DART: DISTRIBUTIONAL ADVERSARIAL RECURRENT TRAINING FOR ALGORITHM LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent reasoning models (RRMs) can solve structured problems, achieving *easy-to-hard* generalization through iterative computation in hidden space. These models are typically trained with instance-level supervision, which we hypothesize becomes increasingly brittle under easy-to-hard stressors that expand the candidate output space or leave more completions locally plausible. We propose **Distributional Adversarial Recurrent Training (DART)**, a training framework that replaces single-point supervision with a local target distribution around the ground-truth solution and aligns model outputs with this distribution through an adversarial objective. DART provides a richer learning signal and encourages more stable iterative trajectories toward valid solutions. When evaluated on Maze, Chess, and masked Sudoku with multiple RRMs, including Deep Thinking Systems and Tiny Recursive Models, DART improves solution quality, stability, and robustness under the evaluated distribution shifts. Comparisons with label smoothing, Gaussian softened targets, and progressive training show that DART is not explained by target softening alone and is complementary to training schemes that stabilize long-horizon recurrence. These results identify DART as a promising approach for improving robustness across the evaluated recurrent reasoning models.

1 INTRODUCTION

Logical reasoning has become a central benchmark for evaluating the reasoning capabilities of intelligent systems Chollet (2019); Chollet et al. (2025). In large language model literature, this goal is often pursued through *explicit reasoning* and *inference-time search*. LLM based approaches show that increasing inference-time computation—through sampling Brown et al. (2024), branching Wei et al. (2022); Yao et al. (2023), and verification Shao et al. (2025)—can substantially improve performance on complex reasoning tasks. These methods rely on large scale models to generate explicit reasoning traces and thus inherit the limitations of autoregressive models Bachmann & Nagarajan (2025).

Parallel to these developments, another line of work explores *implicit reasoning via iterative computation in hidden space*. Recurrent reasoning models (RRMs), such as *Tiny Recursive Models* (TRM) Jolicoeur-Martineau (2025) and *Hierarchical Reasoning Models* (HRM) Wang et al. (2025), can solve challenging logical reasoning tasks by repeatedly applying a fixed computation block, effectively scaling depth at test time. These results suggest that reasoning can emerge from *iterative transformations of hidden states* without explicitly generating intermediate reasoning traces. However, these models are primarily designed and trained on in-distribution datasets, and their performance on out-of-distribution data has not been systematically examined.

Beyond logical reasoning and pattern matching, prior work Kaiser & Sutskever (2016); Graves et al. (2014) suggests that RRMs can learn *algorithms* from data and exhibit systematic generalization capability. Extending this idea, Schwarzschild et al. Schwarzschild et al. (2021) introduced the *Deep Thinking System* (DTS), an RRM trained on relatively *simple* problem instances that can generalize to substantially *harder* ones at test time by increasing the number of recurrent reasoning steps.[1] This

[1] A problem instance is considered harder than another instance if its shortest solution is longer than that of the other. We can create harder problems by increasing the problem sizes or decreasing the amount of information in problem statements. For example, a larger maze requires a longer shortest path to escape than a smaller maze, a 50% masked Sudoku requires more steps to solve than a 30% masked Sudoku.

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.

phenomenon, referred to as *easy-to-hard generalization*, suggests that models capable of genuine reasoning do not merely interpolate within the training distribution, but instead learn algorithmic procedures that can be systematically applied to more complex inputs.

Despite these promising results, RRMs are still typically optimized at the instance level: for each logical input, the model is trained to match a single target solution. We hypothesize that this creates a challenge for *easy-to-hard* generalization because, as problem complexity increases, point-wise supervision must guide recurrent updates through a much larger candidate output space while observing only one reference solution. In this paper, we use candidate-space expansion and Sudoku candidate entropy as operational stressors rather than direct estimates of the valid/invalid solution ratio. Under these stressors, iterative reasoning models often exhibit unstable performance on difficult inputs, requiring substantially more iterations and still failing to converge to correct solutions.

To mitigate this limitation of instance-level training, we introduce **Distributional Adversarial Recurrent Training (DART)** for Reasoning Models. Inspired by latent-variable methods such as VAEs Kingma & Welling (2022), our approach replaces single-point supervision with a target distribution that defines a neighborhood around the correct solution by injecting random noise during training. This local soft target preserves the original solution under $\arg\max$ decoding, but it should not be interpreted as enumerating multiple distinct discrete solutions. Instead, it enlarges the supervised neighborhood around the observed solution in the probability simplex, making the learning signal less brittle than a single one-hot point.

To align generated reasoning states with this local target distribution, we introduce an **adversarial training objective** inspired by Wasserstein GANs (WGANs) Arjovsky et al. (2017). The reasoning model acts as a generator that produces candidate solution distributions, while a critic network learns to distinguish generated reasoning states from samples drawn from the soft target distribution. By combining supervised reasoning loss with conditional joint matching, our method encourages correct predictions that remain close to an input-conditioned target neighborhood, yielding induced trajectory stability on complex instances. Unlike generic noisy-label or GAN-style auxiliary losses, DART matches problem-solution pairs rather than unconditional output realism: the critic evaluates whether recurrent predictions remain near input-compatible successful traces.

DART is generally applicable to different RRM architectures. We evaluate it on both the Deep Thinking System (DTS) and the Tiny Recursive Model (TRM). For DTS, we consider two complementary easy-to-hard reasoning tasks: two-dimensional maze solving and chess puzzles. Across these tasks, DART improves DTS by increasing accuracy and yielding more stable performance across reasoning iterations. Following Jolicoeur-Martineau (2025); Wang et al. (2025), we evaluate DART on TRM for Sudoku. Because TRM is originally trained on hard Sudoku instances with a fixed masking level in both training and evaluation, it is not naturally formulated as an easy-to-hard benchmark. To study easy-to-hard generalization, we vary the Sudoku masking ratio, train the model on puzzles with low mask ratios, and evaluate on more heavily masked ones. Since heavily masked puzzles may admit multiple completions, we evaluate Sudoku primarily by whether the prediction is a valid Sudoku board that preserves the observed clues, with exact match to the reference solution used only as a stricter diagnostic. Under this setting, DART also improves TRM's reasoning capability on more difficult Sudoku instances.

Overall, we summarize our contribution as follows:

- We study *easy-to-hard* generalization under operational candidate-space stressors. Rather than claiming to measure the full valid/invalid solution ratio, we use controlled maze scaling, Sudoku masking, and a local Sudoku candidate-entropy diagnostic to test when point-wise supervision becomes brittle.
- We reformulate recurrent reasoning as conditional joint matching between problem inputs and local solution-neighborhood samples, and propose **distributional adversarial recurrent training (DART)** to induce stable trajectories near input-compatible solution traces.
- We show that DART improves the performance of multiple RRMs across a range of logical reasoning tasks, yielding more stable reasoning outcomes across iterations and stronger valid-solution consistency on masked Sudoku. We further compare against label smoothing, Gaussian softened targets without a critic, and progressive training baselines to separate target softening, conditional critic training, and long-horizon stabilization.

2 Related Work

Easy-to-hard generalization and algorithm learning. Sun et al. Sun et al. (2024) pose a fundamental question: if models are trained only on human-labeled data, how can they surpass human-level performance or solve problems that are harder than those seen during training? This challenge, termed *easy-to-hard generalization*, has motivated research on enabling deep learning models to learn underlying algorithms for logical tasks and generalize to larger, more complex instances. Recurrent neural networks (RNNs) and related architectures are well suited to this setting because they can flexibly adjust computational depth to input complexity Gers & Schmidhuber (2001). Graves et al. Graves et al. (2014) propose the Neural Turing Machine (NTM), which mimics key behaviors of classical computers, enabling algorithm induction. The NTM can learn procedures such as sorting and copying and generalize to input sequences substantially longer than those seen during training. Neural GPU Kaiser & Sutskever (2016) can learn arithmetic procedures such as addition and multiplication and generalize to arbitrary-length sequences. Deep Thinking systems Schwarzschild et al. (2021) extend this direction by iteratively refining intermediate representations through recurrent blocks. Original DTSs suffer from the *overthinking* problem where more reasoning iterations lead to worse performance Bansal et al. (2022). To mitigate this issue, Bansal et al. Bansal et al. (2022) introduced both the recall architecture, which preserves input information across recurrent steps, and a progressive training objective that exposes the model to longer reasoning horizons during training. Subsequent work further showed that progressive loss alone does not fully guarantee stable recurrence in compact DTS models; for small hidden representations, overthinking can persist unless the recurrent dynamics are more directly constrained, for example through Lipschitz-controlled architectures Bear et al. (2024). We use the recall architecture as the backbone in all DTS experiments and explicitly compare against progressive training in Section 5. More recent RRMs such as Tiny Recursive Models (TRM) Jolicoeur-Martineau (2025) and Hierarchical Reasoning Models (HRM) Wang et al. (2025) achieved strong results on challenging logical tasks while using far fewer parameters than LLMs. HRM uses a hierarchical architecture that combines slow and fast thinking to capture both local detail and global structure, whereas TRM simplifies this design with fewer parameters and emphasizes iterative refinement of model outputs.

Adaptive computation and iterative inference. Adjusting computational depth based on input complexity has been studied extensively in adaptive computation. Graves et al. (2016) propose Adaptive Computation Time (ACT) Graves (2016), which enables recurrent models to halt once a learned stopping probability reaches a threshold. Universal Transformers Dehghani et al. (2018) and Adaptive Recurrent Vision Veerabadran et al. (2023) applied this idea to a range of settings. PonderNet Banino et al. (2021) improves the halting mechanism in ACT and outperforms ACT on logical extrapolation tasks such as prefix sums. More broadly, Bansal et al. (2022); Kaya et al. (2018); Eyzaguirre & Soto (2020) studied models with variable numbers of test-time iterations or layers. These methods are complementary to DART: they decide how much computation to allocate or how to train models over longer computation horizons, whereas DART changes the supervision signal used at each training example.

