DENSE SUPERVISION FOR ITERATIVE NEURAL COMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Iterative Neural Computations (INCs), such as Universal Transformers and Deep Thinking networks, have demonstrated strong capabilities across algorithmic, reasoning, and perceptual tasks, by iteratively applying a shared function multiple steps. Despite their promise, training these systems remains challenging. These models are typically trained by applying a loss only at the final iterate, a recipe we call *endpoint supervision*. We hypothesize that this recipe is the cause behind two coupled failure modes: (1) gradients from the final loss must backpropagate through the entire chain of repeated updates, producing a learning signal that is both noisy in magnitude and unreliable in direction, and (2) intermediate iterations are left unconstrained, allowing the model to learn a fixed-horizon strategy that breaks down when more iterations are run at test time. We introduce Dense Intermediate Consistency for Endpoints (DICE), an architecture-agnostic framework that addresses both issues by supervising the full trajectory of computation. DICE attaches a *shared* readout head to every iterate and applies auxiliary losses at each step, providing gradient signals that bypass the deep composition while enforcing that every intermediate state remains semantically meaningful. Across three representative architectures, DICE yields consistent gains, including 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. Overall, these results indicate that dense supervision is a simple and broadly effective remedy for the failure modes of endpoint-supervised INCs.

1 INTRODUCTION

Many modern neural networks solve problems by thinking iteratively rather than in a single pass: they refine an internal representation step by step before committing to an answer. This pattern underlies a diverse and growing family of architectures, including Universal Transformers (Dehghani et al., 2019), Deep Equilibrium Models (Bai et al., 2019), Deep Thinking networks (Schwarzschild et al., 2021b), looped recursive Transformers (Geiping et al., 2025), and looped language models (Giannou et al., 2023; Zhu et al., 2025). In this paper, we unify these approaches under a single concept, which we term Iterative Neural Computations (INCs). Despite their differences in architecture, target domain, and training regime, these models often share a common training recipe: **a loss is computed only at the final iterate, and gradients are propagated back through the entire chain of updates** (Fig. 1).

Yet, INCs trained under this recipe still struggle to generalize on complex reasoning tasks that require long trajectories of computation (Kohli et al., 2026; Bear et al., 2024; Es' kin & Smorkalov, 2026). These struggles have been documented in prior work, but they are typically treated as architecture-specific issues: training instabilities to be patched (Bai et al., 2019; 2021; Geiping et al., 2025), or extrapolation failures to be engineered around (Schwarzschild et al., 2021b; Bansal et al., 2022). We hypothesize that the shared cause is *endpoint supervision* 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 before reaching them; and (ii) intermediate iterations are left entirely unconstrained, so the model has no incentive to behave well beyond the trained iteration count.

Based on this hypothesis, we propose a simple, model-agnostic framework, Dense Intermediate Consistency for Endpoints (DICE). DICE adds auxiliary losses at intermediate iterations, decoded

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

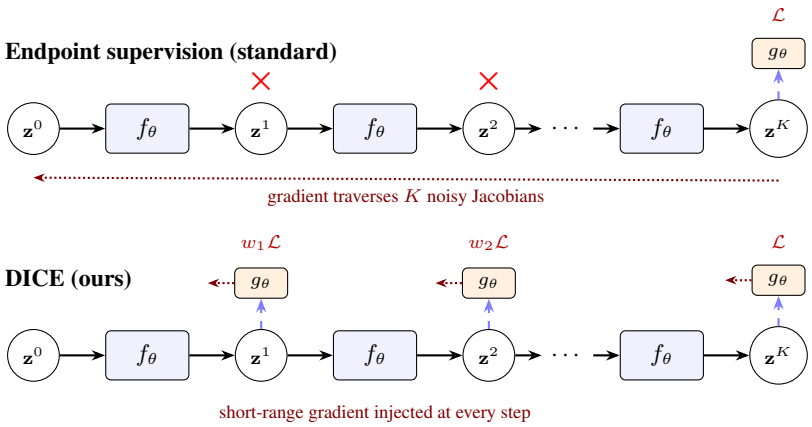


Figure 1: **The two problems and how dense supervision solves them.** Top: endpoint supervision provides a single gradient that must traverse K Jacobians (**Problem 1**) and leaves intermediate states unsupervised (**Problem 2**, denoted as $\times$). Bottom: DICE adds a *shared* readout g_θ at every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

through a readout head that is shared with the endpoint. The auxiliary losses shorten the gradient paths that reach early steps, while the shared readout head forces every intermediate iterate to be a valid prediction in its own right. The change is architecture-agnostic, adds less than 5% to training compute, and requires no modification to the underlying iterative model. Empirically, DICE delivers substantial gains across three representative INCs: near-perfect extrapolation on prefix-sum tasks where the endpoint baseline collapses to near-zero accuracy, $+7.4$ pp exact-match accuracy on 30×30 maze solving with a recursive Transformer, and $+3.97$ pp on bAbI with a Universal Transformer, together with a $6.6\times$ inference-time speedup via adaptive halting. Direct measurements of gradient alignment and variance further confirm that DICE mitigates the gradient-degradation pathology. Overall, dense supervision emerges as a simple but consistently effective remedy for the failure modes of endpoint-supervised INCs.

2 Problem Setup

We first define the formal abstraction of Iterative Neural Computations, and then describe precisely the two failure modes induced by endpoint supervision. The targeted solutions and the combined DICE objective are derived in Sec. 3.

2.1 Iterative Neural Computations and Endpoint Supervision

Iterative Neural Computations. We consider models that produce a prediction by repeatedly applying a shared update rule to a hidden state. Concretely, given an input $\mathbf{x} \in \mathcal{X}$ and a target $\mathbf{y} \in \mathcal{Y}$, the model is specified by three components:

- an **encoder** $h_\theta : \mathcal{X} \to \mathcal{Z}$ mapping $\mathbf{x}$ to initial hidden state $\mathbf{z}^0 = h_\theta(\mathbf{x})$ in a latent space $\mathcal{Z}$;
- an **update function** (or *block*) $f_\theta : \mathcal{Z} \to \mathcal{Z}$, applied repeatedly to refine the state;
- a **readout head** $g_\theta : \mathcal{Z} \to \mathcal{Y}$ that decodes a hidden state into an output in the target space $\mathcal{Y}$.

