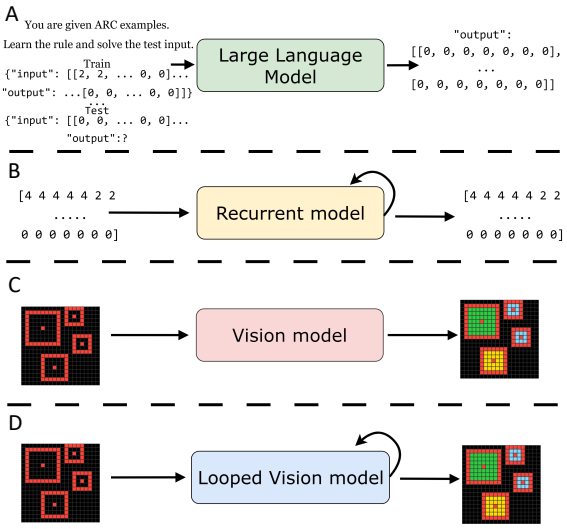


Figure 2: **Comparison of input representations and inference paradigms for ARC.** (A) LLMs operate on a 1D textual token sequence obtained by serializing the ARC grids into a prompt (*e.g.*, JSON/ASCII). (B) Recurrent token models also take a 1D sequence, but with a discrete grid-tokenization that pads the grid to a fixed canvas and inserts special boundary tokens (*e.g.*, PAD/EOS), yielding a fixed-length token stream. (C) VARC follows a vision formulation, encoding the grid as a 2D spatial input processed in a single forward pass. (D) Ours combines the vision input with looped/iterative inference, repeatedly refining internal repre- sentations and predictions across multiple steps, bridging spatial inductive bias and recurrent computation.

computation for difficult cases.

3 Method

Figure 3 illustrates the overall architecture of **Loop-ViT**. As depicted in Figure 3 (B), our model introduces iterative state refinement through a weight-tied core. We first formalize the recurrent update rule (Sec. §3.1). We then detail the *Hybrid Encoder Block*, which integrates convolutional and attention mechanisms for heterogeneous processing of image and task tokens (Sec. §3.2). Finally, we describe a simple entropy-based exit rule for inference-time compute allocation (Sec. §3.3) and the training protocol (Sec. §3.4).

3.1 Loop-ViT Architecture

Let $\text{emb}(\cdot) : \mathbb{X} \to \mathbb{R}^{M\times d}$ be the input embedding function that maps a visual grid x and task-specific context c to a sequence of M tokens with hidden dimension d. Let $\mathcal{M}_\theta(\cdot) : \mathbb{R}^{M\times d} \to \mathbb{R}^{M\times d}$ be a core transformer trunk parameterized by θ. As illustrated in Figure 3 (A), a standard non-looped vision transformer stacks L distinct layers such that the total function $\mathcal{F}$ is:

$$\mathcal{F}(\cdot) := \text{head} \circ \mathcal{L}_{\theta_L} \circ \cdots \circ \mathcal{L}_{\theta_1} \circ \text{emb}(\cdot), \quad (1)$$

where $\text{head}(\cdot)$ is the output projection layer. In contrast, our proposed **Loop-ViT** reuses the same core trunk $\mathcal{M}_\theta$ for T iterations. Let $t \in \{1, \ldots, T_{\max}\}$ be the number of loop steps. As shown in the unrolled view of Figure 3 (B), the state z_t evolves through the recursive application of the transition operator:

$$z_{t+1} = \mathcal{M}_\theta(z_t + e_t), \quad z_0 = \text{emb}(\cdot), \quad (2)$$

where e_t is a learned step-dependent embedding that disambiguates the computation progress. The final output is given by $\mathcal{F}^{(t)}(\cdot) = \text{head}(z_t)$. This formulation forces the model to learn a unified, step-wise transition rule that can be applied repeatedly. Crucially, scaling the computational duration T does not increase the parameter count, which makes recurrence a natural way to trade computation for capacity. For inference iterations exceeding the training budget, we adopt an identity extrapolation for the step embeddings: $e_t = e_{T_{\text{train}}}$ for all $t > T_{\text{train}}$.

3.2 Hybrid Encoder Block

We hypothesize that ARC tasks require two distinct modes of processing: *local pattern matching* (e.g., continuing a line or filling a region) and *global rule induction* (e.g., detecting symmetry or gravity). To support this, our **Hybrid Block** explicitly fuses the strengths of convolutions and attention. The depth-wise convolution in the FFN acts as a cellular automaton update rule, processing local neighborhoods to maintain spatial consistency. Simultaneously, the global attention mechanism broadcasts rule information across the entire grid, enabling long-range reasoning.