Adversarial guided training. Adversarial training has been widely adopted in generative modeling, beginning with Generative Adversarial Networks (GANs) Goodfellow et al. (2014) and followed by advances such as WGAN Arjovsky et al. (2017). Adversarial objectives have also been used to guide structured learning. SADM Yang et al. (2024) allows diffusion models to capture semantic constraints among generated samples during denoising through adversarial training and batch-level similarity matrices. AdvBN Shu et al. (2020) improves robustness by perturbing feature statistics rather than input pixels. AdvBN generates worst-case perturbations of feature distributions (mean and variance) to simulate challenging style shifts and can be combined with existing data augmentation techniques to further improve performance.

Our contribution vs prior work. In this work, we study easy-to-hard generalization under candidate-space stressors where point-wise supervision must remain stable as the output space grows or the input leaves more completions locally plausible. Viewing logical reasoning as conditional generation, we use adversarial training to match input-solution pairs against a local neighborhood around the reference solution. Based on this perspective, we propose DART as a method for inducing more stable iterative reasoning in RRMs. DART does not modify the recurrent architecture or learn

a halting policy. Instead, it replaces a single one-hot target with a local soft target distribution and uses an input-conditioned critic to match generated input-solution pairs with that local target neighborhood. This target distribution is a Gaussian-softened neighborhood around a reference solution, not a sampler over all valid completions. Unlike label smoothing, DART does not simply reduce confidence by assigning mass uniformly to incorrect classes. Unlike noisy-label regularization, the perturbation is paired with a critic that evaluates compatibility with the problem instance. Unlike standard GAN-style losses for structured prediction, DART targets recurrent reasoning dynamics: its goal is not output realism alone, but inducing predictions to remain near input-compatible solution traces under distribution shift.

3 PRELIMINARY

We consider structured reasoning tasks that map an input instance x to a discrete solution y^*, such as maze paths, chess moves, and Sudoku grids. A recurrent reasoning model (RRM) produces a candidate solution through iterative computation rather than a single feed-forward pass. In a Deep Thinking System (DTS) Schwarzschild et al. (2021); Bansal et al. (2022), a hidden solution representation is repeatedly refined before an output head predicts $\hat{y}$. In a Tiny Recursive Model (TRM) Jolicoeur-Martineau (2025), recursive low-level and high-level states perform repeated draft-check-refine updates before decoding the final answer.

Both families are usually trained with instance-level supervision,

$$\mathcal{L}_{sup} = \ell(\hat{y}, y^*), \tag{1}$$

where ℓ is cross-entropy or mean squared error. This treats the correct solution as a single deterministic target. DART keeps the same recurrent backbones but replaces the brittle one-point target with a local target distribution and an input-conditioned critic signal for conditional joint matching. Appendix A gives the full DTS/TRM formulation used in our experiments.

4 PROPOSED METHOD

This section presents the details of our proposed **DART** framework. We show how to reformulate a reasoning problem as a local distributional supervision problem (Sec. 4.1) and how to align the model's outputs with this reference-conditioned neighborhood (Sec. 4.2). DART is based on the principle that structured reasoning targets should not be treated as isolated points in output space. Under easy-to-hard shift, recurrent models can pass near valid solutions but drift into invalid configurations as computation continues. We therefore train the model to match a local, input-conditioned neighborhood around the observed reference solution so that recurrent updates are encouraged to stay near feasible solution traces rather than only hit a one-hot target at a single iteration. This neighborhood is a tractable proxy for local solution structure; it is not an empirical distribution over all valid completions.

4.1 SOFT SOLUTION DISTRIBUTION

We consider discrete logical reasoning problems with input x and target solution y^* such that $x, y^* \in \{c_0, c_1, \ldots, c_k\}^{m \times n}$, where c_0 denotes a padding token and $c_1, \ldots, c_k$ denote valid element values. The padding token allows us to handle problems where input and output have different sizes. To transform a solution to a distribution, we first represent each solution element as a one-hot vector $v_{i,j} = \text{onehot}(y^*_{i,j}) \in \{0,1\}^{k+1}$. The solution y^* is represented as a 3D tensor $v^* \in \{0,1\}^{m \times n \times (k+1)}$. A reasoning model produces output $\hat{y} \in [0,1]^{m \times n \times (k+1)}$ where each vector $y_{i,j} \in [0,1]^{k+1}$ is a probability vector. The discrete solution is obtained by applying the *argmax* operator to $\hat{y}$.

At each training iteration, Gaussian noise is added to each one-hot vector to create a distribution around that vector:

$$\tilde{v}_{i,j} = v^*_{i,j} + \epsilon_{i,j}, \quad \epsilon_{i,j} \sim \mathcal{N}(0, \sigma^2 I_{k+1}). \tag{2}$$

To preserve the original solution under *argmax* decoding, we restrict $\epsilon \in (-0.5, 0.5)$ and normalize the perturbed vector $\tilde{v}_{i,j}$ with softmax to produce vector $\hat{v}_{i,j}$ of the same form as $\hat{y}_{i,j}$. After

this operation, the target representation $\hat{v}$ and the model's output $\hat{y}$ lie in the same probability simplex, allowing generated predictions to be aligned directly with a local soft neighborhood around the observed solution. This construction preserves the original discrete target under $\arg\max$; it is not intended to represent all semantically distinct valid solutions when a problem admits multiple completions.

4.2 Adversarial Solution Alignment

To align generated solutions with the local target neighborhood, we introduce an adversarial objective inspired by Wasserstein GANs. Let $P_{\text{gen}}(\hat{y} \mid x)$ be the model solution distribution, and let $P_{\text{ref}}(\hat{v} \mid x, y^*)$ be the Gaussian-softened proxy around the reference solution. DART matches the conditional joint distributions $P_{\text{data}}(x, y^*)P_{\text{ref}}(\hat{v} \mid x, y^*)$ and $P_{\text{data}}(x, y^*)P_{\text{gen}}(\hat{y} \mid x)$ using a critic on (x, s). In implementation, P_{gen} is the empirical joint distribution over minibatch pairs $(x, \hat{y})$ induced by the current reasoning model, and the expectations below are estimated from minibatch samples. We use P_{ref} only as a local compatibility surrogate, not as a sampler over all valid discrete completions.

4.2.1 Reasoning Critic Network

A reasoning critic network C_ψ with parameters ψ assigns higher scores to softened target solutions and lower scores to generated ones. For each problem instance, the critic receives a concatenated input consisting of the original problem representation x and a candidate soft solution s, where s is either a generated solution $\hat{y}$ from the reasoning model or a softened target solution $\hat{v}$ sampled from the local target distribution. Its output is a scalar score that measures how well the candidate solution matches the input problem and conforms to the local target structure. In our implementation, $\hat{y}$ is the model output from the final recurrent iteration used during training. This design keeps DART model-agnostic: the adversarial objective can be attached to RRMs with different recurrence structures, including nested recurrent models such as TRM, without modifying their internal states or adding losses at architecture-specific intermediate steps. It also aligns the critic with the solution that is decoded at inference time, while leaving inference cost unchanged because the critic is discarded after training. Formally, the critic input is defined as $z = [x, s], s \in \{\hat{y}, \hat{v}\}$, where $[\cdot, \cdot]$ denotes concatenation along the feature dimension. To keep the overall framework simple and stable, we construct the critic with the same backbone architecture as the reasoning model. In other words, the critic reuses the same type of projection layer and recurrent reasoning block as the generator, and differs only in the final head, which outputs a scalar score rather than a structured solution tensor. Specifically, the critic is defined as follows:

$$\xi_0 = p_c(z), \quad \xi_{t+1} = r_c(\xi_t), \quad C_\psi(x, s) = q_c(\xi_T), \tag{3}$$

where p_c is an input projection layer, r_c is a recurrent reasoning module, and q_c maps the final hidden representation to a scalar score. Because the critic shares the same architectural form as the generator, it can evaluate candidate solutions using a compatible reasoning process while remaining lightweight to integrate into existing recurrent reasoning models. Intuitively, this allows the critic to learn a compatibility score between the input instance and the reference-centered soft solution neighborhood, while preserving architectural alignment between the two networks. Our ablation study examines how the generator behaves when the critic is implemented as either a recurrent model or a feedforward model.

4.2.2 Training Objective

Critic optimization The critic is trained to maximize the Wasserstein distance between the two distributions:

$$\mathcal{L}_{critic} = -\Big(\mathbb{E}_{\hat{v}\sim P_{\text{ref}}}[C_\psi(x, \hat{v})] - \mathbb{E}_{\hat{y}\sim P_{\text{gen}}}[C_\psi(x, \hat{y})]\Big) \tag{4}$$

subject to a Lipschitz constraint on C_ψ. We enforce this constraint with critic weight clipping, following the original WGAN formulation. During this step, only the critic parameters ψ are updated.

Reasoning model optimization The reasoning model is trained with both supervised learning and the conditional critic objective. We define the adversarial generator loss as

$$\mathcal{L}_{adv} = -\mathbb{E}_{\hat{y}\sim P_{\text{gen}}}[C_\psi(x, \hat{y})]. \tag{5}$$

This loss is optimized jointly with the supervised loss using a convex weighting:

$$\mathcal{L}_{total} = (1 - \lambda)\mathcal{L}_{sup} + \lambda\mathcal{L}_{adv}. \tag{6}$$

Unless otherwise stated, we use $\lambda = 0.5$ in all DART experiments. We do not tune λ per task, so the reported results use a single fixed weighting across Maze, Chess, and Sudoku.