All three components share the trainable parameter vector θ. The model unrolls f_θ for $K \in \mathbb{N}$ steps, producing a trajectory of hidden states

$$\mathbf{z}^0 \xrightarrow{f_\theta} \mathbf{z}^1 \xrightarrow{f_\theta} \cdots \xrightarrow{f_\theta} \mathbf{z}^K, \qquad \mathbf{z}^k := f_\theta(\mathbf{z}^{k-1}) \ \text{ for } k = 1, \ldots, K, \tag{1}$$

where $\mathbf{z}^k \in \mathcal{Z}$ denotes the hidden state at iteration k, and the integer K is the *depth* or *number of iterations*. The final prediction is read out from the last state, $\hat{\mathbf{y}} = g_\theta(\mathbf{z}^K)$. We refer to $\mathbf{z}^1, \ldots, \mathbf{z}^{K-1}$ as the *intermediate iterations* and to $\mathbf{z}^K$ as the *endpoint*.

Table 1: **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

A wide range of architectures fit this abstraction by instantiating the triple $(h_\theta, f_\theta, g_\theta)$ in different ways. We highlight three representative families that we consider in our analysis.

Universal Transformers reuse a single Transformer block across depth, iterating it K times instead of using K separate layers. The hidden state $\mathbf{z}^k$ represents all token embeddings, h_θ initializes them with positional encoding, and g_θ maps selected tokens to outputs. Parameter sharing turns depth into recurrence over a fixed-length sequence, and these models are commonly studied on synthetic reasoning, algorithmic, and language tasks that require more steps than fixed-depth models allow.

Deep Thinking networks are convolutional or convolutional-recurrent models in which h_θ embeds the raw input (e.g., a grid) into a spatial feature map, f_θ repeatedly applies residual convolutions to propagate information across it, and g_θ decodes per-position predictions. Designed explicitly for algorithmic extrapolation, they are evaluated on prefix-sum, maze solving, and chess puzzles, where increasing K at test time should enable larger or harder instances than seen during training.

Recursive Transformers (e.g., TRM) recursively apply a Transformer block f_θ to a hidden state representing a partial solution, conditioned on the input at each step. Compared to looped Transformers, the emphasis is on framing each application of f_θ as refining a candidate output, rather than fixed layer tying. They are typically benchmarked on multi-step reasoning, planning, and puzzles, where answers emerge by progressively revising an internal hypothesis.

Tab. 1 summarizes how each family instantiates the encoder, block, readout, and choice of K. Across them, f_θ may be a tied-weight Transformer block, or a recursive refinement Transformer, with K fixed in advance, sampled during training, or set by a solver at inference. Our analysis is agnostic to these choices and depends only on the shared structure: a trajectory of hidden states generated by repeated application of f_θ, with supervision on the endpoint $\mathbf{z}^K$.

Endpoint loss. Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ denote a per-example loss function (*e.g.*, cross-entropy). The standard training objective, which we call the *endpoint loss*, supervises only the final iterate:

$$\mathcal{L}_{\text{end}}(\theta; \mathbf{x}, \mathbf{y}) \;=\; \ell\big(g_\theta(\mathbf{z}^K),\, \mathbf{y}\big)\,. \tag{2}$$

Throughout the paper, expectations over the data distribution are left implicit; statements about $\mathcal{L}_{\text{end}}$ should be read as applying to a single example or to its empirical average, as appropriate.

2.2 Two Failure Modes of Endpoint Supervision

We now state the two technical problems that endpoint supervision induces. They share a common cause in that the loss in Eq. (2) depends on $\mathbf{z}^1, \ldots, \mathbf{z}^{K-1}$ *only* through the chain that produces $\mathbf{z}^K$, but they manifest in different ways and motivate different remedies.

Gradient-signal degradation through the iteration chain. Training requires the parameter gradient $\nabla_\theta \mathcal{L}_{\text{end}}$, which by the chain rule decomposes into contributions from each iteration k, where the contribution from iteration k traverses the Jacobian product $\mathbf{J}_{k\to K} := \prod_{j=k}^{K-1} \frac{\partial f_\theta(\mathbf{z}^j)}{\partial \mathbf{z}^j} \in \mathbb{R}^{d\times d}$, where d is the dimension of the latent space $\mathcal{Z}$, and $\partial f_\theta(\mathbf{z}^j)/\partial \mathbf{z}^j$ is the Jacobian of f_θ at $\mathbf{z}^j$ – the same product that drives vanishing- and exploding-gradient behavior in deep recurrent networks (Pascanu et al., 2013). Even when the magnitude of $\mathbf{J}_{k\to K}$ is well controlled (*e.g.*, by normalization or spectral constraints), its *direction* accumulates noise as $K - k$ grows (Pennington et al., 2017). We observe this failure mode in the Multiscale Deep Equilibrium Model (Sec. B.2), whose fixed point $\mathbf{z}^\star = f_\theta(\mathbf{z}^\star)$ is found by a Broyden solver-a quasi-Newton method that refines $\mathbf{z}$ via a low-rank Jacobian approximation. The solver unrolls $K \approx 27$ steps, and at epoch 13 training collapses: top-1 accuracy drops from $\sim 85\%$ to 39.65% before recovering. The collapse is driven by noisy gradients reaching the early solver steps that anchor the iterate trajectory in a stable region of latent space.

Unconstrained intermediate iterations. The endpoint loss in Eq. (2) treats $\mathbf{z}^1, \ldots, \mathbf{z}^{K-1}$ as latent scratch space: any trajectory that produces a correct $\mathbf{z}^K$ is equally rewarded, regardless of what the intermediate iterations encode. As a result, intermediate iterations need not lie in a region of $\mathcal{Z}$ from which g_θ can produce a meaningful output; in particular, they need not represent partial or incrementally improving estimates of $\mathbf{y}$. This becomes critical when the test-time iteration count K_{test} differs from the training count K_{train}, as is standard practice for iterative reasoning models that aim to extrapolate to harder problems by "thinking longer." For $K_{\text{test}} > K_{\text{train}}$, the iterations $\mathbf{z}^{K_{\text{train}}+1}, \mathbf{z}^{K_{\text{train}}+2}, \ldots, \mathbf{z}^{K_{\text{test}}}$ lie outside the regime ever reached during training, and the model has no incentive to make f_θ behave well on them. This is the algorithmic-extrapolation difficulty identified by Schwarzschild et al. (2021b): success requires *more* computation than was seen in training, but the loss provides no pressure on what intermediate computation should look like.