The core trunk $\mathcal{M}_\theta$ consists of L hybrid encoder layers. Each layer balances global relational reasoning with local spatial updates.

Internal Layer Structure. As shown in Figure 3 (C), each layer consists of Multi-Head Self-Attention (MHSA) followed by a Heterogeneous ConvGLU as the Feed-Forward Network (FFN). To better capture spatial relationships in ARC grids, the MHSA employs Rotary Positional Embeddings (RoPE) (Su et al., 2024). Given an input sequence $Z \in \mathbb{R}^{M\times d}$, we first project it into query, key, and value manifolds for each head h:

$$Q_h, K_h, V_h = ZW_h^Q, ZW_h^K, ZW_h^V. \quad (3)$$

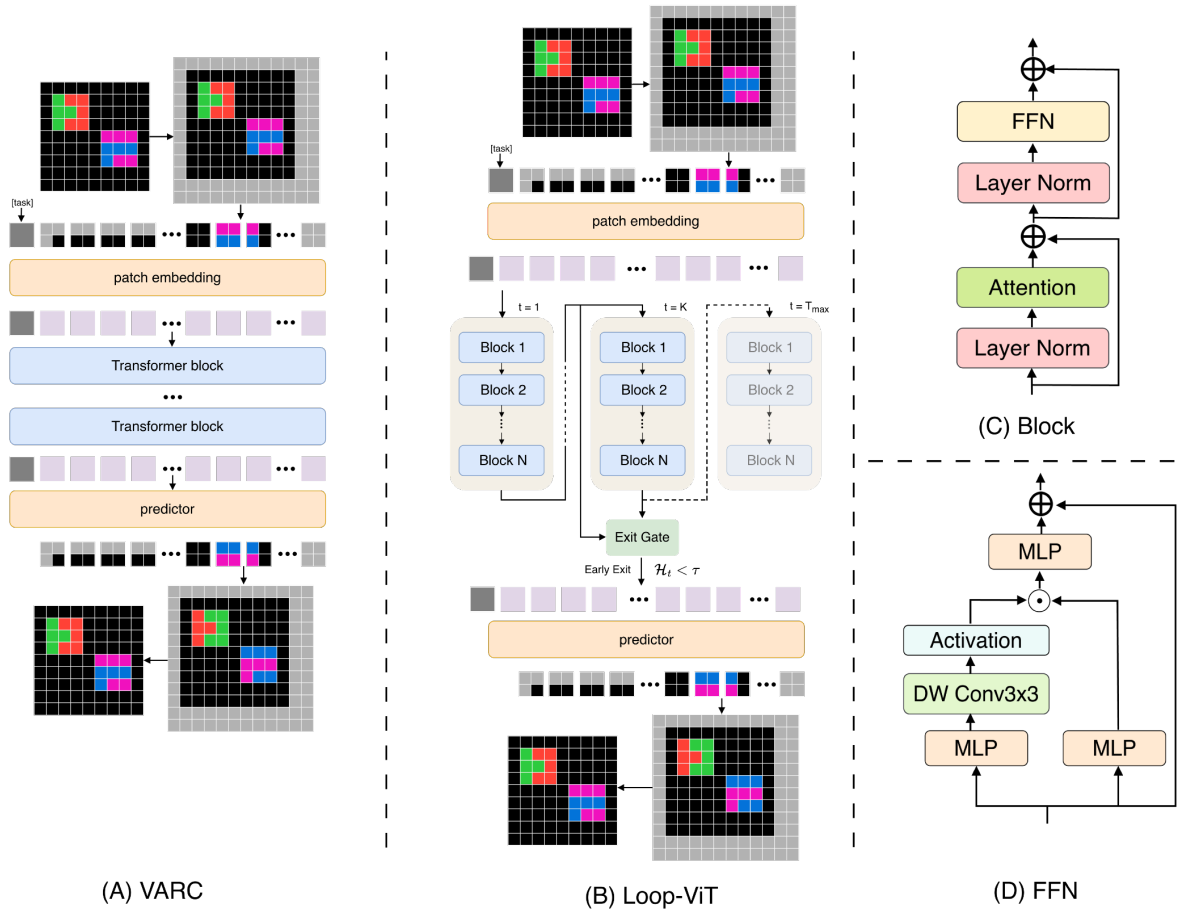


Figure 3: **The overall pipeline of the proposed LoopViT.** (A) Comparison of the standard VARC pipeline versus our Loop-ViT pipeline. Loop-ViT introduces iterative state refinement through a weight-tied core. (B) Detailed unrolled view of the Loop-ViT recurrence, where the state z_t acts as a dynamic memory. (C) Structure of the Hybrid Transformer Block, employing RMSNorm and Rotary Positional Embeddings. (D) The Heterogeneous Feed-Forward Network (ConvGLU), which splits processing pathways to apply depth-wise convolution solely to image tokens while preserving task tokens, reconciling local spatial updates with global rule induction.

The RoPE operator f_{R} is applied to Q_h and K_h to inject relative positional information. The output of a single head O_h is then computed as:

$$O_h = \mathrm{Softmax}\left(\frac{f_{\mathrm{R}}(Q_h) f_{\mathrm{R}}(K_h)^T}{\sqrt{d_h}}\right) V_h, \quad (4)$$

where W_h^Q, W_h^K, W_h^V are learnable projections and d_h is the head dimension. The final MHSA output integrates all heads via concatenation and a linear projection W^O:

$$\mathrm{MHSA}(Z) = \mathrm{Concatenate}(O_1, \ldots, O_H) W^O. \quad (5)$$

To ensure numerical stability during deep recurrence, we adopt a pre-norm configuration with RMSNorm (Zhang and Sennrich, 2019). The full layer transition is defined as:

$$Z' = Z + \mathrm{MHSA}(\mathrm{RMSNorm}(Z)) \quad (6)$$

$$Z_{\mathrm{out}} = Z' + \mathrm{ConvGLU}(\mathrm{RMSNorm}(Z')). \quad (7)$$

Heterogeneous Processing. The primary engine of our visual induction is the Heterogeneous ConvGLU, as illustrated in Figure 3 (D). Recognizing that task-level context tokens and spatial image patches require distinct inductive biases, we apply depthwise convolutions selectively. For a sequence Z, we first compute the gated and value representations:

