Designing Training Objectives for Iterative Reasoning Agents: Dense Supervision as an Adaptive Mechanism

author names withheld

Under Review for NExT-Game 2026

Abstract

Modern AI agents increasingly rely on iterative computation – chain-of-thought, looped Transformers, and recurrent reasoners – to "think longer" before acting. The training objective these systems are optimized under is itself a design choice that shapes the agent's downstream behavior: which intermediate states are reachable, whether reasoning extrapolates beyond training-time horizons, and whether the agent can halt adaptively. The standard recipe – supervising only the final iterate (*endpoint supervision*) – treats this design problem implicitly and, we argue, badly. Viewed as a mechanism that the training procedure imposes on the learner's internal dynamics, endpoint supervision induces two coupled pathologies: (i) gradient signals at early reasoning steps are noisy in magnitude and unreliable in direction, and (ii) intermediate states are left unconstrained, so the agent learns a fixed-horizon strategy that breaks when more reasoning is permitted at test time. We introduce Dense Intermediate Consistency for Endpoints (DICE), a training-time mechanism that attaches a *shared* readout head to every iterate and applies auxiliary losses across the full trajectory. The shared readout enforces *anytime decodability*: every intermediate state is a valid action under the agent's output interface, making variable-depth reasoning well-defined rather than emergent. Across three representative iterative architectures, DICE yields consistent gains – near-perfect algorithmic extrapolation on prefix sums, $+7.4$ pp exact-match accuracy on maze solving, and a $6.6\times$ inference-time speedup via adaptive halting.

1. Introduction

Agentic AI systems are increasingly built around *iterative* reasoning. Chain-of-thought prompting, looped Transformers [11, 12, 34], Universal Transformers [8], Deep Equilibrium Models [2], Deep Thinking networks [26], and recursive reasoners [14] all share a computational pattern: a shared update function is applied repeatedly to refine an internal state before the agent commits to an output. We unify these under *Iterative Neural Computations* (INCs). In agentic settings, this inner loop is what lets an agent allocate more compute to harder problems and – in principle – behave robustly when its environment demands more deliberation than was anticipated at training time.

The training objective an INC is optimized under is therefore not a neutral implementation detail. It is a *mechanism*: it determines which trajectories of internal computation are rewarded, which intermediate states are reachable, and how the agent's behavior degrades or improves when the inference-time compute budget changes. Yet across the architectures above, the dominant recipe is the same – **a loss is computed only at the final iterate, and gradients are propagated back through the entire chain of updates**. We call this *endpoint supervision*, and we argue it is a poor mechanism for the design problem at hand.

© .

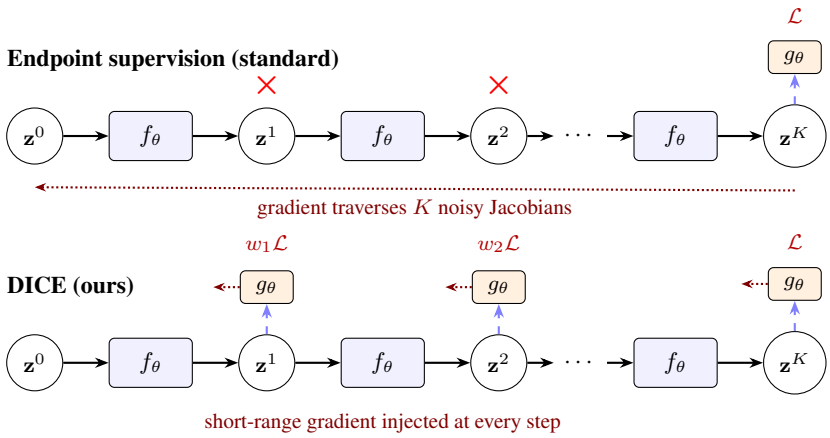

Figure 1: **The two problems and how dense supervision solves them.** Top: endpoint supervision provides a single gradient that must traverse K Jacobians (**Problem 1**) and leaves intermediate states unsupervised (**Problem 2**, denoted as $\times$). Bottom: DICE adds a *shared* readout g_θ at every iterate. Auxiliary losses inject direct gradients (**Solution 1**), and the shared head forces every iterate to be decodable (**Solution 2**).

INCs trained under this recipe consistently struggle to generalize on tasks that require long trajectories of computation [6, 10, 15]. These failures have been documented but typically treated as architecture-specific bugs to patch [2, 4, 11] or engineer around [5, 26]. We argue the shared cause is the supervision mechanism itself, which induces two coupled failure modes: (i) the gradient signal that reaches early iterations is noisy and unreliable, because it must traverse the full chain of updates; and (ii) intermediate iterations are left entirely unconstrained, so the model has no incentive to behave well beyond the trained iteration count. The second mode is particularly damaging for agents: it means an agent given more inference-time compute has no guarantee of producing a meaningful intermediate action, undermining the very property – variable-depth deliberation – that motivates iterative architectures.