3 DICE: Dense Intermediate Consistency for Endpoints

We propose Dense Intermediate Consistency for Endpoints (DICE), a novel framework for iterative neural computation that addresses both failure modes of endpoint supervision identified in Sec. 2.2. DICE is composed of two components: (i) **Dense Supervision** and (ii) **Adaptive Stability Halting**.

3.1 Dense Supervision

Recall from Sec. 2.1 that $f_\theta : \mathcal{Z} \to \mathcal{Z}$ is the iterative update function, $g_\theta : \mathcal{Z} \to \mathcal{Y}$ is the readout head that decodes a hidden state into a prediction, $\mathbf{z}^k$ is the hidden state at iteration k, $\mathbf{y}$ is the target, and ℓ is a per-example loss. The DICE objective is:

$$\mathcal{L}_{\text{dense}} = \underbrace{\ell\big(g_\theta(\mathbf{z}^K),\, \mathbf{y}\big)}_{\text{endpoint loss}} + \alpha \sum_{k=1}^{K-1} \underbrace{w_k}_{\text{schedule}} \underbrace{\ell\big(g_\theta(\mathbf{z}^k),\, \mathbf{y}\big)}_{\text{auxiliary loss at step } k} \tag{3}$$

The objective has two hyperparameters. The scalar $\alpha > 0$, which we call the *auxiliary strength*, sets the overall weight of the intermediate losses relative to the endpoint loss. The sequence $\{w_k\}_{k=1}^{K-1}$, which we call the *weight schedule*, allocates this weight across iterations and is normalized so that $\sum_{k=1}^{K-1} w_k = 1$. We consider three natural schedules: *uniform* ($w_k \propto 1$), which weights all iterations equally; *linear* ($w_k \propto k$), which gradually increases the weight toward later iterations; and *exponential* ($w_k \propto 2^k$), which concentrates weight near the endpoint. Crucially, the readout head g_θ in Eq. (3) is the *same* mapping at every iteration k, with no per-step parameters.

Two design elements deserve emphasis. First, the auxiliary loss at iterate k backpropagates through only k applications of f_θ, rather than through all K, providing a short-range gradient signal at every point along the iteration trajectory and directly addressing the gradient-degradation issue. Second, the readout head g_θ is shared across all iterations, forcing every iterate to lie in a region of $\mathcal{Z}$ that decodes to a meaningful prediction and addressing the unconstrained-iterations issue. This design requires no architectural changes and adds negligible cost: $K-1$ additional evaluations of a (typically linear) readout head per training step, less than 5% of total compute in all of our settings.

3.2 Adaptive Stability Halting

Beyond improved training, DICE provides a useful inference-time property. Because every intermediate iterate $\mathbf{z}^k$ is *decodable* through the shared readout g_θ, DICE-trained models behave as anytime predictors: predictions tend to improve monotonically with additional computation and stabilize before the maximum iteration count is reached. This enables a simple *adaptive halting* scheme that terminates iteration once the prediction has converged.

Method. Let $\hat{\mathbf{y}}_k = g_\theta(\mathbf{z}^k)$ denote the prediction at iteration k. We monitor the ℓ_1 change in the predictive distribution and stop when it falls below a threshold for m consecutive steps:

$$\left\|\mathrm{softmax}(\hat{\mathbf{y}}_k) - \mathrm{softmax}(\hat{\mathbf{y}}_{k-1})\right\|_1 < \epsilon.$$

Compared with discrete agreement (*e.g.*, argmax consistency), this criterion captures diminishing updates in confidence, avoids premature stopping when predictions oscillate between near-equivalent modes, and yields a more reliable signal of true convergence. We describe the probability-stability variant in Fig. 2.

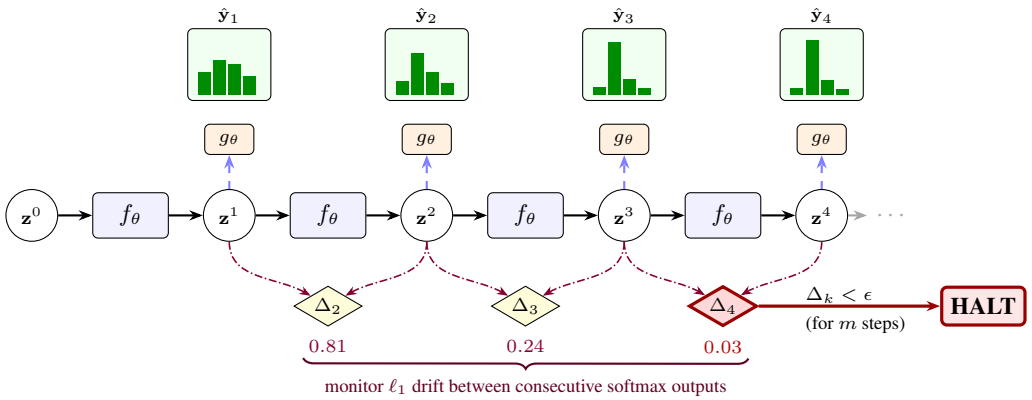


Figure 2: **Adaptive halting via probability stability.** At each iteration k, the shared readout g_θ produces a predictive distribution $\hat{\mathbf{y}}_k$ (green histograms). We compare consecutive softmaxed predictions through the ℓ_1 change Δ_k (purple diamonds). As the network converges, the distribution 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.

Interpretation. Because each iterate is trained to produce a valid prediction under a shared readout, convergence in prediction space is a reliable proxy for convergence in representation space. The halting condition therefore approximates a fixed-point detector in prediction space, without requiring access to the internal latent dynamics. In practice, many inputs converge early, yielding substantial reductions in average iteration count without retraining or modifying the model. Appendix D.2 shows that, for the cross-entropy losses used in our experiments, a small dense objective directly bounds this prediction-space drift, connecting the halting criterion to the training loss.

4 Experiments

We evaluate DICE on three iterative systems. For each, we keep the base architecture and training procedure unchanged; the only modification is the loss function: we add intermediate readout losses as in Eq. (3). We first present our main results from Sec. 4.2 to 4.4, then a direct measurement of the gradient pathology DICE is designed to fix, a four-way ablation that decomposes DICE into its constituent mechanisms (Sec. 4.6).