$$[X_{\mathrm{gate}}, X_{\mathrm{val}}] = \mathrm{Linear}_1(Z). \quad (8)$$

We then partition X_{gate} into task tokens G_{task} and image tokens G_{img}. While G_{task} bypasses the spatial operator to preserve abstract rules, we reshape

G_{img} to a 2D grid and apply a 3×3 depthwise convolution (DW-Conv) to capture local connectivity:

$$\hat{G}_{\text{img}} = \text{Flatten}(\text{DW-Conv}(\text{Reshape}(G_{\text{img}}))). \tag{9}$$

The augmented gate $\hat{X}_{\text{gate}}$ is then reassembled via concatenation: $\hat{X}_{\text{gate}} = [G_{\text{task}}, \hat{G}_{\text{img}}]$. The final output is derived as:

$$\text{ConvGLU}(Z) = \text{Linear}_2(\sigma(\hat{X}_{\text{gate}}) \odot X_{\text{val}}), \tag{10}$$

where σ denotes the activation function (e.g., SiLU). This dual-track prior facilitates the decomposition of visual reasoning: MHSA facilitates global task induction, while ConvGLU executes local spatial transformations.

3.3 Dynamic Exit via Entropy-Based Stopping

A recurrent model naturally permits instance-dependent computation at inference time. We therefore use a simple entropy-based exit rule as a practical compute-control heuristic, motivated by the variable complexity of ARC tasks: simple geometric transformations may stabilize in few iterations, while complex algorithmic puzzles can require prolonged refinement.

The inference-time **Dynamic Exit** mechanism uses the magnitude of predictive entropy. Let $P_t = \text{softmax}(\text{head}(z_t))$ be the predicted probability distribution over the grid categories at step t. We quantify the model's confidence through the average pixel-wise Shannon entropy:

$$\mathcal{H}_t = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} P_{t,i}(c) \log P_{t,i}(c), \tag{11}$$

where N is the number of pixels and C is the number of color categories. During inference, generation halts at step t if the entropy falls below a confidence threshold $\mathcal{H}_t < \tau$, where $\tau = 0.05$. If the threshold is not met, computation continues until reaching a hard limit of $T_{\max}$ iterations. Once halted, the state is frozen ($z_k = z_t, \forall k > t$), effectively bypassing further core layer computations. We treat this entropy-based strategy as a practical compute-saving heuristic that uses prediction stabilization as a proxy for convergence.