We propose a simple, model-agnostic training mechanism, **Dense Intermediate Consistency for Endpoints (DICE)**. DICE adds auxiliary losses at intermediate iterations, decoded through a readout head that is *shared* with the endpoint. The auxiliary losses shorten the gradient paths reaching early steps, while the shared readout head enforces *anytime decodability*: every intermediate iterate must be a valid prediction under the agent's output interface. The change is architecture-agnostic and adds less than 5% to training compute.

Contributions. (i) We frame the training objective of an INC as a mechanism that shapes the agent's reasoning trajectory, and identify two coupled failure modes of endpoint supervision. (ii) We propose DICE, a drop-in training mechanism enforcing anytime decodability via a shared readout and short-range gradients via dense auxiliary losses. (iii) Empirically, DICE yields near-perfect extrapolation on prefix sums, $+7.4$ pp on 30×30 maze solving, and $+3.97$ pp on bAbI with a Universal Transformer. (iv) A four-way ablation and direct gradient diagnostics decompose the gains into gradient shortening and shared-representation constraint (Sec. E and F). (v) Anytime decodability enables a simple adaptive-halting rule giving a $6.6\times$ inference-time speedup.

2. Setup and Method

Iterative Neural Computations. Given input $\mathbf{x} \in \mathcal{X}$ and target $\mathbf{y} \in \mathcal{Y}$, an INC is specified by an **encoder** $h_\theta : \mathcal{X} \to \mathcal{Z}$ producing the initial state $\mathbf{z}^0 = h_\theta(\mathbf{x})$, an **update function** $f_\theta : \mathcal{Z} \to \mathcal{Z}$, and a **readout head** $g_\theta : \mathcal{Z} \to \mathcal{Y}$, all sharing parameters θ. The model unrolls f_θ for K steps, producing the trajectory $\mathbf{z}^k := f_\theta(\mathbf{z}^{k-1})$, and predicts $\hat{\mathbf{y}} = g_\theta(\mathbf{z}^K)$. Universal Transformers, Deep Thinking nets, DEQs, and recursive Transformers all instantiate this triple in different ways (see Sec. B). Our analysis depends only on the shared structure.

Endpoint supervision and its failure modes. Let ℓ be a per-example loss. The standard objective is the *endpoint loss*

$$\mathcal{L}_{\text{end}}(\theta; \mathbf{x}, \mathbf{y}) \;=\; \ell\big(g_\theta(\mathbf{z}^K),\, \mathbf{y}\big)\,. \tag{1}$$

Viewed as a mechanism, $\mathcal{L}_{\text{end}}$ provides outcome-only feedback: any trajectory producing a correct $\mathbf{z}^K$ at the trained depth is equally rewarded. This induces two failure modes. (1) *Gradient-signal degradation*: the chain-rule contribution from iteration k traverses the Jacobian product $\prod_{j=k}^{K-1} \partial f_\theta(\mathbf{z}^j)/\partial \mathbf{z}^j$, the same product that drives vanishing/exploding gradients in deep recurrent networks [22]. Even when magnitudes are controlled, its *direction* accumulates noise as $K - k$ grows [23], an effect that Sec. E measures directly. (2) *Unconstrained intermediates*: $\mathbf{z}^1, \ldots, \mathbf{z}^{K-1}$ are treated as latent scratch space, so they need not lie in a region of $\mathcal{Z}$ from which g_θ produces a meaningful output. This is fatal whenever the test-time iteration count K_{test} exceeds K_{train} – precisely the regime in which an iterative agent "thinks longer" [26].

The DICE objective. DICE addresses both modes with a single change to the loss:

$$\mathcal{L}_{\text{dense}} \;=\; \ell\big(g_\theta(\mathbf{z}^K),\, \mathbf{y}\big) \;+\; \alpha \sum_{k=1}^{K-1} w_k\, \ell\Big(g_\theta(\mathbf{z}^k),\, \mathbf{y}\Big)\,, \tag{2}$$

where $\alpha > 0$ is the auxiliary strength and $\{w_k\}_{k=1}^{K-1}$ a normalized weight schedule (uniform $w_k \propto 1$, linear $w_k \propto k$, or exponential $w_k \propto 2^k$). Crucially, g_θ is the *same* mapping at every iteration. The auxiliary loss at step k backpropagates through only k applications of f_θ, shortening the gradient path. The shared readout enforces anytime decodability: every iterate must decode to a valid prediction under one interface. No architectural changes are required, and the cost is $K-1$ additional readout evaluations per step ($< 5\%$ overhead).

Adaptive Stability Halting (ASH). Because every iterate is decodable, DICE-trained models behave as anytime predictors. Let $\hat{\mathbf{y}}_k = g_\theta(\mathbf{z}^k)$. We halt when the predictive distribution stabilizes:

$$\|\text{softmax}(\hat{\mathbf{y}}_k) - \text{softmax}(\hat{\mathbf{y}}_{k-1})\|_1 < \epsilon \quad \text{for } m \text{ consecutive steps.}$$

This approximates a fixed-point detector in prediction space rather than representation space. Sec. K shows the dense objective directly bounds the prediction-space drift used here.