4.1 Baselines

We compare DICE against several strong representative baselines. **Endpoint**: The standard endpoint supervision, serving as our primary baseline. **GRPO** (Shao et al., 2024): Applies a group-normalized policy gradient to the terminal iterate alone, providing outcome-reward credit only at the endpoint. **LSRL** (Ren, 2025): Applies dense intermediate supervision via reinforcement learning, decoding each iterate through the shared readout and using an intermediate verifier to assign per-step rewards optimized with GRPO. **RLTT** (Williams & Tureci, 2026): Distributes the outcome-based reward signal across all iterations by taking a weighted sum of per-step log-probabilities under the shared readout head, providing trajectory-level credit assignment without intermediate verifiers. All baselines use the same amount of training data per task for fair comparison.

4.2 Deep Thinking

Setup. We evaluate DICE against baselines on the prefix sums task (Schwarzschild et al., 2021a). Given a sequence of bits, the task is to produce another bit sequence of the same length, where each position encodes the running sum of previous bits taken modulo two. Sequences that are longer are treated as more challenging than shorter ones. This is the cleanest test of extrapolation: the model must generalize from training-time iteration counts to larger test-time counts. The model is trained

Table 2: **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 3: **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

on 32-bit prefix-sum problems with fixed 30 unrolls; tested on 64-bit, 72-bit, 128-bit, and 256-bit inputs at 40 iterations. Training budget is 80 epochs. For checking the Adaptive Stability Halting (ASH) mechanism, we extend the inference budget to 80 iterations on each prefix-sums width.

Results. The endpoint baseline achieves only 11.70% extrapolation accuracy on 64-bit inputs at 40 test iterations, and degrades to near zero on longer inputs (Tab. 2). Adding dense supervision with a linear schedule yields **100.00%** extrapolation accuracy on 64-, 72-, and 128-bit inputs, and **11.88%** on 256-bit inputs (Tab. 2). The shared readout forces each iterate $\mathbf{z}^k$ to decode, via $g_\theta(\mathbf{z}^k)$, to a meaningful partial solution. This constrains the model to develop a *genuinely iterative* refinement strategy: each step makes explicit progress toward the answer, so applying more steps at test time continues this progress rather than breaking a fixed-length strategy. On the easier widths (64, 72, 128), ASH is a pure compute saving, finish in much fewer than the given 80 iterations budget (Tab. 3).

4.3 Recursive Transformer

Setup. We evaluate DICE on the recursive Transformer architecture on maze navigation task (Lehnert et al., 2024). The task is to find the optimal path through a 30×30 maze, where the shortest solution length exceeds 110 steps. It is commonly used as a benchmark for training large language models on search-oriented problems (Darlow et al., 2025; Su et al., 2025). The base model is the Tiny Recursive Model (TRM) (Jolicoeur-Martineau, 2025) with L=2 layers, H=3 high-level cycles, and L_c=4 low-level cycles ($K = 12$ total). 30×30 hard mazes, 1,000 training examples, 4,000 epochs. DICE is applied at each H-cycle using the shared LM head, with gradient checkpointing for memory efficiency. TRM exposes an outer supervision loop of length N_{sup} that iteratively refines the answer $\mathbf{y}$ by repeatedly invoking the recursive H/L-cycle and re-decoding through the shared readout g_θ. We monitor the stability of the predictive distribution at the outer-step level for adaptive halting.

Results. As shown in Tab. 4b, DICE outperforms all baselines by achieving the highest exact accuracy on maze task. Tab. 4 shows the learning trajectory; two phenomena are visible. The first is convergence acceleration. At epoch 2,000, DICE (w=1.0) achieves 69.7% exact accuracy versus 27.1% for the baseline. DICE surpasses the baseline's best accuracy (73.6%, achieved at 4,000 epochs) by epoch 3,000, using only 75% of the training budget. This acceleration comes from the fact that direct gradients at each H-cycle give f_θ clear, step-specific learning signals about how each cycle should transform the representation, rather than a single noisy signal that must be decomposed across 12 iterations. Second, DICE helps improve final accuracy. The best checkpoint reaches **81.0%** exact accuracy, surpassing the baseline's 73.6% by +7.4 pp. The shared readout forces each H-cycle to produce a progressively better maze solution: cycle 1 must decode to *something*, even a rough partial path, which gives cycle 2 a better starting point, and so on. Without this constraint, the model might use cycle 1 for opaque feature extraction and produce a complete answer only at cycle 3. Using ASH, we have two observations: (i) adaptive halting slightly *improves* accuracy over fixed-N_{sup} inference (81.05% vs. 80.96%), suggesting that later refinement iterations occasionally drift from a converged answer that the halting criterion correctly locks in; and (ii) the average outer-step count of 2.43 confirms that DICE concentrates useful predictions into the earliest refinement steps, as intended by the design of shared-readout constraint.

4.4 Universal Transformers

Setup. We evaluate DICE against baseline methods on the bAbi question answering task (Weston et al., 2016). The aim is that each task tests a unique aspect of language understanding and reasoning,

Table 4: **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.

Table 5: **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×
DICE (w.o. adaptive halt)	16 (fixed)	80.96%	16.00	1.00×
DICE (w. adaptive halt)	≤ 16	**81.05%**	**2.43**	6.58×

including the ability to retrieve facts in a story, relational reasoning, and multi-argument reasoning. The base model is Universal Transformer.

Result. As shown in Tab. 6, DICE outperforms all baselines by achieving the highest test accuracy. The endpoint baseline achieves 89.72% accuracy, while DICE can achieve 93.69 % accuracy, surpassing the endpoint supervision by +3.97 pp. With ASH, DICE use 50% fewer refinement steps.

4.5 Direct Measurement of Gradient Pathology

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

Table 6: **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

Setup. We train the Universal Transformer model on the bAbI 10k dataset with $K = 6$ unrolled steps and a Deep Thinking model on the prefix-sum dataset with $K = 30$ unrolled steps and, at each training step, log two per-step diagnostics for every iterate $k \in \{1, \ldots, K\}$:

1. **Gradient alignment with the endpoint.** Let $g_k = \nabla_\theta \ell\big(g_\theta(\mathbf{z}^k), \mathbf{y}\big)$ denote the parameter-space gradient produced by reading out at step k. We report the cosine similarity $\cos\big(g_k, g_K\big)$, where g_K is the gradient produced at the endpoint. This isolates the *direction* of the learning signal at step k relative to the only signal that endpoint supervision actually receives.
2. **Mini-batch gradient variance.** For each step k we compute the per-example gradients $g_k^{(i)}$ on a mini-batch and report the trace of their covariance, $\mathrm{Var}_i\big[g_k^{(i)}\big]$. This isolates the *stochastic noise* of the signal at step k.