During this step, the reasoning model parameters θ are updated while the critic parameters ψ remain fixed. The full alternating training procedure is provided in Appendix A.2.

5 EXPERIMENTS

5.1 DATASETS, MODEL ARCHITECTURE AND TRAINING

Following Schwarzschild et al. Schwarzschild et al. (2021), we evaluate DART on easy-to-hard Maze and Chess benchmarks for Deep Thinking Systems (DTS). We also reformulate Sudoku Wang et al. (2025) as a masking-based easy-to-hard benchmark and use it to evaluate DART on Tiny Recursive Models (TRM) under partial observability. Unless otherwise stated, reported means and standard deviations are computed over at least three random seeds; the main Maze CE/DART scaling comparison uses five independent seeds as described in Appendix B.1. Detailed dataset descriptions, architectures, and hyperparameters are provided in Appendix C.1.

5.2 RESULTS

We organize the empirical evidence around two primary stress tests. Maze asks whether DART regularizes recurrent reasoning as spatial extrapolation enlarges the output space that point-wise supervision must control. Sudoku asks the same question in a different architecture and failure mode: TRM must complete a partially observed constraint-satisfaction problem as more input cells are removed. Chess is reported as a supporting case for late-iteration collapse and progressive training in Appendix B.5. Table 1 summarizes the main results and gives a compact operational proxy for candidate-space expansion behind these shifts.

Table 1: Representative easy-to-hard results and candidate-space expansion proxy. Accuracy values are mean $\pm$ std. Maze CE/DART uses five independent seeds for each size, while Sudoku uses at least three seeds per method. The proxy column is a deterministic log-size of the unconstrained candidate output space, not a count of valid completions, invalid completions, or their ratio. Maze uses DTS final accuracy; Sudoku uses valid+clue consistency.

Task	Setting	$\log_{10} \lvert\mathcal{Y}_{\text{proxy}}\rvert$	CE	DART
Maze	13×13	50.9	99.74 ± 0.40	$\mathbf{99.99 \pm 0.01}$
Maze	31×31	289.3	79.55 ± 21.60	$\mathbf{99.88 \pm 0.17}$
Sudoku	30% mask	23.2	99.2 ± 0.0	$\mathbf{99.8 \pm 0.1}$
Sudoku	50% mask	38.6	51.8 ± 3.8	$\mathbf{64.8 \pm 3.4}$

Primary analysis: difficulty scaling in Maze. Maze exposes how point-wise supervision becomes brittle as the easy-to-hard shift grows. DTS models are trained on 9×9 mazes and evaluated at larger sizes. At 13×13, CE already reaches $99.90 \pm 0.09\%$ peak accuracy, so the shift is mild enough for standard supervision. At 31×31, however, CE reaches only $79.65 \pm 21.62\%$ peak accuracy over five seeds, confirming that the high variance is not an artifact of a small three-seed sample. DART remains near-saturated at the same harder scale, with $99.88 \pm 0.16\%$ peak accuracy and $99.88 \pm 0.17\%$ final accuracy (Figure 1; full table in Appendix B.1). We use this as an operational stress test: as the maze grows, the binary path-mask output space expands from roughly $10^{50.9}$ to $10^{289.3}$ possible masks (Table 1), and the local target distribution plus critic is associated with more stable recurrent predictions under this larger output space.

The Maze comparison also clarifies the distinction between DART and progressive training. Progressive loss mainly stabilizes the iteration horizon by exposing the model to longer recurrent rollouts during training, but it does not directly reshape the supervision signal around valid solution neighborhoods. This matters because prior work Bear et al. (2024) shows that compact DTS models can still overthink when the hidden representation is small, suggesting that recurrence stability may require

stronger constraints than schedule-based training alone. Consistent with this view, the available Progressive CE runs on 31×31 Maze reach only $75.01 \pm 23.61\%$ peak accuracy, far below DART's five-seed $99.88 \pm 0.16\%$. DART therefore addresses a different failure mode: it biases recurrent predictions toward input-compatible solution traces rather than only training the model to survive longer rollouts.

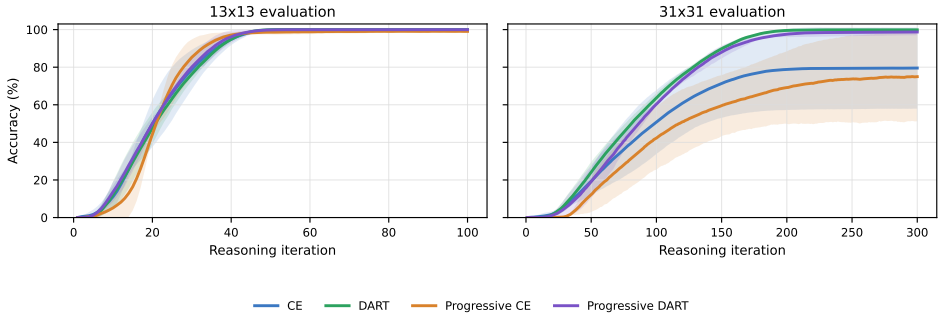


Figure 1: Maze accuracy as the evaluation size increases from 13×13 to 31×31. The CE failure mode is mostly absent at the smaller size but appears under harder spatial extrapolation; the shaded regions show variation across independent seeds.

Primary analysis: partial-observation scaling in Sudoku. Sudoku tests whether the same principle transfers beyond DTS. We train TRM with 20% masking and evaluate on 10,000 puzzles with 30%, 40%, and 50% masking. Because heavier masking can remove enough clues that the puzzle need not have a unique completion, we do not treat exact match to the original solved grid as the primary correctness metric. Instead, Figure 2 counts a prediction as correct when it is a valid Sudoku board and preserves the observed clues. Under this validity-and-clue metric, DART improves over CE by 0.5% at 30% masking, 4.8% at 40%, and 13.0% at 50%. This scaling pattern is important because heavier masking changes the structure of the problem, not only its raw accuracy: the unconstrained masked-cell assignment space grows from about $10^{23.2}$ candidates at 30% masking to $10^{38.6}$ at 50% masking (Table 1). This quantity is not a solution-space entropy or a valid/invalid ratio; it is an operational stressor indicating that point-wise supervision must select a compatible completion under increasing underdetermination.

To make this stressor more instance-specific, we compute a local ambiguity score from Sudoku constraints. For each blank cell i, let C_i be the set of digits locally compatible with the observed row, column, and box constraints, and define $A(x) = |\mathcal{B}|^{-1} \sum_{i \in \mathcal{B}} \log |C_i|$ over blank cells $\mathcal{B}$. We then split each test set into five candidate-entropy quantiles. Figure 3 shows that this local ambiguity increases both within each mask ratio and from 30% to 50% masking. DART's valid+clue gain over CE is negligible in the easiest 30% setting, grows from $+2.3$ to $+8.7$ points across ambiguity bins at 40% masking, and remains large across all 50% bins ($+11.4$–$+14.5$ points). This trend is consistent with DART helping most under local ambiguity stressors, without implying a measured causal valid/invalid ratio.

Constraint-level diagnostics support this interpretation. Table 2 reports final predictions at 50% masking, where the easy-to-hard shift is strongest. DART improves valid+clue consistency from 51.8% to 64.8%, exact match from 32.0% to 40.4%, masked-cell accuracy from 81.8% to 85.5%, and average violated units remain low at 1.1 per board. The component ablations show that both pieces matter: softened targets without a critic preserve observed clues but collapse to 0.0% valid+clue consistency, while a hard-target critic without soft/noise target construction reaches only 36.4% valid+clue consistency. We also test a direct CE+MSE objective, $(1-\lambda_{\text{mse}})\mathcal{L}_{sup} + \lambda_{\text{mse}} \|\hat{y} - \hat{v}\|_2^2$ with $\lambda_{\text{mse}} = 0.5$, which matches generated solutions to softened targets without a critic. It fails under the hardest shift, reaching only 24.5% valid+clue consistency and 69.8% masked-cell accuracy. Thus, the gain is not obtained by merely adding a pointwise distribution-matching penalty; the conditional critic provides a task-level compatibility signal that is especially important when the puzzle is highly underdetermined. These results show that the TRM gains are not merely extra correct cells; DART better anchors the recursive computation to the observed puzzle while preserving global Sudoku validity. This mirrors Maze at a structural level: in both settings, CE degrades under larger candidate-space stressors, while DART remains more stable and input-compatible. Appendix B.2 reports the same diagnostics for all masking ratios, and Appendix B.6 gives the broader component-ablation discussion.