3. Experiments

We evaluate DICE on three iterative systems, keeping architecture and training procedure unchanged in each case – only the loss differs. **Baselines.** *Endpoint*: standard endpoint supervision. *GRPO* [27]: group-normalized policy gradient at the terminal iterate. *LSRL* [24]: dense intermediate RL supervision with an intermediate verifier. *RLTT* [33]: outcome reward distributed across iterations via a weighted log-prob sum. All baselines use the same data per task. Implementation details in Sec. D.

Table 1: **Deep Thinking on prefix sums.** Test-time extrapolation accuracy (%) at 40 iterations.

Test bits	Endpoint	GRPO	LSRL	RLTT	DICE
64	11.70	20.10	34.25	40.08	**100.00**
72	0.12	14.64	22.09	25.19	**100.00**
128	0.00	1.49	0.56	5.33	**100.00**
256	0.00	0.00	0.03	0.00	**11.88**

Table 2: **ASH on prefix sums.** It is a pure compute saving.

Bits	Acc	Avg iters
64	100.00	15.53
72	100.00	23.37
128	100.00	47.27
256	99.95	71.97

Table 3: **Recursive Transformer on maze solving.** *Left:* Exact accuracy (%) at each evaluation epoch. DICE produces dramatically faster learning and higher final accuracy. *Right:* Comparison of training strategies on exact-match accuracy.

Configuration	Ep. 1k	Ep. 2k	Ep. 3k	Ep. 4k
Endpoint	0.0	27.1	59.7	73.6
DICE, w=0.1	3.8	36.7	69.6	74.7
DICE, w=0.3	0.7	68.8	66.4	73.3
DICE, w=1.0	0.7	**69.7**	**78.9**	**81.0**

(a) Per-epoch evaluation.

Method	Acc.	vs ep.
Endpoint	73.6	–
GRPO	74.3	+0.7 pp
LSRL	74.8	+1.2 pp
RLTT	75.0	+1.4 pp
DICE	**81.0**	**+7.4 pp**

(b) Comparison vs. baselines.

Deep Thinking on prefix sums (variable-depth extrapolation). Given a sequence of bits, the model must output the running XOR. Trained on 32-bit inputs with random-length unrolls, tested on 64-, 72-, 128-, and 256-bit inputs at 40 iterations [25]. This is a clean test of variable-depth reasoning. The endpoint baseline achieves only 11.70% on 64-bit inputs and degrades to near zero on longer inputs (Tab. 1). DICE with a linear schedule yields **100%** extrapolation accuracy on 64-, 72-, and 128-bit inputs and 11.88% on 256-bit inputs. The shared readout constrains each iterate to decode to a meaningful partial solution, forcing f_θ to implement a genuinely iterative refinement strategy so that additional test-time steps continue the computation rather than break a fixed-length recipe. ASH terminates well below the 80-iteration budget at no accuracy cost (Tab. 2).

Recursive Transformer on maze solving. We use the Tiny Recursive Model (TRM) [11] on 30×30 mazes with shortest-path length >110 [20], a benchmark for LLM-style search reasoning [7, 28]. DICE outperforms all baselines on exact accuracy (Tab. 3). Two effects are visible (Tab. 3). First, *convergence acceleration*: at epoch 2,000, DICE reaches 69.7% exact accuracy vs. 27.1% for the baseline, surpassing the baseline's 4,000-epoch best by epoch 3,000. Second, DICE's best checkpoint reaches **81.0%** exact accuracy, +7.42 pp over the baseline. ASH slightly *improves* accuracy over fixed-budget inference (81.05% vs. 80.96%) with an average outer-step count of 2.43, indicating DICE concentrates useful predictions into early refinement steps.

Universal Transformer on bAbI. On bAbI question-answering [32], the endpoint baseline achieves 89.72%; DICE achieves **93.69%** (+3.97 pp) and outperforms all baselines (Tab. 5). With ASH, DICE uses 50% fewer refinement steps.

Table 4: **Adaptive halting on TRM (maze, epoch 4k).** DICE improves training-time accuracy over the endpoint baseline; applying prediction-stability halting on the outer N_{sup} refinement loop then yields an additional $6.6\times$ inference-time speedup with no accuracy loss.

Configuration	Outer steps	Exact acc.	Avg. steps	Speedup
Endpoint	16 (fixed)	73.63%	16.00	$1.00\times$
DICE (w.o. adaptive halt)	16 (fixed)	80.96%	16.00	$1.00\times$
DICE (w. adaptive halt)	≤ 16	**81.05%**	**2.43**	$6.58\times$

Why DICE works: diagnostics and ablations. We isolate the mechanism with two analyses, deferred to the appendix for space. **Gradient diagnostics** (Sec. E): under endpoint supervision, the gradient at early iterations is anti-aligned with the endpoint signal ($\cos < 0$ for the first two-thirds of the chain) and has mini-batch variance of 10^2–10^4. DICE keeps gradients aligned and compresses variance by 6–8 orders of magnitude past step 10. **Four-way ablation** (Sec. F): comparing endpoint, repeated-final-state loss (same extra terms but no path shortening), intermediate losses with *separate* heads, and DICE separates the contributions of gradient shortening and shared representation. The ordering **DICE** $>$ **separate heads** $>$ **repeated final** $>$ **endpoint** is clean: both mechanisms contribute, and the shared readout adds value on top of gradient shortening alone. Sec. K gives a matching toy analysis showing that the relative noise of a length-L backpropagation path grows as $(1 + \sigma^2/\mu^2)^L - 1$, so longer paths under endpoint supervision are exponentially noisier than the short paths DICE provides.