We compare two training regimes that are identical in architecture and optimizer and differ only in the loss: endpoint supervision ($\mathcal{L} = \ell(g_\theta(\mathbf{z}^K), \mathbf{y})$) and DICE (Eq. (3)). Diagnostics are averaged over a fixed evaluation mini-batch, replicated across seeds, at a representative point in training.

Results. As shown in Fig. 3a, 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. In other words, 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. This provides an empirical signature of a gradient direction that degrades along the chain. 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 in Deep Thinking model, and it never becomes anti-aligned. The shortened gradient path replaces a single long-range, direction-corrupted signal with a dense field of locally faithful ones. In addition, as shown in Fig. 3b (bottom), under endpoint supervision, mini-batch variance at intermediate steps reaches $\sim 10^2$–10^4, i.e., the per-example gradients used to update f_θ disagree across examples by several orders of magnitude in squared norm. This is the noisy long-range gradient signal. DICE compresses this variance by 6–8 orders of magnitude past step 10 (down to $\sim 10^{-7}$). Together, they confirm the degradation of the gradient signal directly: endpoint supervision delivers, at most early iterations, a gradient that is wrong in direction and noisy in magnitude.

4.6 Decomposing the Mechanism

Setup. To disentangle the effects of **gradient shortening**, **representation constraint**, and **additional regularization**, we perform a four-way ablation on the Universal Transformer. We compare against three controls that progressively isolate the contributions of intermediate supervision, gradient-path length, and representational sharing. (1) The **Endpoint baseline** uses standard training with supervision applied only to the final state. (2) The **final-state repeated-loss control** adds the same number of auxiliary loss terms with identical total weight, but applies all of them to the final state $\mathbf{z}^K$; this controls for increased supervision, regularization, and effective loss scale without shortening the gradient path. (3) The **intermediate losses with separate heads** variant applies supervision at intermediate steps, but uses independent readout heads g_k for each step, preserving the benefit of shorter gradient paths while removing the constraint of a shared, decodable representation. (4) Finally, **DICE** applies intermediate losses with a *shared* readout head across all steps.

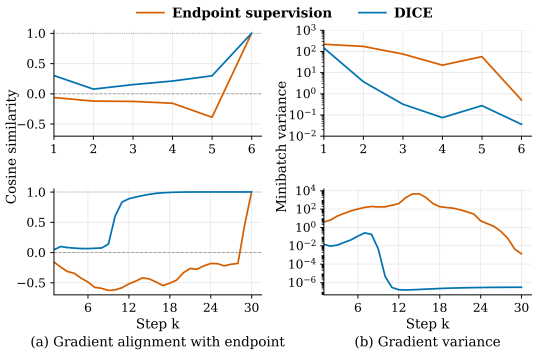


Figure 3: **Gradient pathology under endpoint supervision.** (a) Cosine similarity between step-k and endpoint gradients: endpoint supervision yields anti-aligned early iterates ($\cos < 0$), snapping to 1.0 only at $k = K$, while DICE aligns within a few steps. (b) Per-step gradient variance across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

Results. The decomposition is clean (Tab. 7). The comparison **(3)** $>$ **(2)** isolates the effect of **gradient shortening**, showing that improvements are not due to additional regularization but to shorter backpropagation paths. The comparison **(4)** $>$ **(3)** isolates the effect of the **shared-representation constraint**, where gains arise from enforcing a common, decodable latent space across steps. Finally, **(4)** $>$ **(1)** reflects the **combined effect** of both mechanisms. In Sec. D.1, the theoretical analysis on a simple toy setup assuming random Gaussian Jacobians shows that the noise in the gradient grows

w.r.t increasing reasoning depth in endpoint supervision. Together with the diagnostics in Sec. 4.5, this completes a single causal account of why DICE works.

5 Related Work

Iterative architectures. Universal Transformers (Dehghani et al., 2019), DEQs (Bai et al., 2019; 2020), Deep Thinking networks (Schwarzschild et al., 2021b), and looped Transformers (Giannou et al., 2023; Zhu et al., 2025; Geiping et al., 2025; Wang et al., 2025; Jolicoeur-Martineau, 2025) all iterate a shared function, trained with endpoint supervision. DICE is a drop-in replacement for the loss of any of these systems.

Deep process supervision. Lee et al. (2015) and Szegedy et al. (2015) add auxiliary classifiers at intermediate layers of feedforward networks. There, each layer has *distinct* parameters, so deep supervision improves per-layer feature quality. In iterative models, parameters are *shared*, so dense supervision reshapes the *dynamics* of a single recurrent system rather than improving features at independent layers. This is a qualitatively different effect (Sec. 4.6). Lightman et al. (2024) supervise individual reasoning steps via a learned verifier at inference time. Uesato et al. (2022) compares process and outcome supervision. DICE provides a step-level signal at *training* time via gradient backpropagation. DICE shapes dynamics from the ground up, while RL post-training has to first overcome whatever degenerate fixed points or shortcut behaviors the base model already learned (Ren, 2025; Williams & Tureci, 2026).

Gradient flow in recurrent and deep networks. Skip connections (He et al., 2016), gradient clipping (Pascanu et al., 2013), and layer normalization (Ba et al., 2016) address gradient pathologies through architectural modifications. DICE addresses the same class of problems through the *loss function*, making it applicable even in settings where f_θ is difficult to modify directly. Sec. B.1 shows DICE outperforms the union of these techniques.

6 Limitations

DICE has limitations that future work should address. The optimal α and weight schedule vary across systems with no universal default, suggesting adaptive schedules as a natural next step. DICE also assumes the readout head g_θ produces a meaningful prediction at every step, which may not hold for tasks with qualitatively distinct intermediate states (*e.g.*, multi-phase computations) and could require per-phase readouts. Finally, dense supervision is a strong constraint requiring every iterate to be independently useful; some tasks may instead benefit from intermediate states that support non-trivial multi-step computations without directly producing valid outputs, and understanding when to relax this constraint remains open.