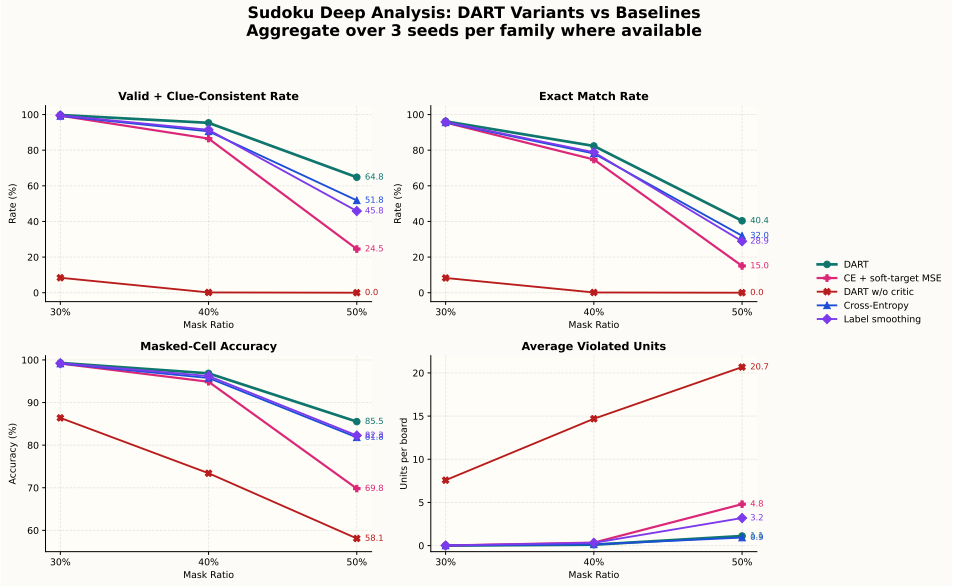


Figure 2: TRM-Sudoku deep analysis across masking ratios. DART is compared with CE, label smoothing, softened targets without a critic, and a direct CE+MSE objective that matches softened targets to generated solutions without a critic. All component families use at least three independent seeds.

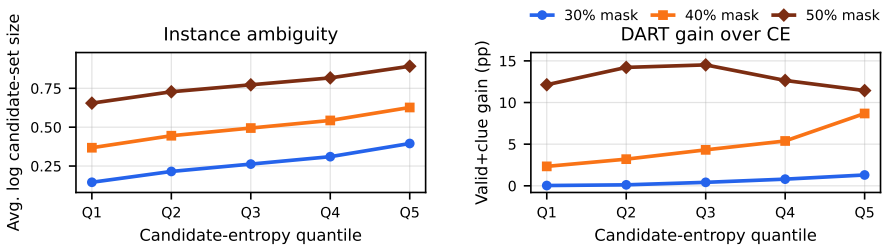


Figure 3: Sudoku ambiguity analysis. Left: average log candidate-set size over blank cells, using candidate sets induced by row/column/box constraints. Right: DART gain over CE on valid+clue consistency within the same ambiguity quantile.

Table 2: TRM-Sudoku component diagnostics at 50% masking. Values are means over at least three seeds per method.

Method	Exact match	Valid+clue consistent	Masked-cell acc.	Avg. violated units
CE	32.0	51.8	81.8	0.9
Label smoothing	28.9	45.8	82.3	3.2
DART w/o critic	0.0	0.0	58.1	20.7
CE+soft-target MSE	15.0	24.5	69.8	4.8
DART w/o soft/noise target	22.2	36.4	75.6	5.6
DART	**40.4**	**64.8**	**85.5**	1.1

We further test whether the hardest Sudoku setting can be solved simply by increasing TRM's iterative compute. Figure 4 sweeps the number of high-level cycles H from 3 to 6 at 50% masking. Extra cycles help only modestly: DART improves from 64.8% to 66.4% valid+clue consistency, while CE moves from 51.4% to 52.0%. The CE+soft-target MSE baseline remains nearly flat (24.5% to 25.0%), and the hard-target critic variant improves from 36.4% to 39.1%, but both remain far below full DART.

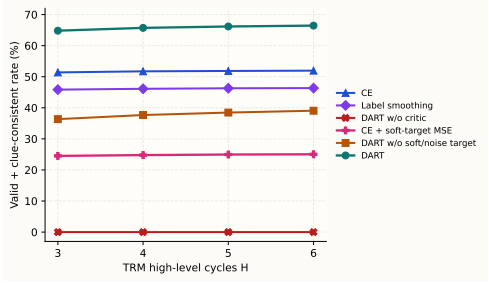


Figure 4: Effect of increasing TRM high-level cycles on Sudoku at 50% masking.

Thus, the mask-50 failure is not primarily a lack of recurrent refinement steps. The training objective determines whether additional TRM compute moves predictions toward input-compatible valid completions, and DART maintains the largest margin while benefiting most from the added cycles. We do not apply DTS-style progressive loss to TRM because TRM already contains nested low-level and high-level recurrence; a progressive horizon schedule is therefore not a natural drop-in baseline for this architecture. Appendix B.3 reports the full sweep.

6 Discussion

While RRMs with DART are highly effective on spatial algorithmic tasks such as Maze, their gains are more nuanced on games and constraint-satisfaction tasks such as Chess and Sudoku. Unlike Maze, whose difficulty scales mainly with input size and path length, these tasks are dominated by strategic depth, ambiguity, and combinatorial structure. Chess requires selecting an optimal move among many plausible candidates, while masked Sudoku involves an expanding space of possible completions that demands hypothesis generation, verification, and correction. This exposes a key limitation of current recurrent reasoning models. Although they can iteratively refine predictions, they lack explicit mechanisms for verification, structured exploration, and backtracking. DART improves extrapolation by expanding the local soft neighborhood around the observed target and inducing more stable reasoning trajectories, but it primarily mitigates instability rather than solving the underlying search problem. It also does not enumerate multiple semantically distinct valid discrete solutions; the perturbation preserves the observed target under $\arg\max$ and provides a smoother continuous supervision signal around that target. As a result, DART makes reasoning more reliable without fully eliminating the root causes of failure in complex tasks. Our candidate-space analysis is operational rather than exhaustive. We vary maze size and Sudoku masking to control the difficulty shift, and the Sudoku candidate-entropy diagnostic provides a local ambiguity proxy, but we still do not estimate the true number of valid completions, invalid candidates, their ratio, or solver-level difficulty for each instance. Developing such diagnostics would make the proposed stress-test framing more explanatory and is an important direction for future work. Our progressive-training experiments further show that long-horizon stability and easy-to-hard extrapolation are related but distinct. Progressive training is especially effective at removing late-iteration collapse on Chess, while DART is more important under the larger Maze difficulty shift. Combining distributional training with explicit curriculum, verification, or search mechanisms is therefore a natural next step. These findings suggest that improving training objectives alone may not be sufficient for problems that inherently require search and verification. A promising direction is to combine distributional training with stronger reasoning primitives, such as learned verification modules, scoring functions, or structured backtracking. More broadly, integrating distribution-guided learning with search-based reasoning may provide a more complete framework for tackling complex reasoning tasks.

7 Conclusion

We introduced **Distributional Adversarial Recurrent Training (DART)**, a training framework for recurrent reasoning models that replaces purely instance-level supervision with conditional joint matching between problem inputs and local neighborhoods around the ground-truth solution. By modeling logical reasoning as a process of progressively moving predictions toward an input-conditioned soft target neighborhood, DART provides a richer learning signal than standard cross-entropy and induces more stable iterative reasoning. Across Maze, Chess, and masked Sudoku, DART improves peak solution quality and robustness under distribution shift, with especially large gains when point-wise supervision becomes unstable. On Sudoku, this improvement is measured primarily by valid solutions that preserve the observed clues, rather than only by exact match to a reference completion. Additional comparisons with label smoothing, Gaussian softened targets, and progressive training show that target softening alone is task-dependent, while conditional critic training and progressive training address complementary failure modes. The CE+soft-target MSE ablation further shows that direct pointwise matching to softened targets is not sufficient to recover DART's Sudoku gains under severe masking. These benefits appear consistently across the evaluated DTS and TRM settings, suggesting that distribution-guided objectives are a promising principle for improving recurrent reasoning systems.

AI USE STATEMENT

In this work, we used generative AI tools to assist in interpreting and reformatting experimental logs into analytical text. We did not use generative AI tools for formulating mathematical claims, developing theoretical models, or generating synthetic datasets, and qualitative data analysis is not applicable to this work. Additionally, we used generative AI tools for drafting parts of the paper, editing to improve readability, preparing figures and tables, and LaTeX formatting. We reviewed all AI-assisted work by verifying generated text, code, figures, and numerical summaries directly against our raw experiment outputs. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction, 2025. URL https://arxiv.org/abs/2403.06963.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv: 2107.05407*, 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. *arXiv preprint arXiv: 2202.05826*, 2022.

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

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-agi-2: A new challenge for frontier ai reasoning systems. *arXiv preprint arXiv: 2505.11831*, 2025.

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

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv: 1807.03819*, 2018.

Cristobal Eyzaguirre and Alvaro Soto. Differentiable adaptive computation time for visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Felix Gers and E. Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *Neural Networks, IEEE Transactions on*, 12:1333 – 1340, 12 2001. doi: 10.1109/72.963769.

Ian Goodfellow et al. Generative adversarial nets. In *NeurIPS*, 2014.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv: 1603.08983*, 2016.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv: 1410.5401*, 2014.

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

Lukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.08228.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. *International Conference on Machine Learning*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

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 M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 6695–6706. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/3501672ebc68a5524629080e3ef60aef-Paper.pdf.

Zhihong Shao, Yuxiang Luo, Chengda Lu, Z. Z. Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. *arXiv preprint arXiv: 2511.22570*, 2025.

Manli Shu, Zuxuan Wu, Micah Goldblum, and T. Goldstein. Encoding robustness to image style via adversarial feature perturbations. *Neural Information Processing Systems*, 2020.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, S. Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. *Neural Information Processing Systems*, 2024. doi: 10.48550/arXiv.2403.09472.

Vijay Veerabadran, Srinivas Ravishankar, Yuan Tang, Ritik Raina, and Virginia de Sa. Adaptive recurrent vision performs zero-shot computation scaling to unseen difficulty levels. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/3a40e042c66e84659249f3254460c123-Abstract-Conference.html.

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 Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv: 2201.11903*, 2022.