3.4 Training Strategy

We employ a training protocol to learn a recurrent transition rule under a fixed unroll budget.

Fixed-Depth Training. The model is trained on the ARC and RE-ARC datasets. Crucially, we do *not* use dynamic halting during training. Instead, we unroll the core $\mathcal{M}_\theta$ for a fixed number of steps T (e.g., $T = 12$) and apply supervision on the final output. This procedure ensures that the model learns a robust transition rule $\mathcal{M}_\theta$ that converges within the allocated budget, rather than overfitting to early exits. The objective is a standard per-pixel cross-entropy loss:

$$\mathcal{L}_{\text{offline}} = \text{CrossEntropy}(P_T, Y_{\text{gt}}). \tag{12}$$

Test-Time Training (TTT). During evaluation, we fine-tune the shared weights θ on the few-shot demonstrations of each task. We generate augmented views (rotations, flips, color permutations) of the support examples to create a task-specific batch. This adaptation phase specializes the shared recurrent operator to the puzzle.

4 Experiments

This section evaluates **Loop-ViT** on ARC-AGI. Our goal is to study whether recurrent computation provides a better parameter-efficiency trade-off than fixed-depth visual models. The evaluation is organized around three questions: (***i***) how Loop-ViT compares against LLMs and vision baselines; (***ii***) how performance changes with recurrent depth and core depth; and (***iii***) how the internal predictions evolve across steps.

4.1 Experimental Setup

Datasets and Benchmarks. Primary evaluation is conducted on the ARC-AGI-1 benchmark (Chollet, 2019). Following state-of-the-art pixel-based methodologies (Hu et al., 2025), we augment the training split with synthetic samples from the RE-ARC generator (Hodel, 2024). We report **Pass@2 accuracy** in percentage (%). Results on ARC-AGI-2 are also provided to assess out-of-distribution generalization.

Implementation Framework. A two-stage training pipeline is adopted: (i) offline pre-training on augmented datasets; (ii) Test-Time Training (TTT) (Sun et al., 2020) on few-shot demonstrations. During TTT, the model specializes its weights to the specific task through local augmentations (*e.g.*, rotations and flips). To isolate the effect of recurrence, Loop-ViT follows the VARC visual front end and uses the same

Table 1: Performance comparison on ARC-AGI benchmark. **Loop-ViT** demonstrates superior parameter efficiency, with a modest 11.2M parameters outperforming a 73M-parameter ensemble. Best results are **bold**, the second results are underlined.

Model	#Params	ARC-AGI-1	ARC-AGI-2
Large Language Models (LLMs)			
Deepseek R1 (Guo et al., 2025)	671B	15.8	1.3
Claude 3.7 8k (Foundation, 2025)	N/A	21.2	0.9
o3-mini-high (Foundation, 2025)	N/A	34.5	3.0
GPT-5 (Foundation, 2025)	N/A	44.0	1.9
Grok-4-thinking (Foundation, 2025)	1.7T	66.7	16.0
Bespoke (Grok-4) (Berman, 2025)	1.7T	79.6	29.4
Recurrent Models			
HRM (Wang et al., 2025)	27M	40.3	5.0
TRM (Jolicoeur-Martineau, 2025)	7M	44.6	7.8
Vision Models			
VARC (Hu et al., 2025)	18M	54.5	8.3
VARC (ensemble) (Hu et al., 2025)	73M	60.4	11.1
Loop-ViT (Small)	**3.8M**	60.1	10.0
Loop-ViT (Medium)	**11.2M**	63.8	11.5
Loop-ViT (Large)	**18M**	**65.8**	**14.2**
human results			
avg.human (LeGris et al., 2024)	–	60.2	–
best.human	–	98.0	100.0

ARC/RE-ARC augmentation and TTT protocol as the corresponding vision baselines.

Model Configurations. We analyze three variants designated as Small (3.8M params), Medium (11.2M params), and Large (18M params). We set $T_{\max} \in [20, 28]$ for the Small variant and $T_{\max} \in [4, 8]$ for Medium/Large, depending on model scale and task complexity.

4.2 Main Results: Performance and Parameter Efficiency

Table 1 summarizes the performance of Loop-ViT. The main observation is that a recurrent visual model can be competitive with larger fixed-depth vision baselines and many large language model systems on ARC-AGI.