Table 5: **Universal Transformer on bAbi benchmark.** Comparison of training strategies.

Method	**Test accuracy**	**vs endpoint**
Endpoint	0.8972	–
GRPO	0.8988	+0.16 pp
LSRL	0.8982	+0.10 pp
RLTT	0.8987	+0.15 pp
DICE	0.9369	+3.97 pp

4. Related Work

Iterative architectures. Universal Transformers [8], DEQs [2, 3], Deep Thinking networks [26], and looped Transformers [11, 12, 14, 31, 34] all iterate a shared function under endpoint supervision and are the inner loops of recent reasoning-focused LLM systems. DICE is a drop-in replacement for the training mechanism of any of them.

Process supervision. Lee et al. [19] and Szegedy et al. [29] add auxiliary classifiers at intermediate layers of feedforward networks, where each layer has *distinct* parameters and deep supervision improves per-layer features. In iterative models, parameters are *shared*, so dense supervision reshapes the *dynamics* of a single recurrent system rather than improving features at independent layers (Sec. F). Lightman et al. [21] supervise reasoning steps at inference via a learned verifier, and Uesato et al. [30] compare process and outcome supervision for LLM reasoning. DICE provides a step-level signal at *training* time, shaping reasoning dynamics from the ground up, while RL post-training [24, 33] must first overcome the degenerate fixed points the base model already learned – a consequential distinction for agents, since the training-time mechanism determines which reasoning trajectories are reachable at all.

Gradient flow. Skip connections [13], gradient clipping [22], and layer normalization [1] address gradient pathologies architecturally. DICE addresses the same problems through the *loss*, applicable when f_θ is difficult to modify. Sec. G shows DICE outperforms the union of these techniques.

5. Discussion

An iterative reasoning agent that can "think longer" before acting is only as good as the guarantees its training procedure imposes on the intermediate states of that thinking. Endpoint supervision imposes none: any internal trajectory that produces a correct final answer at the trained depth is acceptable, even one whose intermediate iterates are nonsensical under the agent's own decoding interface. This is exactly the property an agent should not have if its inference-time compute budget is allowed to vary – as it must in any realistic deployment. Dense supervision is a principled change to the training mechanism: it imposes anytime decodability as a hard constraint on the agent's internal dynamics, making adaptive halting and variable-depth deliberation well-defined rather than emergent.

Limitations. Optimal α and weight schedule vary across systems with no universal default. DICE assumes g_θ produces meaningful predictions at every step, which may not hold for tasks with qualitatively distinct intermediate states (*e.g.*, multi-phase computations). Dense supervision is a strong constraint requiring every iterate to be independently useful; some tasks may benefit from intermediates that support non-trivial multi-step computations without directly producing valid outputs.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

[3] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *Advances in neural information processing systems*, 33:5238–5250, 2020.

[4] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Stabilizing equilibrium models by Jacobian regularization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 554–565. PMLR, 2021.

[5] Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Logical extrapolation without overthinking. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[6] Jay Bear, Adam Prügel-Bennett, and Jonathon Hare. Rethinking deep thinking: Stable learning of algorithms using lipschitz constraints. *Advances in Neural Information Processing Systems*, 37:97027–97052, 2024.

[7] Luke Darlow, Ciaran Regan, Sebastian Risi, Jeffrey Seely, and Llion Jones. Continuous thought machines. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[8] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations (ICLR)*, 2019.

[9] Adji Bousso Dieng, Rajesh Ranganath, Jaan Altosaar, and David Blei. Noisin: Unbiased regularization for recurrent neural networks. In *International Conference on Machine Learning*, pages 1252–1261. PMLR, 2018.

[10] Vasiliy A Es' kin and Mikhail E Smorkalov. Dynamical systems theory behind a hierarchical reasoning model. *arXiv preprint arXiv:2603.22871*, 2026.

[11] Jonas Geiping, Sean 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. *arXiv preprint arXiv:2502.05171*, 2025.

[12] Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning (ICML)*, 2023.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[14] Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

[15] Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026.

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[17] Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29, pages 4601–4609, 2016.

[18] Alex Lamb, Jonathan Binas, Anirudh Goyal, Sandeep Subramanian, Ioannis Mitliagkas, Yoshua Bengio, and Michael Mozer. State-reification networks: Improving generalization by modeling the distribution of hidden representations. In *International Conference on Machine Learning*, pages 3622–3631. PMLR, 2019.

[19] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 562–570, 2015.

[20] Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping, 2024.

[21] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

[22] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1310–1318, 2013.

[23] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in neural information processing systems*, 30, 2017.

[24] Hangliang Ren. Lsrl: Process-supervised grpo on latent recurrent states improves mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12534–12545, 2025.