Ling Yang, Haotian Qian, Zhilong Zhang, Jingwei Liu, and Bin Cui. Structure-guided adversarial training of diffusion models, 2024. URL https://arxiv.org/abs/2402.17563.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv: 2305.10601*, 2023.

A Additional Preliminaries and Training Procedure

A.1 Recurrent Reasoning Backbones

Deep Thinking Systems (DTSs) Schwarzschild et al. (2021) produce an entire solution representation and refine it iteratively. A DTS f_θ can be written as

$$\phi_0 = p(x), \quad \phi_{t+1} = r(\phi_t), \quad \hat{y} = h(\phi_T), \tag{7}$$

where p projects the input, r is a recurrent reasoning module, and h is an output head. To preserve the original problem throughout recurrence, we use the Recall variant Bansal et al. (2022),

$$\phi_{t+1} = r([\phi_t, x]). \tag{8}$$

Tiny Recursive Models (TRMs) Jolicoeur-Martineau (2025) refine a draft solution through nested low-level and high-level recurrent states inspired by the Hierarchical Reasoning Model Wang et al. (2025). Denoting the low-level state by z_t^ℓ and the high-level answer state by z_t^h, TRM can be written as

$$z_{t+1}^\ell = r_\ell(z_t^\ell, z_t^h, x), \quad z_{t+1}^h = r_h(z_{t+1}^\ell, z_t^h), \quad \hat{y} = h(z_T^h). \tag{9}$$

The low-level state acts as a workspace for local constraints, while the high-level state stores the current global hypothesis. TRM alternates between several low-level refinement steps and a high-level update, producing a repeated draft-check-refine computation.

A.2 DART Training Algorithm

Algorithm 1 Training Recursive Reasoning Models with Adversarial Alignment

Require: training data (x, y^*), reasoning model f_θ, critic C_ψ
Require: perturbation variance σ, adversarial weight λ
while not converged **do**
  Sample minibatch (x, y^*)
  $v_{i,j} \leftarrow \text{onehot}(y^*_{i,j})$
  $\tilde{v}_{i,j} \leftarrow v_{i,j} + \epsilon_{i,j}, \quad \epsilon_{i,j} \sim \mathcal{N}(0, \sigma^2 I)$
  $\hat{v}_{i,j} \leftarrow \text{softmax}(\tilde{v}_{i,j})$
  $\hat{y} \leftarrow f_\theta(x)$
  Compute critic loss

$$\mathcal{L}_{critic} = -\Big(\mathbb{E}_{y \sim P_{\text{real}}}[C_\psi(x, y)] - \mathbb{E}_{\hat{y} \sim P_{\text{gen}}}[C_\psi(x, \hat{y})]\Big)$$

  Update critic parameters ψ
  Compute $\mathcal{L}_{sup} = \ell(\hat{y}, y^*)$
  Compute $\mathcal{L}_{adv} = -\mathbb{E}_{\hat{y} \sim P_{\text{gen}}}[C_\psi(x, \hat{y})]$
  Update θ with $\mathcal{L}_{total} = (1 - \lambda)\mathcal{L}_{sup} + \lambda \mathcal{L}_{adv}$
end while

B Additional Experimental Results

B.1 Main Result Tables

Table 3: Maze scaling results for standard and progressive training (mean $\pm$ std over independent seeds). Standard CE and DART use five seeds for both sizes; progressive baselines are reported over the completed independent runs.

Method	13×13 Peak	13×13 Final (100)	31×31 Peak	31×31 Final (300)
CE	99.90 ± 0.09	99.74 ± 0.40	79.65 ± 21.62	79.55 ± 21.60
Progressive CE	99.17 ± 0.83	99.06 ± 0.89	75.01 ± 23.61	74.96 ± 23.69
DART	$\mathbf{99.99 \pm 0.01}$	$\mathbf{99.99 \pm 0.01}$	$\mathbf{99.88 \pm 0.16}$	$\mathbf{99.88 \pm 0.17}$
Progressive DART	99.98 ± 0.01	99.98 ± 0.01	98.75 ± 1.77	98.73 ± 1.78

B.2 CONSTRAINT-LEVEL SUDOKU ANALYSIS

To better understand the TRM results, we evaluate final Sudoku predictions using constraint-level diagnostics in addition to exact match. This distinction matters because the masked input may not uniquely determine the original solved grid. Exact match measures whether the predicted grid equals the reference completion used to create the puzzle, and is therefore a strict diagnostic rather than the only notion of correctness. Clue consistency measures whether predictions preserve the observed cells in the masked puzzle. Valid+clue consistency is the primary correctness metric for this non-unique setting: it measures the fraction of predictions that are both structurally valid Sudoku boards and consistent with the observed clues. Masked-cell accuracy evaluates only the cells hidden from the input.

Table 4: Constraint-level diagnostics for TRM on masked Sudoku (mean $\pm$ std over at least three seeds per method). All models are evaluated on the same 10,000 test puzzles per masking ratio.

Mask	Method	Exact match	Clue consistent	Valid+clue consistent	Masked-cell acc.	Avg. clue violations
30%	CE	95.6 ± 0.1	99.4 ± 0.1	99.2 ± 0.0	99.1 ± 0.0	0.010 ± 0.003
30%	Label smoothing	95.8 ± 0.1	99.7 ± 0.1	99.5 ± 0.1	99.2 ± 0.0	0.006 ± 0.002
30%	Soft target, no critic	8.3 ± 3.6	$\mathbf{100.0 \pm 0.0}$	8.4 ± 3.7	86.4 ± 1.8	$\mathbf{0.000 \pm 0.000}$
30%	CE+soft-target MSE	95.5 ± 0.2	99.5 ± 0.3	99.4 ± 0.2	99.2 ± 0.1	0.009 ± 0.005
30%	DART	$\mathbf{96.2 \pm 0.2}$	99.8 ± 0.1	$\mathbf{99.8 \pm 0.1}$	$\mathbf{99.3 \pm 0.1}$	0.002 ± 0.002
40%	CE	78.1 ± 0.9	92.3 ± 1.0	90.6 ± 0.9	95.8 ± 0.2	0.133 ± 0.036
40%	Label smoothing	78.9 ± 1.4	94.2 ± 0.3	91.4 ± 1.5	96.2 ± 0.3	0.091 ± 0.009
40%	Soft target, no critic	0.1 ± 0.1	$\mathbf{100.0 \pm 0.0}$	0.2 ± 0.2	73.4 ± 2.4	$\mathbf{0.000 \pm 0.000}$
40%	CE+soft-target MSE	74.8 ± 4.9	88.5 ± 6.2	86.5 ± 5.9	94.9 ± 1.5	0.260 ± 0.225
40%	DART	$\mathbf{82.4 \pm 0.8}$	96.9 ± 0.3	$\mathbf{95.3 \pm 0.6}$	$\mathbf{96.8 \pm 0.2}$	0.038 ± 0.004
50%	CE	32.0 ± 2.4	58.2 ± 3.7	51.8 ± 3.8	81.8 ± 1.0	0.851 ± 0.135
50%	Label smoothing	28.9 ± 3.8	61.8 ± 2.3	45.8 ± 6.2	82.3 ± 1.4	0.676 ± 0.080
50%	Soft target, no critic	0.0 ± 0.0	$\mathbf{100.0 \pm 0.0}$	0.0 ± 0.0	58.1 ± 2.0	$\mathbf{0.000 \pm 0.000}$
50%	CE+soft-target MSE	15.0 ± 8.6	29.9 ± 18.3	24.5 ± 14.2	69.8 ± 9.8	3.209 ± 2.661
50%	DART	$\mathbf{40.4 \pm 2.4}$	74.7 ± 1.9	$\mathbf{64.8 \pm 3.4}$	$\mathbf{85.5 \pm 0.9}$	0.354 ± 0.039

The diagnostic clarifies how DART helps TRM under heavier masking. At 30% masking, CE, label smoothing, CE+soft-target MSE, and DART are already near saturation, so the gains are small. At 40% and 50% masking, the input leaves many more cells unspecified, and the dominant challenge becomes preserving observed clues while completing the hidden cells consistently. Averaged over at least three seeds, DART substantially reduces clue violations and improves valid+clue consistency. Label smoothing improves some clue-consistency metrics but does not reliably improve the primary valid+clue metric under the hardest shift. Softened targets without a critic preserve clues by construction-like behavior but fail to produce structurally valid boards, showing that local target relaxation alone is not enough. CE+soft-target MSE is stronger than softened targets without a critic at 30%–40% masking, but it degrades sharply at 50% masking, indicating that direct pointwise matching to the softened target is not a substitute for the conditional critic.

The 50% setting also reveals a useful limitation. DART improves clue consistency and exact match, but its standalone structural-validity rate is lower than CE in this diagnostic. This suggests that under severe ambiguity, DART primarily regularizes predictions toward the observed input and target completion, while not fully solving global Sudoku constraint satisfaction. This is consistent with our broader conclusion that DART improves recurrent reasoning stability but does not replace explicit verification or search.

B.3 SUDOKU H-CYCLE SWEEP

To separate objective effects from raw iterative compute, we evaluate the Sudoku 50% masking setting while increasing the number of TRM high-level cycles from $H = 3$ to $H = 6$. Table 5 reports valid+clue consistency, the primary metric for this non-unique completion setting. Increasing H improves every trainable objective only slightly. DART rises by 1.6 points and remains more than 14 points above CE at $H = 6$. The CE+soft-target MSE objective stays near 25% across the sweep, showing that the failure of direct softened-target matching is not repaired by extra TRM cycles. The hard-target critic variant receives a larger compute gain ($+2.7$ points), but remains far below full DART, indicating that adding critic pressure to one-hot targets does not recover the benefit of the soft/noise target neighborhood. CE and label smoothing receive less than one point from the same compute increase, while softened targets without a critic remain structurally invalid despite preserving observed clues in the constraint diagnostic above.