Loop-ViT shows a clear benefit from recurrence. By recycling weights across iterations, Loop-ViT (Large) achieves 65.8% on ARC-1, surpassing the 73M VARC ensemble. This suggests that scaling iterative computation can be a useful alternative to increasing raw layer count.

4.3 Analysis of Recurrent Refinement

The results above are best understood as evidence for iterative refinement rather than one-shot recognition. We therefore use the remaining diagnostics to connect the performance gains with the recurrent design, without claiming a full theory.

Prediction Stabilization. In Figure 4 and Figure 8, prediction changes become smaller across iterations while entropy decays in parallel. This means later steps typically make smaller corrections than earlier ones, which is consistent with the model gradually sharpening an initially coarse visual state.

Exit as a Difficulty Signal. The dynamic-exit plots in Figure 7 and Figure 9 suggest that the entropy rule is not just a convenience for saving compute. Easier cases tend to stop earlier, while harder cases require more steps, so the exit decision tracks task difficulty in a way that is qualitatively aligned with the benchmark.

What We Do Not Claim. These observations do not prove stable attractors, latent chain-of-thought behavior, or a complete explanation of the solver. What they do show is more modest: the recurrent core behaves like a refinement process, and the adaptive halting rule follows that refinement in a sensible way.

4.4 Ablation Study

To evaluate the effectiveness of our design choices, we conduct a series of ablation experiments. We focus on two critical axes: (i) the structural trade-offs between parameter budget (space) and computational depth (time), and (ii) the necessity of spatial inductive biases in the recurrent core.

Space-Time Joint Scaling. Figure 5 explores the trade-off between core block depth (B) and number of unrolled loop steps (T). The diverging trajectories reveal two regimes: (i) For low-capacity cores ($B = 2$), increasing T yields the largest gains as weight-tied recurrence emulates deeper computation; (ii) For high-capacity cores ($B = 10$), performance continues to scale with T up to the computational limit, reaching 63.9%.

Inductive Bias: Hybrid vs. Vanilla. The Hybrid Block (DW-Conv + MHSA) is compared against a standard Vanilla Transformer across core depths. As shown in Figure 6, the Hybrid architecture maintains a consistent lead. This suggests that local spatial priors are a fundamental requirement for grounding abstract reasoning in grid-based visual domains, regardless of model depth.

Impact of Dynamic Exit. The effectiveness of adaptive halting is evaluated by comparing our dynamic-step model (constrained to $T \in [4, 8]$)

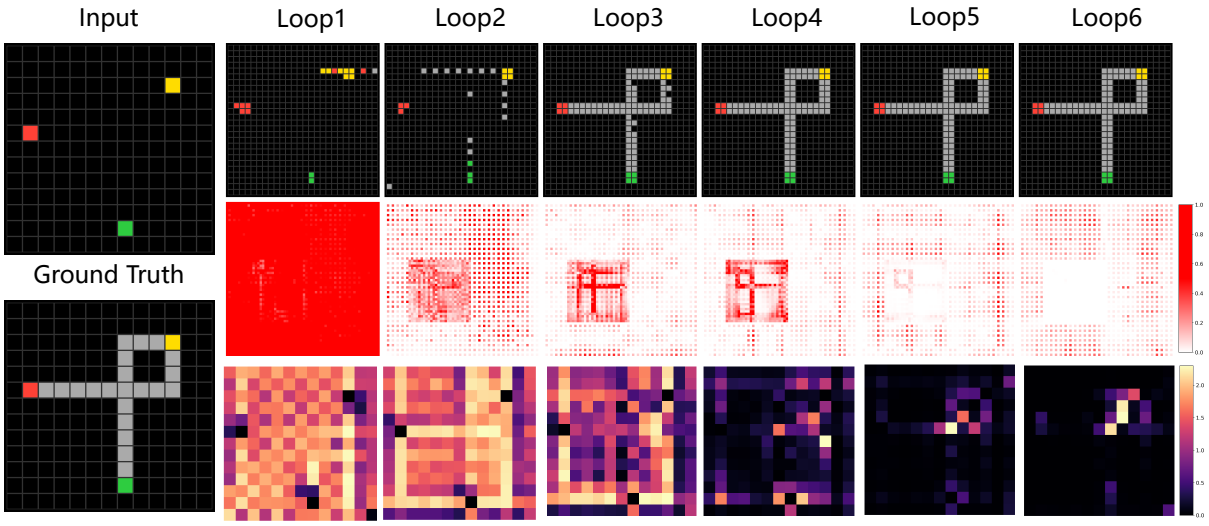


Figure 4: **Iterative Prediction Refinement in Loop-ViT.** (Top) The model's output progressively approaches the ground truth through successive iterations. (Middle) Pixel-wise difference maps between consecutive steps show decreasing prediction volatility. (Bottom) Entropy measurements demonstrate the stabilization of the model's confidence. This diagnostic view shows how recurrent processing enables gradual convergence over steps.