[25] Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Arpit Bansal, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. Datasets for studying generalization from easy to hard examples. *arXiv preprint arXiv:2108.06011*, 2021.

[26] Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[27] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[28] DiJia Su, Sainbayar Sukhbaatar, Michael Rabbat, Yuandong Tian, and Qinqing Zheng. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces. In *International Conference on Learning Representations (ICLR)*, 2025.

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[30] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

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

[32] Jason Weston, Antoine Bordes, Sumit Chopra, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. In *International Conference on Learning Representations (ICLR)*, 2016.

[33] Jonathan Williams and Esin Tureci. Prioritize the process, not just the outcome: Rewarding latent thought trajectories improves reasoning in looped language models. *arXiv preprint arXiv:2602.10520*, 2026.

[34] Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

Table 6: **Diverse architectures, one abstraction.** Every system is a loop over a shared f_θ.

Architecture	Transition f_θ	Readout g_θ	K	Domain
Universal Transformer	Transformer block	Output projection	Adaptive (ACT)	Language
Deep Thinking	Recurrent block	Linear head	40 (test)	Algorithmic
Recursive Transformer	Transformer cycle	LM head	12	Structured

Appendix A. Impact Statement

This work introduces a general training-time framework for iterative neural computation. Its societal consequences will largely depend on the downstream systems it enables. On the positive side, dense supervision can substantially reduce the inference-time compute required by iterative reasoners, with potential to lower energy consumption and reduce the carbon footprint of large-scale deployment. Our approach may also contribute to more reliable models for high-stakes domains such as scientific discovery, formal verification, automated planning, and education, where step-by-step refinement is more trustworthy than single-pass prediction. The anytime-predictor property exposes intermediate predictions to inspection, which we view as a useful primitive for interpretability and oversight of agentic systems. On the negative side, any technique that makes reasoning models more capable potentially amplifies broader risks associated with advanced AI systems, including misuse for disinformation, automation of harmful tasks, and economic displacement.

Appendix B. INC Instances

A wide range of architectures fit the INC abstraction by instantiating the triple $(h_\theta, f_\theta, g_\theta)$ differently. **Universal Transformers** reuse a single Transformer block across depth, iterating it K times. **Deep Thinking networks** are convolutional or convolutional-recurrent models in which f_θ repeatedly applies residual convolutions; they are designed for algorithmic extrapolation. **Recursive Transformers (TRM)** recursively apply a Transformer block to a hidden state representing a partial solution. Tab. 6 summarizes how each family instantiates the encoder, block, readout, and choice of K.

Appendix C. Adaptive Stability Halting: Schematic

Appendix D. Implementation Details

Universal Transformer. Hidden size 128, 4 heads, max 10 ACT steps. Adam, lr 10^{-3}, batch 64, 100 epochs, patience 15. Readout: shared output projection at each ACT step. Sweep: 3 schedules $\times$ 4 α values $\times$ 3 seeds = 36 runs per task.

DEQ. MDEQ-Large, 3 resolution streams. SGD with momentum 0.9, initial lr 0.1, cosine annealing, weight decay 10^{-4}, batch 128, 60 epochs. Broyden solver: 27 steps at train and test. Intermediate readout applied to K evenly-spaced solver iterates using the shared classification head. Reported best schedule: exponential ($w_k \propto 2^k$) with $K = 5$.

Deep Thinking. Recurrent architecture, hidden size 256. Adam, lr 10^{-3}, decay $\times 0.1$ at epoch 60, batch 64, 80 epochs. Linear weight schedule ($w_k \propto k$).

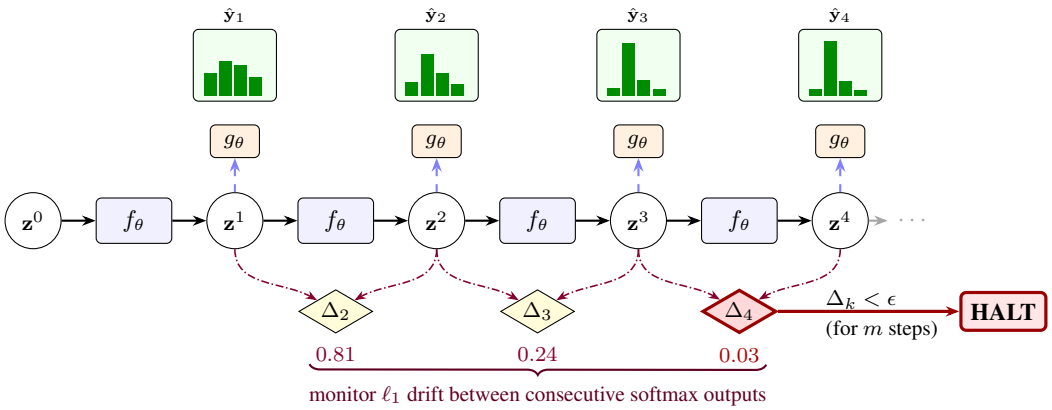

Figure 2: **Adaptive halting via probability stability.** At each iteration k, the shared readout g_θ produces a predictive distribution $\hat{\mathbf{y}}_k$ (green histograms). We compare consecutive softmaxed predictions through the ℓ_1 change Δ_k (purple diamonds). As the network converges, the distribution sharpens and stabilizes; once Δ_k falls below threshold ϵ for m consecutive steps, computation halts. This avoids the failure modes of discrete agreement (*e.g.*, argmax flipping between near-equivalent modes) and tracks confidence-level convergence.