7 Conclusion

Iterative neural models, a growing class spanning Universal Transformers, DEQs, Deep Thinking networks, recursive Transformers, and looped language models, suffer from two problems rooted in endpoint supervision: gradient signal degrades through long iteration chains, and unconstrained intermediate states preclude extrapolation. We address each with a targeted mechanism: intermediate auxiliary losses inject direct gradients, and a shared readout head enforces anytime decodability. Together, these mechanisms form a single framework that requires no architectural changes and adds negligible computational overhead. The result is consistent improvement across three fundamentally different iterative systems. The shared-readout property enables a simple inference-time adaptive-halting scheme that delivers significant speedup at no cost in accuracy. Each gain is traceable to the specific failure mode that dense supervision resolves.

AI USE STATEMENT

We used generative AI tools solely to improve the writing and presentation of this paper, including grammar correction, wording, and clarity. We did not use generative AI tools for generating research ideas, developing methods, conducting experiments, analyzing results, or producing figures, and no other AI-assisted tasks are applicable to this work.

All AI-assisted text was reviewed and edited by the authors. The authors verified the accuracy and consistency of the final text and take full responsibility for the final content of this work, including all claims, results, and artifacts produced with the aid of generative AI.

ETHICS 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 rather than on any specific application studied here. On the positive side, dense supervision can substantially reduce the inference-time compute required by iterative reasoners, which has the potential to lower energy consumption, reduce the carbon footprint of large-scale model deployment. Our approach may also contribute to the development of 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 believe is a useful primitive for interpretability. On the negative side, any technique that makes reasoning models more capable potentially amplifies the broader risks associated with advanced AI systems, including misuse for disinformation, automation of harmful tasks, and economic displacement. We encourage future work to pair such methods with investment in safety and responsible deployment practices.

REPRODUCIBILITY STATEMENT

DICE requires no architectural changes, and every experiment in this paper reuses the base architecture and optimizer of the system it is applied to; the only change is the loss in Eq. (3). Full hyperparameters for all four systems, including weight schedules, auxiliary strengths, optimizer settings, and iteration counts, are given in Sec. A, and the per-experiment configurations, including the compute used, are listed in Sec. C. The datasets are all publicly available benchmarks, cited at first use in Sec. 4. Complete statements and proofs of Propositions 1 and 2 appear in Sec. D.1 and D.2.

REFERENCES

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

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

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

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*, pp. 554–565. PMLR, 2021.

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.

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.

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

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

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

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

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.

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.

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)*, pp. 770–778, 2016.

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

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

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

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, pp. 4601–4609, 2016.

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*, pp. 3622–3631. PMLR, 2019.

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)*, pp. 562–570, 2015.

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.

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.

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

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.

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

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*, 2021a.

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)*, 2021b.

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.

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.

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)*, pp. 1–9, 2015.

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.

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.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, 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.

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.

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.

A 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. No additional parameters. 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 ($w_k \propto k$).

Recursive Transformer. 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.

B Additional Ablations

B.1 DICE vs. Post-Hoc Stabilization

Given that the pathology is one of *direction*, not just magnitude, we should expect post-hoc techniques that target gradient norm to provide only marginal relief. We confirm this on the bAbI 1k question-answering benchmark using the Universal Transformer.

Tab. 8 shows that techniques that directly target gradient magnitude and stability, skip connections, state normalization, and gradient clipping, provide only incremental gains over the baseline (51.25% $\rightarrow$ 54.75% when combined). This is consistent with our claim that the core issue is not just exploding or vanishing norms, but the degradation of gradient *direction* through long Jacobian chains, as measured directly in Sec. 4.5. In contrast, DICE achieves substantially higher accuracy (57.64%), outperforming the best combined baseline by 2.9 absolute points. Injecting short-range gradient signals at every iteration is fundamentally more effective than post-hoc stabilization of a single long, noisy gradient 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**

B.2 Catastrophic Training Collapse in Deep Equilibrium Models

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 (e.g., Broyden or Anderson acceleration). Depth K adapts to input difficulty, and gradients flow through the implicit function theorem rather than every iterate. DEQs target sequence modeling, dense prediction, and structured inference, offering arbitrarily deep computation at constant memory. This experiment demonstrates gradient signal degradation, and exhibits an unambiguous training failure that DICE eliminates.

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; DICE passes stably.

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

Setup. MDEQ-Large (Bai et al., 2020) on CIFAR-10 (Krizhevsky & Hinton, 2009), trained for 60 epochs. The Broyden fixed-point solver runs ∼27 iterations. We supervise $K = 5$ evenly-spaced solver iterates using the classification head, using exponential weight schedules.

Problem. At epoch 13, the baseline DEQ suffers a catastrophic training collapse: test accuracy drops from ∼85% to 39.65% (near random on CIFAR-10). The model recovers over subsequent epochs, eventually reaching 91.08% at its best, but the collapse wastes training budget and indicates unstable optimization. The cause is the ∼27-step gradient chain through the Broyden solver: a perturbation in the solver trajectory at early steps is amplified through 27 Jacobian products, which is the regime in which the directional pathology of Sec. 4.5 is most severe, pushing the parameters into an unstable region.

Solution. With dense supervision, the model passes through epoch 13 stably at 85.17% accuracy (Tab. 10). The intermediate losses anchor each solver iterate to a region of latent space where the classification loss is low; the solver trajectory cannot wander into unstable regions because it is constrained at multiple points by the auxiliary losses. Beyond stability, DICE achieves **91.65%** top-1 accuracy, surpassing the baseline's 91.08% by +0.57 pp (Tab. 9). It reaches 90% accuracy at epoch 32, compared to epoch 40 for the baseline, which is 20% faster convergence.

B.3 Constraining Intermediate States via a Shared Readout

Prior work has attempted to regularize intermediate representations, but with indirect or costly mechanisms. Noisin (Dieng et al., 2018) injects noise into hidden states and maximizes expected likelihood, encouraging smoothness but not semantic consistency. Professor Forcing (Lamb et al., 2016) enforces distributional alignment between training and generation via an adversarial discriminator, and State-Reification (Lamb et al., 2019) introduces a denoising autoencoder that projects hidden states onto a learned manifold. The latter two approaches require training auxiliary networks, increasing both computational complexity and implementation burden.

Empirical comparison. We evaluate this design on the bAbI 10k benchmark with the Universal Transformer (Tab. 11). Noisin provides a small gain over the baseline (+1.5 pp). DICE achieves a substantially larger improvement (+4.0 pp) without introducing stochasticity or auxiliary networks.