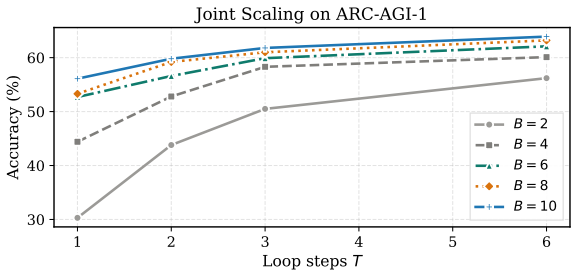


Figure 5: **Joint scaling of core block depth (B) and loop steps (T) on ARC-AGI-1.** We vary the number of layers in the recurrent core (B) and the number of unrolled iterations (T). Each line represents a fixed core depth. The effect of recurrence is most visible in lower-capacity core models (*e.g.*, $B = 2$), where increasing T from 1 to 6 yields a large performance gain. Performance continues to scale with T even for deeper cores, suggesting that computation time can compensate for limited parameter space.

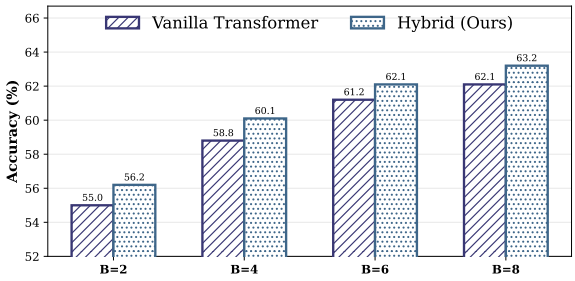


Figure 6: **Ablation of Inductive Bias: Hybrid vs. Vanilla Core** across different core block depths (B) on ARC-AGI-1. We compare our Hybrid architecture (incorporating depth-wise convolutions in the FFN) against a standard Vanilla Transformer core. The Hybrid core consistently maintains an accuracy gap over the Vanilla baseline across all depths. This indicates that local spatial priors are useful for grounding abstract reasoning in the image domain, and this benefit does not disappear as model depth increases.

against a fixed 6-step baseline. As shown in Figure 7, the entropy-based exit rule achieves higher accuracy while using lower *average* inference compute on ARC-AGI-1.

Efficiency vs. Difficulty. We further analyze the correlation between inference steps and task difficulty using the Loop-ViT variant with $B = 2$ (approximately 5M parameters) in Figure 9. Samples that exit early (*e.g.*, at Step 5) yield higher accuracy (83.33%), whereas those requiring more iterations (Step 8) are harder (45.80%). This indicates that the exit rule correlates with task difficulty.

4.5 Mechanistic Insights

Beyond quantitative performance, we conduct a qualitative analysis to understand the internal refinement process of Loop-ViT. By visualizing prediction changes and attention evolution, we aim to see how the recurrent state evolves across steps.

Prediction Evolution. As illustrated in Figure 4, Loop-ViT's predictions become progressively more stable across iterations. Quantitative

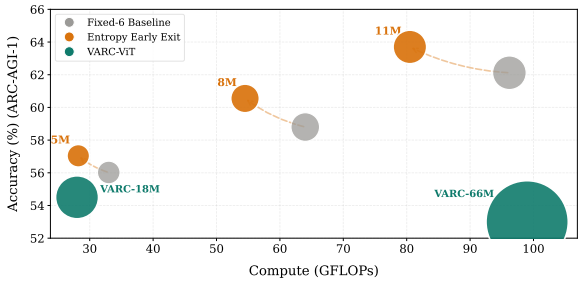


Figure 7: **Accuracy–Compute–Params comparisons.** The horizontal axis is total inference compute, the vertical axis is Accuracy, and the circle radius corresponds to model Parameters. For Loop-ViT, GFLOPs accounts for unrolled recurrence and is computed using the *average executed steps* under entropy-based halting. Our entropy-based exit strategy (orange) consistently surpasses the fixed-step baselines (grey) across all model scales (B=2, 4, 6), showing a better accuracy–compute Pareto frontier. Feed-forward VARC baselines are included for comparison.