Recursive Transformer (TRM). L=2 layers, H=3 high-level cycles, L_c=4 low-level cycles. Adam, lr 10^{-4}, weight decay 1.0, cosine schedule, global batch 384, EMA model, 4,000 epochs, eval every 1,000 epochs. Readout: shared LM head applied at each H-cycle. Gradient checkpointing over H-cycles for memory efficiency. ASH: $\epsilon = 5 \times 10^{-3}$, patience $m = 1$.

Compute. All experiments were run on NVIDIA GeForce RTX 2080 Ti GPUs and NVIDIA A100 GPUs.

Appendix E. Direct Measurement of Gradient Pathology

We verify the gradient-signal degradation directly: under endpoint supervision, the gradient at early iterations is **both poorly aligned with the endpoint signal and orders of magnitude noisier across mini-batches**.

Setup. We train the Universal Transformer on bAbI 10k with $K = 6$ unrolled steps and a Deep Thinking model on prefix sums with $K = 30$. At each training step, we log two per-step diagnostics for every iterate $k \in \{1, \ldots, K\}$. **(1) Gradient alignment with the endpoint.** Let $g_k = \nabla_\theta \ell(g_\theta(\mathbf{z}^K), \mathbf{y})$ denote the parameter-space gradient produced by reading out at step k. We report $\cos(g_k, g_K)$, where g_K is the gradient at the endpoint. **(2) Mini-batch gradient variance.** For each step k, we compute the per-example gradients $g_k^{(i)}$ and report the trace of their covariance, $\mathrm{Var}_i[g_k^{(i)}]$. We compare endpoint supervision and DICE; everything else is held fixed.

Results. Under endpoint supervision the cosine similarity between g_k and the reference signal g_K is *negative* for the first two-thirds of the chain, hovering between -0.6 and -0.2, and only collapses to 1.0 at $k = K$ by construction (Fig. 3a). The optimizer is asked to update f_θ based on a signal that, for the majority of iterations, points in nearly the opposite direction of what the endpoint loss would

prefer. With DICE, the auxiliary loss at step k produces a gradient whose direction is well aligned with the endpoint within roughly 10 of 30 steps on Deep Thinking, and never becomes anti-aligned. The shortened gradient path replaces a single long-range, direction-corrupted signal with a dense field of locally faithful ones. Under endpoint supervision, mini-batch variance at intermediate steps reaches $\sim 10^2$–10^4 (Fig. 3b); DICE compresses this variance by 6–8 orders of magnitude past step 10 (down to $\sim 10^{-7}$).

Appendix F. Decomposing the Mechanism: Four-Way Ablation

To disentangle the effects of **gradient shortening**, **representation constraint**, and **additional regularization**, we ablate four variants on the Universal Transformer. (1) **Endpoint baseline**: supervision on the final state only. (2) **Final-state repeated-loss control**: adds the same number of auxiliary loss terms with identical total weight, but applies all to $\mathbf{z}^K$; controls for increased supervision and effective loss scale without shortening the gradient path. (3) **Intermediate losses with separate heads**: supervision at intermediate steps with independent readout heads g_k per step, preserving shorter gradient paths while removing the shared-decoding constraint. (4) **DICE**: intermediate losses with a *shared* readout head.

The decomposition is clean (Tab. 7). **(3)** $>$ **(2)** isolates **gradient shortening**: improvements are not from extra regularization but from shorter backpropagation paths. **(4)** $>$ **(3)** isolates the **shared-representation constraint**: enforcing a common decodable latent space across steps adds value beyond gradient shortening alone. **(4)** $>$ **(1)** is the combined effect.

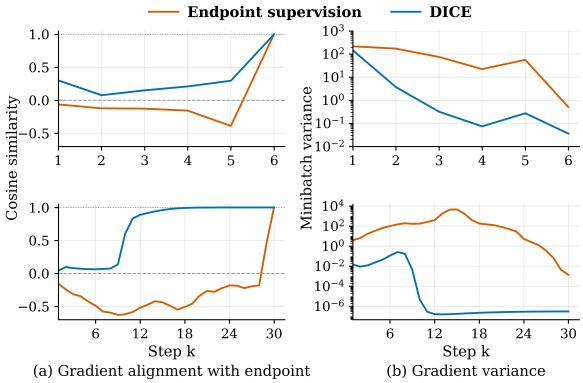

Figure 3: **Gradient pathology under endpoint supervision.** (a) Cosine similarity between step-k and endpoint gradients: endpoint supervision yields anti-aligned early iterates ($\cos < 0$), snapping to 1.0 only at $k = K$, while DICE aligns within a few steps. (b) Per-step gradient variance across mini-batches (log scale). Top is Universal Transformers, bottom is Deep Thinking model.