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	+1.4 pp
DICE (ours)	**0.9369**	+4.0 pp

Discussion. The key distinction is that prior methods attempt to shape the *distribution* of hidden states (via noise or auxiliary networks), whereas DICE directly constrains their *function*: every state must be decodable by the same head. This shifts the problem from regularization to representation design. The resulting model learns to organize its internal computation as a sequence of progressively improving, task-aligned states, eliminating uninformative intermediate states without sacrificing flexibility or adding training overhead.

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

B.4 Which Weight Schedule, and Why

The optimal weight schedule depends on which problem dominates for a given system (Tab. 12). **Exponential** schedules suit systems where Problem 1 dominates (long K, early iterations unreliable). This is consistent with the DEQ result in Sec. B.2, where exponential weighting outperforms linear and uniform schedules. **Linear** schedules suit Problem 2 (extrapolation requires progressive improvement). This result aligns with the prefix-sum result in Sec. 4.2, where a linear schedule yields perfect extrapolation up to 128 bits. **Uniform** schedules suit short-K systems where each cycle should contribute comparably.

Training longer. We ran the maze baseline for 7,000 epochs (75% longer than 4,000 epochs). The best accuracy was 79.5%, which is lower compared to DICE's 81.05% obtained in fewer epochs. The result suggests that longer training does not close the gap.

Gradient clipping on DEQ. We tested gradient clipping at norms $\{1.0, 5.0, 10.0\}$. The epoch 13 collapse still occurs with all clip values, though less severely (accuracy dips to $\sim$55% instead of 39.65%). DICE eliminates the collapse entirely. Gradient clipping treats the symptom (large gradient norms); DICE treats the cause (noisy long-range gradient signal).

C Experimental Configuration

Compute. All experiments were run on NVIDIA GeForce RTX 2080 Ti GPUs, except the Recursive Transformer, which was trained on NVIDIA A100 GPUs.

Constraining intermediate states (Sec. B.3). Noisin standard deviation 0.25. Training: 100 epochs, evaluation every epoch, early-stopping patience 20, 5 bAbI tasks, 3 seeds per task, on the 10k dataset. We load the best checkpoint and report test-set accuracy.

bAbI configuration (Sec. B.1). We use the exponential schedule with $\alpha = 0.5$.

DEQ (Sec. B.2). We report results with the exponential schedule and $K = 5$.

Adaptive halting on TRM (Sec. 4.3). We use $\epsilon = 5 \times 10^{-3}$ and patience $m = 1$.

D Theoretical evidence

D.1 Relative Gradient Noise under Endpoint and Dense Supervision

Motivation. Consider an iterative neural computation

$$z_{k+1} = f_\theta(z_k), \qquad k = 0, \ldots, K-1,$$

trained with a loss applied either only at the endpoint z_K, or densely at intermediate iterations. 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 the gradient direction and magnitude. Dense supervision shortens these backpropagation paths by injecting losses at intermediate iterations.

We isolate this effect in a simple scalar model of multiplicative gradient noise. Let the contribution of a length-L backpropagation path contain a multiplicative factor

$$P_L = \prod_{j=1}^{L} A_j,$$

where $A_1, \dots, A_L$ are independent random variables representing the random Jacobian factors encountered along the path. We measure the noise level of this gradient factor by its relative variance,

$$\mathrm{RV}(P_L) := \frac{\mathrm{Var}(P_L)}{\mathbb{E}[P_L]^2}.$$

Proposition 1 (Exponential growth of relative gradient noise)**.** *Let $A_1, \dots, A_L$ be independent random variables satisfying*

$$\mathbb{E}[A_j] = \mu, \qquad \mathrm{Var}(A_j) = \sigma^2, \qquad \mu \neq 0,$$

and let $P_L = \prod_{j=1}^{L} A_j$. Then the relative variance of the length-L gradient factor is

$$\mathrm{RV}(P_L) = \left(1 + \frac{\sigma^2}{\mu^2}\right)^L - 1.$$

In particular, the relative noise of the endpoint gradient grows exponentially with the number of Jacobian factors L.

Proof. Since the variables $A_1, \dots, A_L$ are independent,

$$\mathbb{E}[P_L] = \mathbb{E}\left[\prod_{j=1}^{L} A_j\right] = \prod_{j=1}^{L} \mathbb{E}[A_j] = \mu^L.$$

Similarly,

$$\mathbb{E}[P_L^2] = \mathbb{E}\left[\prod_{j=1}^{L} A_j^2\right] = \prod_{j=1}^{L} \mathbb{E}[A_j^2].$$

Using

$$\mathbb{E}[A_j^2] = \mathrm{Var}(A_j) + \mathbb{E}[A_j]^2 = \sigma^2 + \mu^2,$$

we obtain

$$\mathbb{E}[P_L^2] = (\mu^2 + \sigma^2)^L.$$

Therefore,

$$\mathrm{Var}(P_L) = \mathbb{E}[P_L^2] - \mathbb{E}[P_L]^2 = (\mu^2 + \sigma^2)^L - \mu^{2L}.$$

Dividing by $\mathbb{E}[P_L]^2 = \mu^{2L}$ yields

$$\frac{\mathrm{Var}(P_L)}{\mathbb{E}[P_L]^2} = \frac{(\mu^2 + \sigma^2)^L - \mu^{2L}}{\mu^{2L}} = \left(\frac{\mu^2 + \sigma^2}{\mu^2}\right)^L - 1 = \left(1 + \frac{\sigma^2}{\mu^2}\right)^L - 1,$$

which proves the claim. □

Implication for endpoint supervision. For endpoint supervision, the gradient contribution to an early step t contains a product of approximately $K - t$ Jacobian factors. By Proposition 1, its relative noise scales as

$$\mathrm{RV}_{\mathrm{end}}(t) = \left(1 + \frac{\sigma^2}{\mu^2}\right)^{K-t} - 1.$$

The farther the step is from the endpoint, the larger the relative gradient noise.

Implication for dense supervision. With dense supervision, the same transition receives auxiliary gradient signals from intermediate losses. A loss applied at step m contributes a path of length $m - t$, rather than $K - t$. Its relative noise is therefore

$$\mathrm{RV}_{\mathrm{dense}}(t, m) = \left(1 + \frac{\sigma^2}{\mu^2}\right)^{m-t} - 1, \qquad m < K.$$

Since $m - t < K - t$, we have

$$\mathrm{RV}_{\mathrm{dense}}(t, m) < \mathrm{RV}_{\mathrm{end}}(t)$$

whenever $\sigma^2 > 0$. Thus, intermediate losses provide gradient signals with strictly smaller relative multiplicative noise than the endpoint loss.