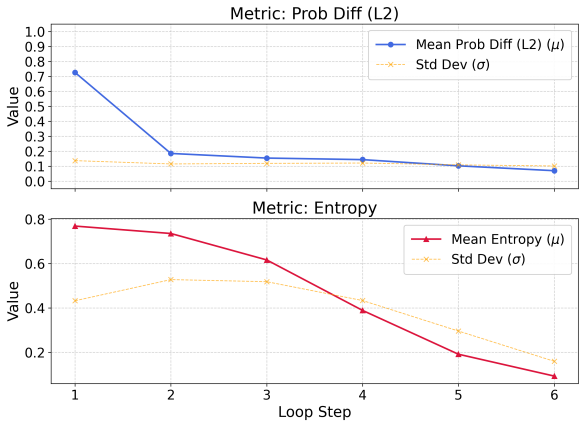


Figure 8: **Quantitative Diagnostics of Recurrent Convergence.** We monitor the evolution of (top) the L_2-normalized difference δ_t and (bottom) the average pixel-wise Shannon entropy $\mathcal{H}_t$ across Loop-ViT iterations. Solid lines and shaded regions represent the mean and variance across the validation set, respectively. The synchronized decay of both prediction volatility and information uncertainty suggests that the model's internal state becomes more stable over time.

analysis in Figure 8 shows a synchronized decay in both prediction volatility (L_2 difference) and uncertainty (entropy). The L_2 difference drops in early iterations, suggesting a rapid commitment to the task geometry. The reduction in mean entropy $\mathcal{H}_t$ indicates that the predictions become more concentrated over time.

Step-wise Attention Dynamics. We visualize the evolution of self-attention patterns across the loop in Figure 10. In earlier steps, attention matrices are relatively dense, reflecting broad context integration. As processing progresses, attention shifts toward more localized patterns focusing on the grid transitions required for the predicted rule.

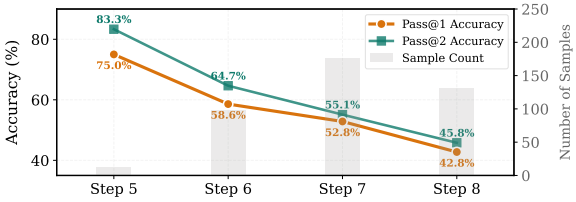


Figure 9: **Efficiency vs. Task Difficulty Analysis.** Using Loop-ViT variant ($B = 2$), we stratify the test set by the number of inference steps Loop-ViT requires before exiting. "Early Exit" (Step 5) samples achieve higher accuracy (83.33%) compared to those requiring full depth (Step 8, 45.80%). This suggests that the exit rule correlates with task difficulty and compute usage.

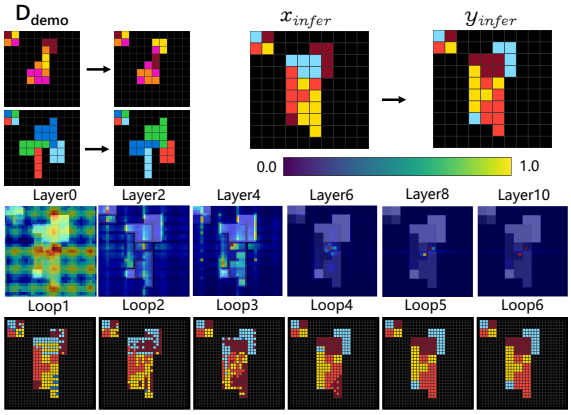


Figure 10: **Evolution of Attention Patterns Across Processing Steps.** We visualize the average self-attention maps across Loop-ViT's recurrent steps. Early steps exhibit broad attention that analyzes the full input context. Later steps develop focused, sparse patterns that precisely track the algorithmic operations needed to solve the ARC task. This shift from broad to localized attention is consistent with step-wise refinement.

5 Conclusion

We studied recurrent computation for visual, rather than serialized, ARC solving. Loop-ViT uses a weight-tied Hybrid Block that combines local convolution and global attention over spatial tokens. On ARC-AGI-1, it improves the parameter-efficiency trade-off over fixed-depth visual baselines, while entropy-based early exit provides inference-time compute control. These results support recurrence as a useful scaling axis for spatially grounded visual reasoning.