Table 7: **Four-way ablation on the Universal Transformer (bAbI 10k).** The improvement decomposes into a gradient-shortening effect (#3 vs. #2) and a shared-representation effect (#4 vs. #3).

Mode	Mean acc.	Δ vs. previous
(1) Baseline (endpoint)	0.8972	—
(2) Final-repeated (extra regularization only)	0.9155	+1.83 pp
(3) Separate heads (short gradients only)	0.9311	+1.56 pp
(4) Shared head (full DICE)	**0.9369**	+0.58 pp

Appendix G. DICE vs. Post-Hoc Stabilization

Given that the pathology is one of *direction*, not just magnitude, post-hoc techniques that target gradient norm should provide only marginal relief. We confirm this on bAbI 1k using the Universal Transformer (exponential schedule, $\alpha = 0.5$).

Tab. 8 shows that skip connections, state normalization, and gradient clipping provide only incremental gains over the baseline (51.25% $\rightarrow$ 54.75% when combined). DICE achieves 57.64%, outperforming the best combined baseline by 2.9 pp. Injecting short-range gradient signals at every iteration is fundamentally more effective than post-hoc stabilization of a single long, noisy path.

Table 8: **DICE vs. post-hoc stabilization on bAbI 1k.** Mean accuracy across tasks. Standard stabilization techniques target gradient magnitude; DICE targets the underlying long-range gradient signal.

Method	Transition skip	State norm	Grad clip norm	Mean acc.
Baseline (None)	No	No	–	0.5125
Norm only	No	Yes	–	0.5153
Skip only	Yes	No	–	0.5365
Clip only	No	No	1.0	0.5437
Skip + Norm + Clip	Yes	Yes	1.0	0.5475
DICE (ours)	No	No	–	**0.5764**

Appendix H. Catastrophic Training Collapse in Deep Equilibrium Models

DEQs replace explicit unrolling with a fixed point $\mathbf{z}^\infty = f_\theta(\mathbf{z}^\infty)$ found via root-finding solver iteration. Depth K adapts to input difficulty, and gradients flow through the implicit function theorem rather than every iterate. This experiment exhibits gradient-signal degradation in its starkest form: the Broyden solver has the longest iteration chain among our systems ($K \approx 27$) and produces an unambiguous training failure that DICE eliminates.

Setup. MDEQ-Large [3] on CIFAR-10 [16], 60 epochs. Broyden solver $\sim$27 iterations. We supervise $K = 5$ evenly-spaced solver iterates using the classification head, with exponential weight schedule.

Problem. At epoch 13, the baseline DEQ suffers a catastrophic collapse: test accuracy drops from $\sim$85% to 39.65%. It recovers and eventually reaches 91.08%, but the collapse wastes training budget. The cause is the $\sim$27-step gradient chain through the Broyden solver: an early-step perturbation is amplified through 27 Jacobian products.

Solution. With dense supervision, the model passes through epoch 13 stably at 85.17% (Tab. 10). Intermediate losses anchor each solver iterate to a low-loss region of latent space. Beyond stability, DICE achieves **91.65%** top-1 accuracy ($+0.57$ pp) and reaches 90% at epoch 32 vs. epoch 40 for the baseline – 20% faster convergence.

Gradient clipping on DEQ. Clipping at $\{1.0, 5.0, 10.0\}$ still admits the epoch-13 collapse, though less severely ($\sim$55% instead of 39.65%). DICE eliminates it entirely. Clipping treats the symptom; DICE treats the cause.

Table 9: **DEQ on CIFAR-10.** Best top-1 accuracy (%). DICE with exponential schedule and $K{=}5$ improves over the endpoint baseline.

Configuration	Acc.	Δ
Baseline (endpoint)	91.08	—
DICE (exp., $K{=}5$)	**91.65**	+0.57

Table 10: **Training stability at DEQ epoch 13.** The baseline collapses to near-random; DICE passes stably.

Configuration	Acc. at ep. 13 (%)
Baseline (endpoint)	39.65
DICE	**85.17**

Appendix I. Constraining Intermediate States via a Shared Readout

Prior work has attempted to regularize intermediate representations indirectly. Noisin [9] injects noise into hidden states. Professor Forcing [17] enforces distributional alignment via an adversarial discriminator. State-Reification [18] uses a denoising autoencoder. The latter two require auxiliary networks.

On bAbI 10k with the Universal Transformer (Noisin std 0.25, 100 epochs, patience 20, 5 tasks, 3 seeds; Tab. 11), Noisin gives $+1.5$ pp; DICE gives $+4.0$ pp without stochasticity or auxiliary networks. Prior methods shape the *distribution* of hidden states; DICE constrains their *function*: every state must be decodable by the same head. This shifts the problem from regularization to representation design.

Table 11: **Constraining intermediate states on bAbI 10k.** Mean accuracy across tasks.