Conclusion. This toy calculation shows that the noise in a backpropagated gradient signal can grow exponentially with the length of the Jacobian chain. Endpoint supervision forces early iterations to learn from the noisiest, longest-path gradient. Dense supervision replaces this single long-range signal with a collection of shorter-path signals, each having exponentially smaller relative noise. Therefore, dense supervision should provide cleaner and more stable optimization signals, especially for early iterations and long-horizon iterative models.

D.2 Prediction-Space Stability under Dense Supervision

Motivation. The adaptive-halting rule in Sec. 3.2 stops when consecutive predictions become stable in probability space:

$$\|\mathrm{softmax}(\hat{y}_{k+1}) - \mathrm{softmax}(\hat{y}_k)\|_1 \le \epsilon.$$

We now explain why dense supervision makes this criterion meaningful. The goal is not to prove that the latent states z_k converge to a fixed point. Rather, we show that when the DICE loss is small, all supervised predictions are close to the same target distribution, and therefore close to each other. Thus DICE encourages approximate fixed-point behavior in the same prediction space used by adaptive halting.

For classification tasks, let

$$p_k(x) = \mathrm{softmax}(g_\theta(z_k(x)))$$

be the predictive distribution at iteration k, and let $e_{\mathbf{y}(x)}$ be the one-hot target distribution. For sequence or grid-output tasks, all losses and distances below are averaged over output positions. We use the standard cross-entropy loss

$$\ell_{\mathrm{CE}}(p_k, \mathbf{y}) = -\log p_k[\mathbf{y}].$$

The DICE objective from Eq. (3) can be written as

$$L_{\mathrm{DICE}}(\theta) = \sum_{k=1}^{K} \lambda_k \mathbb{E}_x \left[\ell_{\mathrm{CE}}(p_k(x), \mathbf{y}(x))\right],$$

where $\lambda_K = 1$ for the endpoint loss and $\lambda_k = \alpha w_k$ for $k < K$. Hence every supervised iteration has a strictly positive weight $\lambda_k > 0$.

Proposition 2 (Dense supervision implies prediction-space stability)**.** *Assume that the DICE objective satisfies*

$$L_{\mathrm{DICE}}(\theta) \le \delta.$$

Then every supervised prediction is close to the target distribution:

$$\mathbb{E}_x \left[\|p_k(x) - e_{\mathbf{y}(x)}\|_1\right] \le \frac{2\delta}{\lambda_k}.$$

Moreover, consecutive predictions are stable:

$$\mathbb{E}_x \left[\|p_{k+1}(x) - p_k(x)\|_1\right] \le 2\delta \left(\frac{1}{\lambda_{k+1}} + \frac{1}{\lambda_k}\right).$$

Thus, low dense loss directly controls the same ℓ_1 prediction drift used by adaptive halting.

Proof. For a one-hot target e_y, we have

$$\|p - e_y\|_1 = |1 - p[y]| + \sum_{c \neq y} p[c] = 2(1 - p[y]).$$

Since $-\log u \geq 1 - u$ for $u \in (0, 1]$, it follows that

$$\|p - e_y\|_1 \leq 2\, \ell_{\mathrm{CE}}(p, y).$$

Because all terms in L_{DICE} are nonnegative, for every supervised iteration k,

$$\lambda_k \mathbb{E}_x \left[\ell_{\mathrm{CE}}(p_k(x), \mathbf{y}(x))\right] \leq L_{\mathrm{DICE}}(\theta) \leq \delta.$$

Therefore,

$$\mathbb{E}_x \left[\ell_{\mathrm{CE}}(p_k(x), \mathbf{y}(x))\right] \leq \frac{\delta}{\lambda_k}.$$

Combining this with the previous inequality gives

$$\mathbb{E}_x \left[\|p_k(x) - e_{\mathbf{y}(x)}\|_1\right] \leq 2\mathbb{E}_x \left[\ell_{\mathrm{CE}}(p_k(x), \mathbf{y}(x))\right] \leq \frac{2\delta}{\lambda_k}.$$

Now consider the distance between consecutive predictions. By the triangle inequality,

$$\|p_{k+1}(x) - p_k(x)\|_1 \leq \|p_{k+1}(x) - e_{\mathbf{y}(x)}\|_1 + \|p_k(x) - e_{\mathbf{y}(x)}\|_1.$$

Taking expectations and applying the bound above to iterations k and $k + 1$, we obtain

$$\mathbb{E}_x \left[\|p_{k+1}(x) - p_k(x)\|_1\right] \leq \frac{2\delta}{\lambda_{k+1}} + \frac{2\delta}{\lambda_k} = 2\delta \left(\frac{1}{\lambda_{k+1}} + \frac{1}{\lambda_k}\right).$$

This proves the claim. □

Implication for adaptive halting. Proposition 2 controls exactly the quantity used by the adaptive-halting rule in Sec. 3.2: the ℓ_1 distance between consecutive predictive distributions. Dense supervision trains every iteration to predict the same target through the same readout head. Therefore, when the dense objective is small, the prediction trajectory must become stable in probability space. Adaptive Stability Halting uses the empirical version of this stability signal at inference time and stops once the trajectory has reached an approximate prediction-space fixed point.

Why endpoint supervision cannot give this guarantee. Endpoint supervision minimizes only

$$L_{\mathrm{end}}(\theta) = \mathbb{E}_x \left[\ell_{\mathrm{CE}}(p_K(x), \mathbf{y}(x))\right].$$

It controls p_K, but gives no bound on p_k for $k < K$, and therefore no bound on

$$\|p_{k+1}(x) - p_k(x)\|_1.$$

A model can therefore learn a fixed-horizon strategy: output arbitrary predictions at intermediate iterations and produce the correct answer only at the trained endpoint. Such a model can have low endpoint loss while being unsuitable for anytime prediction, adaptive halting, or test-time extrapolation to more iterations. Dense supervision rules out this degeneracy by requiring every supervised iteration to be decodable into the same target.

Conclusion. This result formalizes the prediction-space effect of DICE. Dense supervision does not require the hidden states themselves to converge, but it does constrain their decoded predictions to approach the same target distribution. As a result, the output trajectory becomes stable in the same probability space monitored by adaptive halting. Endpoint supervision cannot provide this guarantee because it only constrains the final prediction.