Table 5: Effect of TRM high-level cycles H on Sudoku at 50% masking. Values are valid+clue consistency (%), mean ± std over at least three seeds per method.

Method	$H = 3$	$H = 4$	$H = 5$	$H = 6$	$\Delta_{3\to6}$
CE	51.4 ± 3.6	51.7 ± 3.7	51.9 ± 3.7	52.0 ± 3.7	$+0.6$
Label smoothing	45.8 ± 6.2	46.1 ± 6.1	46.3 ± 6.0	46.3 ± 6.0	$+0.5$
Soft target, no critic	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	$+0.0$
CE+soft-target MSE	24.5 ± 14.2	24.8 ± 14.3	25.0 ± 14.3	25.0 ± 14.3	$+0.5$
Hard-target critic	36.4 ± 5.8	37.7 ± 4.6	38.5 ± 3.9	39.1 ± 3.4	$+2.7$
DART	$\mathbf{64.8 \pm 3.4}$	$\mathbf{65.7 \pm 3.5}$	$\mathbf{66.2 \pm 3.5}$	$\mathbf{66.4 \pm 3.5}$	$\mathbf{+1.6}$

The same pattern appears in stricter diagnostics. From $H = 3$ to $H = 6$, exact match rises from 40.4% to 41.4% for DART, from 31.7% to 32.1% for CE, from 15.0% to 15.3% for CE+soft-target MSE, and from 22.2% to 23.8% for the hard-target critic variant. Average violated units drop from 1.11 to 0.87 for DART, from 1.02 to 0.89 for CE, from 4.80 to 4.40 for CE+soft-target MSE, and from 5.58 to 4.77 for the hard-target critic variant. These small absolute changes suggest that more recurrent compute helps local refinement, but does not change the dominant failure mode unless the objective also encourages input-conditioned valid completions.

B.4 Trajectory Analysis

Figure 5 and Table 6 summarize the trajectory-level evidence. Maze is the main trajectory stress test in the paper; Chess is included here as a supporting late-iteration collapse case.

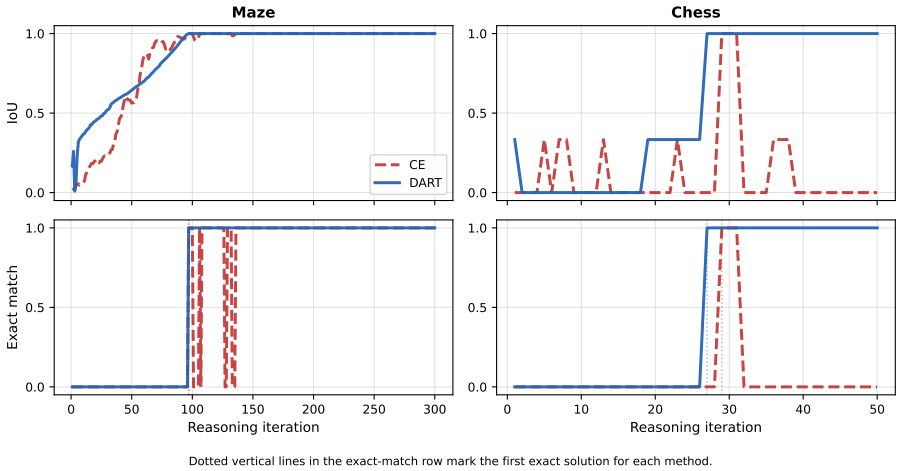


Figure 5: Sample reasoning trajectories for CE and DART on Maze and Chess. The top row shows IoU with the target solution, while the bottom row shows exact match; dotted vertical lines mark the first exact solution. DART reaches and retains exact solutions more reliably, whereas CE can enter the correct region transiently and then drift away.

Table 6: Aggregate trajectory statistics from three-seed evaluations over held-out samples (Maze: 128 samples; Chess: 512 samples). First exact iteration and retention are computed over samples that reach an exact solution at least once.

Task	Method	Solved once (%)	Final exact (%)	First exact iter	Retention after first exact (%)	Exact-match AUC (%)
Maze 31×31	CE	51.6	43.8	88.6	82.0	28.4
Maze 31×31	DART	**100.0**	**100.0**	94.0	**99.9**	**68.9**
Chess	CE	84.8	0.0	28.0	11.6	4.5
Chess	DART	**91.0**	**15.8**	**23.1**	**63.5**	**32.3**

The exact-match trace is important because high overlap alone can hide invalid intermediate states. In Maze, a path prediction may share most cells with the target while containing a small break or spurious branch that makes the decoded path invalid. In Chess, a prediction may assign probability mass to a plausible move neighborhood while still selecting the wrong square after argmax decoding. Exact match therefore exposes whether the recurrent process has truly entered the valid discrete solution set, not merely moved close to it in overlap space.

This distinction clarifies the role of the critic. CE supplies supervision only at the selected target configuration, so once the model reaches a nearly correct state, later recurrent updates can still move it along directions that preserve local overlap but destroy global validity. DART instead compares generated input-solution pairs against samples from a local target neighborhood conditioned on the input. This encourages the recurrent trajectory to remain near successful traces rather than repeatedly crossing the boundary between valid and invalid states. The post-solution stability observed in both Maze and Chess is therefore consistent with the conditional joint-matching formulation of the method.

B.5 Supporting Chess Experiments

Chess provides a supporting case because its dominant failure is late-iteration collapse rather than input masking or spatial scale. CE reaches a moderate peak of $80.81 \pm 0.37\%$ but collapses to $0.92 \pm 0.08\%$ by iteration 50. DART improves peak solution quality to $84.83 \pm 0.03\%$ and keeps the useful-accuracy window open longer: at iteration 40, CE has already fallen to $0.56 \pm 0.11\%$, whereas DART still retains $28.58 \pm 10.47\%$. However, Chess also shows that the critic is not a universal late-iteration stabilizer: the no-critic softened-target baseline has stronger final accuracy on this task, while DART gives the best peak. We therefore treat Chess as a task-specific stability case rather than the central claim. The strongest evidence for DART remains the paired Maze–Sudoku pattern, where gains grow as the valid solution must be recovered from a larger or more underdetermined output space.

Progressive training is a strong fit for this Chess failure mode because it directly targets long-horizon stability. Figure 6 and Table 7 show that Progressive CE preserves performance through iteration 50, while DART obtains the highest non-progressive peak. Progressive DART combines the two effects and obtains the best final Chess accuracy, $83.52 \pm 0.16\%$. This supports the same distinction seen in Maze from the opposite direction: progressive loss can prevent a discovered answer from being overwritten, whereas DART mainly improves the quality and input-compatibility of the solutions reached by the recurrent process.

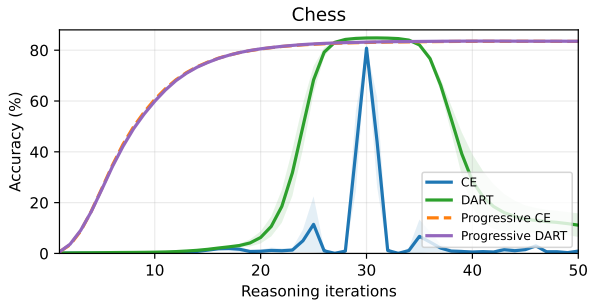


Figure 6: Progressive training baselines on Chess. Progressive training removes the late-iteration collapse of CE and DART, while DART alone reaches the highest peak accuracy.

Table 7: Progressive training baselines on Chess. Accuracy values are mean ± std over three seeds.

Method	Peak	Peak iter	Acc. @ iter 40	Final (50)
CE	80.81 ± 0.37	30, 30	0.56 ± 0.11	0.92 ± 0.08
Progressive CE	83.50 ± 0.03	41, 45	83.48 ± 0.02	83.46 ± 0.02
DART	$\mathbf{84.83 \pm 0.03}$	31, 31	28.58 ± 10.47	11.13 ± 4.64
Progressive DART	83.59 ± 0.09	41, 50	$\mathbf{83.58 \pm 0.09}$	$\mathbf{83.52 \pm 0.16}$

B.6 Target-Smoothing and Component Baselines

This appendix separates four effects that are bundled or adjacent in full DART: ordinary confidence smoothing, local soft/noise target construction, direct pointwise matching to the softened target, and the input-conditioned critic. The CE+soft-target MSE ablation removes the critic and optimizes

$$\mathcal{L}_{\text{CE+MSE}} = (1 - \lambda_{\text{mse}})\mathcal{L}_{sup} + \lambda_{\text{mse}}\, \mathbb{E}_{\hat{v} \sim P_{\text{ref}}} \left[\|\hat{y}_T - \hat{v}\|_2^2 \right], \quad \lambda_{\text{mse}} = 0.5, \tag{10}$$

where $\hat{y}_T$ is the final recurrent output used for training. This baseline uses the same softened target samples as DART, but replaces the learned critic score $C_\psi(x, s)$ with a fixed Euclidean penalty in the output simplex. It therefore tests whether explicit pointwise regression to the softened target is sufficient, without learning an input-conditioned compatibility function over (x, s) pairs.

On Sudoku, Figure 2 and Table 2 show that none of the partial variants matches full DART under 50% masking. Label smoothing remains a uniform target perturbation and drops below CE on valid+clue consistency. Softened targets without a critic preserve observed clues but fail to produce globally valid boards. CE+soft-target MSE directly penalizes the distance between generated solutions and softened targets, but its valid+clue consistency falls to 24.5% at 50% masking, below both CE and the hard-target critic. Conversely, the hard-target critic variant keeps the adversarial critic but removes soft/noise target construction; it improves over the no-critic softened-target collapse, but remains well below full DART in exact match, valid+clue consistency, and masked-cell accuracy.