Method	Mean acc.	vs. baseline
Baseline	0.8972	—
Noise injection (Noisin)	0.9109	+1.5 pp
DICE (ours)	**0.9369**	+4.0 pp

Appendix J. Which Weight Schedule, and Why

The optimal schedule depends on which failure mode dominates (Tab. 12). **Exponential** suits long-K systems where Problem 1 dominates (consistent with DEQ). **Linear** suits Problem 2 (extrapolation needs progressive improvement; consistent with prefix sums). **Uniform** suits short-K systems where each cycle should contribute comparably.

Training longer. The maze baseline run for 7,000 epochs (75% longer) reaches 79.5%, still below DICE's 81.05%. Longer training does not close the gap.

Appendix K. Theoretical Evidence

K.1. Relative Gradient Noise under Endpoint and Dense Supervision

Motivation. Under endpoint supervision, the gradient signal for an early transition must pass through a product of many Jacobians. Even if the gradient norm is controlled, this long product can amplify stochastic variation in direction and magnitude. We isolate this effect in a scalar model of multiplicative gradient noise: let a length-L backpropagation path carry a multiplicative factor

Table 12: **Best DICE configuration across systems.** The optimal schedule correlates with the primary problem being solved.

System	Best schedule	α	K	Rationale
DEQ (MDEQ)	Exponential	2.0	$\sim$27	Long chain; emphasis on later (better) iterates
Universal Transformer	Exponential	0.5	Adaptive (ACT)	Early iterates poor; weight later halts
Deep Thinking	Linear	1.0	40	Extrapolation needs progressive improvement
Recursive TRM	Uniform	1.0	12	Few cycles; each must contribute equally

$P_L = \prod_{j=1}^{L} A_j$, with A_j i.i.d. random Jacobian factors. We measure noise by relative variance, $\mathrm{RV}(P_L) := \mathrm{Var}(P_L)/\mathbb{E}[P_L]^2$.

Proposition 1 (Exponential growth of relative gradient noise) *Let $A_1, \ldots, A_L$ be independent with $\mathbb{E}[A_j] = \mu \neq 0$, $\mathrm{Var}(A_j) = \sigma^2$, and $P_L = \prod_{j=1}^{L} A_j$. Then $\mathrm{RV}(P_L) = (1 + \sigma^2/\mu^2)^L - 1$.*

Proof $\mathbb{E}[P_L] = \mu^L$ and $\mathbb{E}[P_L^2] = \prod_j \mathbb{E}[A_j^2] = (\mu^2 + \sigma^2)^L$ by independence. Thus $\mathrm{Var}(P_L) = (\mu^2 + \sigma^2)^L - \mu^{2L}$. Dividing by μ^{2L} gives the claim. ■

Implication. Endpoint supervision delivers, to step t, relative noise $(1 + \sigma^2/\mu^2)^{K-t} - 1$; an auxiliary loss at step $m < K$ delivers $(1 + \sigma^2/\mu^2)^{m-t} - 1$. Since $m - t < K - t$, intermediate losses provide strictly smaller relative multiplicative noise.

K.2. Prediction-Space Stability under Dense Supervision

Motivation. The adaptive-halting rule stops when consecutive predictions stabilize: $\|\mathrm{softmax}(\hat{y}_{k+1}) - \mathrm{softmax}(\hat{y}_k)\|_1 \leq \epsilon$. We show that when the DICE loss is small, all supervised predictions are close to the same target distribution, hence close to each other. For classification, let $p_k(x) = \mathrm{softmax}(g_\theta(z_k(x)))$ and let $e_{\mathbf{y}(x)}$ be the one-hot target. With cross-entropy loss, the DICE objective is $L_{\mathrm{DICE}}(\theta) = \sum_k \lambda_k \mathbb{E}_x[\ell_{\mathrm{CE}}(p_k(x), \mathbf{y}(x))]$ where $\lambda_K = 1$ and $\lambda_k = \alpha w_k$ for $k < K$.

Proposition 2 (Prediction-space stability) *If $L_{\mathrm{DICE}}(\theta) \leq \delta$, then $\mathbb{E}_x[\|p_k(x) - e_{\mathbf{y}(x)}\|_1] \leq 2\delta/\lambda_k$ and $\mathbb{E}_x[\|p_{k+1}(x) - p_k(x)\|_1] \leq 2\delta(\frac{1}{\lambda_{k+1}} + \frac{1}{\lambda_k})$.*

Proof For a one-hot e_y, $\|p - e_y\|_1 = 2(1 - p[y])$. Since $-\log u \geq 1 - u$ on $(0, 1]$, $\|p - e_y\|_1 \leq 2\ell_{\mathrm{CE}}(p, y)$. Nonnegativity gives $\lambda_k \mathbb{E}_x[\ell_{\mathrm{CE}}(p_k, \mathbf{y})] \leq \delta$, so $\mathbb{E}_x[\|p_k - e_{\mathbf{y}}\|_1] \leq 2\delta/\lambda_k$. The triangle inequality gives the second claim. ■

Implication. Theorem 2 controls exactly the quantity used by adaptive halting. Endpoint supervision cannot give this guarantee: it controls only p_K, allowing the model to learn a fixed-horizon strategy with arbitrary intermediate predictions.

Camera-Ready Edits Summary
Revise the discussion.