
DART: Distributional Adversarial Recurrent Training for Algorithm Learning

Hieu Tran Bao, Phung Thanh Dang & Pham Quang Nhat Minh
FPT IS AI R&D Center
Hanoi, Vietnam

Hoang Thanh Tung
VNU – University of Engineering and Technology
Hanoi, Vietnam

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 becomes increasingly problematic as task difficulty grows: valid solutions occupy a tiny region of the solution space, while invalid solutions proliferate rapidly. 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 RRM, 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 [1, 2]. 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 [3], branching [4, 5], and verification [6]—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 [7].

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) [8] and *Hierarchical Reasoning Models* (HRM) [9], 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 [10, 11] suggests that RRM can learn *algorithms* from data and exhibit systematic generalization capability. Extending this idea, Schwarzschild et al. [12] 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 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, RRM are still typically optimized at the instance level: for each logical input, the model is trained to match a single target solution. This creates a challenge for *easy-to-hard* generalization because, as problem complexity increases, the number of incorrect solutions grows explosively while the set of valid solutions remains highly constrained. As a result, the probability mass of correct solutions becomes vanishingly small relative to the vast space of invalid ones. Consequently, 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 [13], 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) [14]. 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 adversarial distribution alignment, our method encourages the model not only to produce correct solutions but also to remain close to an input-conditioned neighborhood of the target solution, thereby improving solution quality and trajectory stability on complex instances.

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 [8, 9], 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. Under this setting, DART also improves TRM’s reasoning capability on more difficult Sudoku instances.

Overall, we summarize our contribution as follows:

- To the best of our knowledge, we are the first to analyze *easy-to-hard* generalization through the lens of solution-space imbalance. We argue that instance-level training exacerbates this difficulty as the imbalance becomes more severe as the complexity increases.
- We reformulate the solution space of reasoning problems from a distributional perspective and propose **distributional adversarial recurrent training (DART)** to help recurrent reasoning models discover valid solutions more effectively.
- We show that DART improves the performance of multiple RRM across a range of logical reasoning tasks, yielding more stable accuracy across reasoning iterations. We further compare against label smoothing, Gaussian softened targets without a critic, and progressive

¹A 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.

training baselines to separate target softening, adversarial alignment, and long-horizon stabilization.

2 Related Work

Easy-to-hard generalization and algorithm learning. Sun et al. [15] 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 [16]. Graves et al. [11] 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 [10] can learn arithmetic procedures such as addition and multiplication and generalize to arbitrary-length sequences. Deep Thinking systems [12] 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 [17]. To mitigate this issue, Bansal et al. [17] 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 [18]. We use the recall architecture as the backbone in all DTS experiments and explicitly compare against progressive training in Section 5. More recent RRM models such as Tiny Recursive Models (TRM) [8] and Hierarchical Reasoning Models (HRM) [9] 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) [19], which enables recurrent models to halt once a learned stopping probability reaches a threshold. Universal Transformers [20] and Adaptive Recurrent Vision [21] applied this idea to a range of settings. PonderNet [22] improves the halting mechanism in ACT and outperforms ACT on logical extrapolation tasks such as prefix sums. More broadly, [17, 23, 24] 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) [25] and followed by advances such as WGAN [14]. Adversarial objectives have also been used to guide structured learning. SADM [26] allows diffusion models to capture semantic constraints among generated samples during denoising through adversarial training and batch-level similarity matrices. AdvBN [27] 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 identify a key challenge in easy-to-hard generalization as the severe imbalance between the valid and invalid solution spaces. By viewing logical reasoning tasks as a generation process, we argue that adversarial guided training can be applied to address this issue. Based on this perspective, we propose DART as a method for calibrating and regularizing the iterative reasoning process of RRM models. 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 align generated outputs with that local target neighborhood.

3 Preliminary

We consider logical reasoning tasks where the goal is to predict a structured solution y from an input problem instance x . Examples include combinatorial puzzles, constraint satisfaction problems, and algorithmic reasoning tasks.