The same hard-target critic ablation behaves differently across tasks. On Maze, the critic alone is almost sufficient: it reaches $99.95 \pm 0.08\%$ peak accuracy and $99.95 \pm 0.08\%$ final accuracy, matching full DART within the near-saturated regime. On Chess, it reaches a strong peak of $84.65 \pm 0.17\%$, comparable to full DART, but collapses quickly to $21.86 \pm 16.99\%$ at iteration 40 and $6.67 \pm 7.60\%$ at the final iteration. On Sudoku, the same ablation remains much weaker than full DART, reaching only 36.4% valid+clue consistency at the default $H = 3$ and 39.1% even at $H = 6$. These results suggest that the critic alone can induce strong stability on Maze, where the target path structure is comparatively direct, but the local soft/noise target neighborhood becomes important for more ambiguous or constraint-heavy tasks such as Chess and Sudoku. This supports the interpretation that DART is not only target smoothing: the local target neighborhood and the critic work together, with their relative importance depending on the task structure.

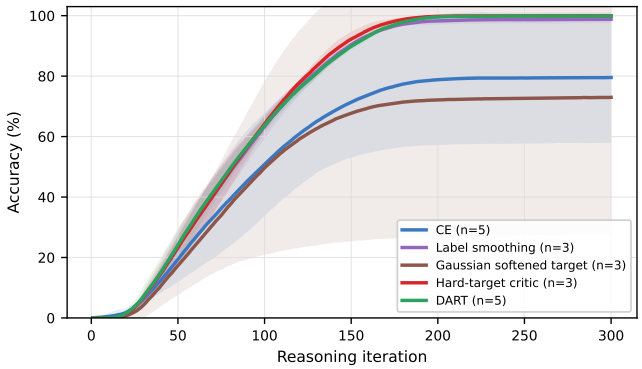


Figure 7: Maze accuracy versus reasoning iterations for target-smoothing and component baselines. Gaussian softened targets use the same local target construction as DART but remove the adversarial critic; hard-target critic keeps the critic but removes the soft/noise target construction.

Table 8: Target-smoothing baselines on Maze 31×31 (mean $\pm$ std over independent seeds). CE and DART use five seeds; label smoothing, Gaussian softened target without a critic, and hard-target critic use three seeds.

Method	Peak	Acc. @ iter 250	Final (300)
CE	79.65 ± 21.62	79.39 ± 21.71	79.55 ± 21.60
Label smoothing ($\epsilon = 0.3$)	98.82 ± 1.21	98.70 ± 1.26	98.80 ± 1.23
Gaussian softened target, no critic	72.98 ± 45.27	72.64 ± 45.30	72.98 ± 45.26
Hard-target critic	$\mathbf{99.95 \pm 0.08}$	$\mathbf{99.95 \pm 0.09}$	$\mathbf{99.95 \pm 0.08}$
DART	99.88 ± 0.16	99.87 ± 0.18	99.88 ± 0.17

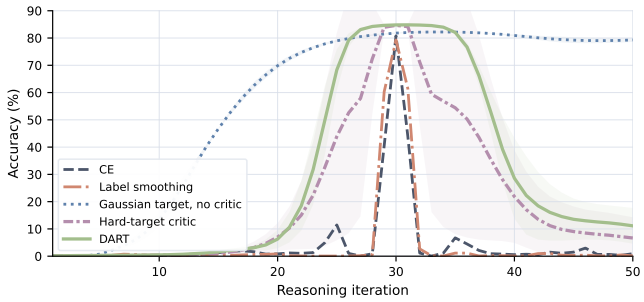


Figure 8: Chess accuracy versus reasoning iterations for target-smoothing and component baselines. Gaussian softened targets without a critic stabilize final accuracy better than DART on this task, while the hard-target critic and full DART reach strong peaks but collapse after the peak.

Table 9: Target-smoothing baselines on Chess. Accuracy values are mean ± std over three seeds.

Method	Peak	Peak iter	Acc. @ iter 40	Final (50)
CE	80.81 ± 0.37	30, 30	0.56 ± 0.11	0.92 ± 0.08
Label smoothing ($\epsilon = 0.3$)	79.29 ± 0.04	30, 30	0.07 ± 0.00	0.14 ± 0.00
Gaussian softened target, no critic	82.22 ± 0.09	33, 34	$\mathbf{80.93 \pm 0.28}$	$\mathbf{79.29 \pm 0.49}$
Hard-target critic	84.65 ± 0.17	30, 31	21.86 ± 16.99	6.67 ± 7.60
DART	$\mathbf{84.83 \pm 0.03}$	31, 31	28.58 ± 10.47	11.13 ± 4.64

C Appendix

C.1 Datasets

In this section, we provide detailed descriptions of the datasets used in our experiments, including their construction procedures and key properties. The Maze and Chess datasets are publicly available through the `easy-to-hard-data` Python package, while the Masked Sudoku benchmark is constructed from the Extreme Sudoku dataset Wang et al. (2025).

C.1.1 Maze Dataset

Maze instances are generated using a depth-first search (DFS) algorithm. The grid is initially initialized with walls at every cell boundary. Starting from a designated cell, DFS is applied to visit all cells while removing walls along the traversal path, ensuring connectivity.

The resulting mazes exhibit non-uniform path length distributions. Although the generation process may introduce duplicate samples, the duplication rate is below 0.5% and is therefore ignored in our analysis. Each maze is paired with a target binary mask indicating the optimal path between the start and end points.

C.1.2 Chess Puzzle Dataset

Chess puzzles are derived from the Lichess database and represented in Forsyth–Edwards Notation (FEN). Each board state is converted into a tensor of size $8 \times 8 \times 12$, where each channel corresponds to a specific piece type and color.

The target output is defined as the optimal move in UCI format, encoded as an 8×8 binary mask marking the origin and destination squares of the move.

The dataset is constructed by analyzing approximately 200 million games using the Stockfish 12/13 NNUE engine. While user identities are not explicitly included, they may be indirectly inferred through puzzle replay on the Lichess platform. The dataset is released under the Creative Commons CC0 license, allowing unrestricted use for research purposes.

C.1.3 MASKED SUDOKU DATASET

The masked Sudoku dataset is constructed from complete Sudoku solutions provided in the Hard Sudoku benchmark Wang et al. (2025). Each example is represented as a 9×9 grid, where entries take values in $\{1, \ldots, 9\}$. To create a partially observed puzzle, we randomly mask a subset of cells and replace them with a special unknown token, while keeping the full solved grid as a reference completion for training and strict exact-match diagnostics.

We use this dataset to define a masking-based easy-to-hard generalization benchmark. During training, each of 1,000 solved grids is converted into a puzzle by masking 20% of its cells. At evaluation time, we increase the masking ratio to 30%, 40%, and 50%, thereby controlling task difficulty through the amount of missing information. Higher masking ratios make the reasoning problem harder because fewer constraints are directly observed in the input, and the resulting puzzle may admit multiple valid completions. We therefore report valid+clue consistency as the primary Sudoku correctness metric: a prediction is counted as correct if it forms a structurally valid Sudoku board and preserves the observed cells.

This construction preserves the same input-output format across difficulty levels while inducing substantial distribution shift in the amount of observable information. In our experiments, we evaluate on a held-out test set of 10,000 masked puzzles, which allows us to assess whether recurrent reasoning models trained on easier partially observed instances can generalize to harder ones under more severe masking.

C.2 HYPERPARAMETERS

Table 10 summarizes the main training hyperparameters used in our experiments. In each training run, we hold out 20% of the training data as a validation set and select for final testing the checkpoint that achieves the highest validation score. For Sudoku, both training and checkpoint selection use puzzles with 20% masking; the validation split is constructed only at this training masking ratio. The 30%, 40%, and 50% masked Sudoku splits are used only for evaluation, so these results measure transfer from the easier 20% masking regime to harder partial-observation settings.

Table 10: Training hyperparameters used in the experimental runs. Dashes indicate that we did not utilize those options.

task	optim.	lr	lr critic	decay schedule	decay factor	warm-up	epochs	clip
Mazes	adamw	1e-3	5e-5	–	–	10	50	0.01
Chess	sgd	1e-2	5e-5	[100, 110]	0.01	3	120	0.01
Sudoku	adam	1e-3	5e-5	[1000]	0.01	10	3000	0.01

C.3 MODEL ARCHITECTURES

For all DTS models on the easy-to-hard tasks, we use the same backbone architecture. The projection layer p is implemented as a convolutional layer that maps the raw input to a hidden representation of dimension $d = 128$. The recurrent block r is implemented as a residual block with two convolutional layers, and the prediction head h is a final convolutional layer that maps the hidden state to the output space. For all tasks, we set the number of recurrent reasoning iterations during training to $T = 30$.

For TRM, we follow the architecture in the original paper, but reduce the hidden size from 512 to 128. During training, we use $H = 3$ high-level cycles and $L = 6$ low-level cycles.

For DART, we set the Gaussian noise standard deviation to $\sigma = 0.3$ for all tasks. The critic is constructed to mirror the generator architecture. For DTS-based models, the critic uses the same architecture as the generator, but with the number of recurrent reasoning iterations set to $T = 5$ during training. For TRM, we use a critic with $H = 2$ high-level cycles and $L = 3$ low-level cycles. The Wasserstein critic is constrained using weight clipping. We use a clipping threshold of 0.01 for Maze, Chess, and Sudoku. For optimization stability, we additionally use gradient clipping with threshold 1 for Sudoku; Maze and Chess do not use gradient clipping. The generator and critic are updated once per minibatch in alternation, and the critic learning rate is reported in Table 10.