Limitations

Our evaluation focuses on ARC-AGI-1, which provides a controlled testbed for abstract visual reasoning but does not cover the full diversity of natural-language and multimodal reasoning tasks. Although we report ARC-AGI-2 results, a finer-grained analysis of out-of-distribution failures remains important. The entropy-based exit rule is a practical inference heuristic with a fixed threshold, and future work should study threshold sensitivity, learned halting, and alternative uncertainty signals. We also leave more extensive multi-seed reporting, RE-ARC leakage checks, and ablations of step embeddings and weight tying to future work.

Acknowledgments

We thank ZODA for providing computational resources and research support for this work.

References

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. 2025. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *Preprint*, arXiv:2507.10524.

Andrea Banino, Jan Balaguer, and Charles Blundell. 2021. Pondernet: Learning to ponder. *Preprint*, arXiv:2107.05407.

Jeremy Berman. 2024a. How I came in first on ARC-AGI-Pub using Sonnet 3.5 with evolutionary test-time compute. *Substack*. Accessed: 2025-10-13.

Jeremy Berman. 2024b. How I got a record 53.6% on ARC-AGI. *Substack*. Accessed: 2025-10-13.

Jeremy Berman. 2025. How I got the highest score on ARC-AGI again swapping Python for English. *Substack*.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. 2025. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. *arXiv preprint arXiv:2502.13842*.

François Chollet. 2019. On the measure of intelligence. *arXiv:1911.01547*.

Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. 2024. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

ARC Prize Foundation. 2025. ARC-AGI benchmarking: Leaderboard and dataset for the ARC-AGI benchmark. https://arcprize.org/leaderboard. Accessed: 2025-11-01.

Alex Graves. 2017. Adaptive computation time for recurrent neural networks. *Preprint*, arXiv:1603.08983.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081).

Michael Hodel. 2024. Addressing the abstraction and reasoning corpus via procedural example generation. *arXiv:2404.07353*.

Keya Hu, Ali Cy, Linlu Qiu, Xiaoman Delores Ding, Runqian Wang, Yeyin Eva Zhu, Jacob Andreas, and Kaiming He. 2025. Arc is a vision problem! *arXiv preprint arXiv:2511.14761*.

Aysja Johnson, Wai Keen Vong, Brenden M Lake, and Todd M Gureckis. 2021. Fast and flexible: Human program induction in abstract reasoning tasks. *arXiv preprint arXiv:2103.05823*.

Alexia Jolicoeur-Martineau. 2025. Less is more: Recursive reasoning with tiny networks. *arXiv:2510.04871*.

Solim LeGris, Wai Keen Vong, Brenden M. Lake, and Todd M. Gureckis. 2024. H-ARC: A robust estimate of human performance on the abstraction and reasoning corpus benchmark. *arXiv:2409.01374*.

Wen-Ding Li, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M. Dunn, Hao Tang, Wei-Long Zheng, Yewen Pu, and Kevin Ellis. 2025. Combining induction and transduction for abstract reasoning. In *ICLR*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Matthew V Macfarlane and Clément Bonnet. 2024. Searching latent program spaces. *arXiv:2411.08706*.

Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. 2023. The ConceptARC benchmark: Evaluating understanding and generalization in the ARC domain. *arXiv:2305.07141*.

Rolf Pfister and Hansueli Jud. 2025. Understanding and benchmarking artificial intelligence: Openai's o3 is not agi. *arXiv preprint arXiv:2501.07458*.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, PC Conway, and P Adam. 2024. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*.

Hao Tang, Keya Hu, Jin Zhou, Sicheng Zhong, Wei-Long Zheng, Xujie Si, and Kevin Ellis. 2024. Code repair with LLMs gives an exploration-exploitation tradeoff. In *NeurIPS*.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. 2025. Hierarchical reasoning model. *arXiv:2506.21734*.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah D. Goodman. 2024. Hypothesis search: Inductive reasoning with language models. In *ICLR*.

Guanyu Xu, Jiawei Hao, Li Shen, Han Hu, Yong Luo, Hui Lin, and Jialie Shen. 2023. LGViT: Dynamic early exiting for accelerating vision transformer. In *ACM MM*.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Preprint*, arXiv:1910.07467.

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, and 14 others. 2025. Scaling latent reasoning via looped language models. *Preprint*, arXiv:2510.25741.