We analyze the effects of different values of σ and different critic architectures in the Ablation section D.

C.4 Computational resources

All experiments were conducted using a NVIDIA RTX 5090 GPU. For the DTS experiments, Maze models require up to two hours. DTS models on Chess are the most computationally demanding and require between 6 and 8 hours to complete. For TRM on Sudoku, training also takes between 6 and 8 hours.

Label smoothing and Gaussian softened-target baselines modify only the target tensor and do not instantiate an additional network, so their training cost is the same as CE up to negligible target-construction overhead. DART adds the critic forward/backward pass during training. The critic mirrors the generator backbone, so trainable parameters approximately double during training, although the critic is discarded at inference time. On our DTS setup, the generator has 783,504 parameters on Maze and 804,240 parameters on Chess.

Table 11: Representative Chess training cost on one RTX 5090. Experiments use three seeds; GPU-hours are reported per run and assume 120 epochs.

Method	Seconds / epoch	Relative to CE	GPU-hours
CE	59.0	1.00×	1.97
Label smoothing	59.0	1.00×	1.97
Gaussian softened target	59.0	1.00×	1.97
DART	104.4	1.77×	3.48
Progressive CE	83.8	1.42×	2.79
Progressive DART	127.4	2.16×	4.25

Inference cost is unchanged by DART because the critic is not used at test time. In our wall-clock measurements, Maze evaluation at 300 iterations takes 334 seconds for both CE and DART, and Chess evaluation at 50 iterations with matched batch size takes 9.5 seconds for CE and 10.0 seconds for DART.

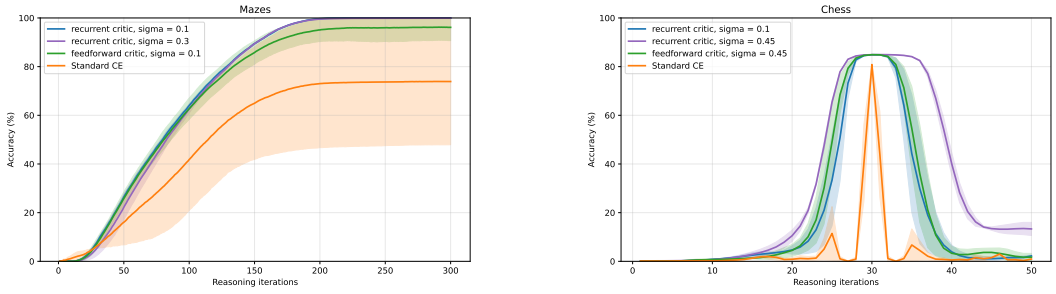
D Additional ablations

D.1 Effect of Target Distribution Width.

We study the effect of target distribution width in DART by varying the standard deviation σ that defines the neighborhood around the ground-truth solution. Intuitively, σ controls how much deviation from the exact solution is tolerated during training. Figure 9a shows that changing σ from 0.1 to 0.3 has minimal effect on Maze. The reasoning trajectories remain stable across iterations for both settings, and the final accuracies are nearly identical ($99.88 \pm 0.12\%$ vs. $99.91 \pm 0.13\%$). These results indicate that performance on Maze is largely insensitive to the precise choice of σ. The effect is more pronounced on Chess (Figure 9b). Both DART variants outperform the CE baseline by roughly 4% in peak accuracy ($84.82 \pm 0.18\%$ and $84.93 \pm 0.03\%$ versus $80.8 \pm 0.37\%$). Moreover, performance remains consistent across $\sigma = 0.1$ and $\sigma = 0.45$, indicating that DART is effective across a broad range of target widths. However, the larger σ produces a wider high-accuracy plateau, whereas the smaller σ peaks more sharply and degrades earlier. This behavior suggests that broader target distributions regularize a wider reference-centered neighborhood, allowing the model to learn nearby target-compatible intermediate states rather than only the exact target trajectory. Consequently, the iterative reasoning process becomes more robust to small accumulated errors, improving stability over long reasoning horizons and generalization to more difficult tasks. Overall, DART remains effective across a broad range of σ values, while larger target widths can provide additional robustness on harder reasoning problems.

D.2 Effects of Critic Architecture

We study the effect of critic architecture in DART by comparing a recurrent critic with a feedforward critic under the same σ. The feedforward critic uses the same residual building block as the recurrent critic, but replaces recurrent reuse with depth by stacking five independent residual blocks



(a) Maze (b) Chess

Figure 9: Effect of the target distribution width σ and the critic architecture on reasoning performance.

instead of applying one shared block for five iterations. Figure 9a shows that, on Maze, DART with a feedforward critic still substantially outperforms the five-seed CE baseline ($96.23 \pm 5.32\%$ vs. $79.65 \pm 21.62\%$), but remains less stable than DART with a recurrent critic ($99.88 \pm 0.12\%$) under the same setting of $\sigma = 0.1$. This suggests that DART does not rely on a recurrent critic to improve performance, although recurrence provides stronger regularization for the iterative reasoning process. This pattern is even clearer on Chess (Fig. 9b). Both critic variants outperform the CE baseline, with peak accuracies of $84.90 \pm 0.03\%$ for the feedforward critic and $84.93 \pm 0.03\%$ for the recurrent critic, compared with $80.8 \pm 0.37\%$ for CE. This suggests that conditional joint matching is not tied to one critic architecture. However, the recurrent critic maintains high accuracy over a wider range of reasoning iterations, whereas the feedforward critic degrades earlier. This suggests that although a feedforward critic is sufficient to improve solution quality, the recurrent critic is better suited to inducing stable intermediate reasoning states over long computation horizons. Overall, DART is robust to the choice of critic architecture, but recurrent critics yield more stable reasoning trajectories, especially on harder tasks.

D.3 Additional Sample Trajectory Analysis

We further inspect individual reasoning trajectories to test the mechanism suggested by our hypothesis. If the adversarial critic is acting as an input-conditioned distributional constraint, then the generated trajectory should not merely hit the correct solution at one iteration; it should enter the high-density region of successful traces and remain close to it. We therefore plot both IoU and exact match: IoU captures partial overlap with the target, while exact match verifies that the discrete solution is fully correct. Conversely, a model trained only with point-wise CE may reach a high-overlap or even exact state transiently, but can drift into invalid regions because later recurrent updates are not directly constrained by the local solution neighborhood.

Figure 5 shows this pattern on representative Maze and Chess examples. On Maze, both CE and DART first reach exact match at iteration 97, but their post-solution behavior differs. CE drops out of exact match 11 times after first solving the instance, while DART remains exact for all subsequent iterations. This supports the view that the critic does not simply accelerate convergence; it stabilizes the trajectory once it reaches the successful region.

The same effect is stronger on Chess, where overthinking is the dominant failure mode. CE reaches exact match at iteration 29 but does not retain it: only 13.6% of the subsequent iterations remain exact, and the final prediction has zero IoU with the target. DART reaches exact match at iteration 27 and stays exact through the final iteration. Thus, even on a task where CE can briefly identify the correct move, the adversarial signal helps prevent the recurrent state from drifting away from the reference-centered neighborhood.

The aggregate trajectory statistics in Table 6 show that this is not an isolated example. On Maze, DART solves every evaluated sample at least once and also finishes every sample in an exact state. CE solves only 51.6% of samples at least once and finishes at 43.8%, indicating that many trajectories either never enter the successful region or enter it only transiently. The exact-match AUC makes this difference clearer: DART spends 68.9% of its evaluated trajectory in an exact state on average, compared with only 28.4% for CE.

On Chess, CE's failure mode is almost entirely a retention failure. It reaches an exact solution at least once for 84.8% of samples, but the final exact-match rate is 0.0%. In other words, CE often discovers a correct move during recurrent refinement, but subsequent iterations overwrite it. DART improves both discovery and retention: it solves 91.0% of samples at least once, reaches the first exact solution earlier on average (23.1 vs. 28.0 iterations), and increases post-solution retention from 11.6% to 63.5%. The exact-match AUC similarly rises from 4.5% to 32.3%, showing that DART spends a much larger fraction of the reasoning horizon in a valid discrete state.

These qualitative trajectories are consistent with the aggregate results in Section 5. DART improves most when point-wise supervision becomes brittle: under large spatial extrapolation in Maze, and under late-iteration overthinking in Chess. The critic encourages generated outputs to remain in regions that resemble softened successful traces for the specific input, which provides a stabilizing force that simple one-hot supervision lacks.

The exact-match trace is important because high overlap alone can hide invalid intermediate states. In Maze, a path prediction may share most cells with the target while containing a small break or spurious branch that makes the decoded path invalid. In Chess, a prediction may assign probability mass to a plausible move neighborhood while still selecting the wrong square after arg max decoding. The exact-match metric therefore exposes whether the recurrent process has truly entered the valid discrete solution set, not merely moved close to it in overlap space.

This distinction clarifies the role of the critic. CE supplies supervision only at the selected target configuration, so once the model reaches a nearly correct state, later recurrent updates can still move it along unconstrained directions that preserve some local overlap but destroy global validity. DART instead compares generated input-solution pairs against samples from a local target neighborhood conditioned on the input. This encourages the recurrent trajectory to remain near the set of successful traces rather than repeatedly crossing the boundary between valid and invalid states. The observed post-solution stability of DART in both Maze and Chess is therefore consistent with the conditional joint-matching formulation of the method.