Deep Thinking Systems (DTSs) [12] depart from classical recurrent architectures by producing the entire (representation of the) solution and refining that iteratively. DTSs process the input holistically, refine the solution through iterative computation, and learn algorithms end-to-end without supervision over individual intermediate steps. A DTS f_θ is defined as follows:

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

where p is an input projection function, r is a recurrent reasoning module applied for T iterations, and h is an output prediction head. A DTS can generalize to substantially harder/easier instances at test time by increasing/decreasing the number of reasoning iterations T at test time. To prevent the model from forgetting the original problem during iterative computation, Bansal et al. [17] proposed the *Recall* architecture, in which the original input x is fed to every recurrent step:

$$\phi_{t+1} = r([\phi_t, x]) \quad (2)$$

In all DTS experiments in this paper, we use the Recall variant as the DTS backbone.

Tiny Recursive Models (TRMs) [8] improve on the *Hierarchical Reasoning Model* [9], which is built on the idea that structured reasoning requires iterative refinement and self-correction rather than a single large forward pass. This design is inspired by the human reasoning process, in which tentative hypotheses are revised multiple times before committing to an answer. TRM constructs a first draft solution, repeatedly checks and refines it, and gradually moves toward a consistent final prediction. It maintains two latent states throughout inference: a *low-level reasoning state* z_t^ℓ , which serves as a flexible workspace for exploring constraints and local relationships, and a *high-level answer state* z_t^h , which stores the current global hypothesis for the final solution.

$$z_{t+1}^\ell = r_\ell(z_t^\ell, z_t^h, x), \quad z_{t+1}^h = r_h(z_{t+1}^\ell, z_t^h), \quad \hat{y} = h(z_T^h), \quad (3)$$

where r_ℓ updates the low-level reasoning state, r_h updates the high-level answer state, and h maps the final high-level state to the output prediction. In practice, TRM reuses a single small computational block—typically a lightweight two-layer transformer or MLP—across many recursive steps, allowing the model to perform deep reasoning with a small number of parameters.

TRM alternates between multiple low-level refinement steps and a high-level update step. Intuitively, the model *thinks* several times in the low-level workspace before revising its global answer hypothesis once, and this cycle is repeated until convergence. This separation between exploratory reasoning and answer consolidation enables TRM to solve structured tasks such as Sudoku through repeated draft-check-refine behavior rather than a single-pass prediction.

Instance-level optimization. Both DTS and TRM are trained to minimize a supervised loss between the predicted solution $\hat{y}$ at the final step T and the ground-truth solution y^* :

$$\mathcal{L}_{sup} = \ell(\hat{y}, y^*) \quad (4)$$

where ℓ is typically cross-entropy or mean squared error. This optimization formulation treats the correct solution as a deterministic target and ignores the local structure of the valid solution space.

4 Proposed Method

This section presents the details of our proposed **DART** framework. We show how to reformulate a reasoning problem as a distribution learning problem (Sec. 4.1) and how to align the model’s distribution with the target solution distribution (Sec. 4.2).

4.1 Soft Solution Distribution

We consider discrete logical reasoning problems with input x and target solution y^* such that $x, y^* \in \{c_0, c_1, \dots, c_k\}^{m \times n}$, where c_0 denotes a padding token and $c_1, \dots, c_k$ 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:

$$\tilde{v}_{i,j} = v_{i,j}^* + \epsilon_{i,j}, \quad \epsilon_{i,j} \sim \mathcal{N}(0, \sigma^2 I_{k+1}). \quad (5)$$

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 *argmax*; 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 distribution, we introduce an adversarial objective inspired by Wasserstein GANs. Let P_{gen} denote the distribution of solutions produced by the reasoning model f_θ , and P_{real} denote the soft solution distribution constructed around the ground-truth solution.

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. 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:

$$\xi_0 = p_c(z), \quad \xi_{t+1} = r_c(\xi_t), \quad C_\psi(x, s) = q_c(\xi_T), \quad (6)$$

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 assess whether a soft solution is globally consistent with the input instance 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:

$$\mathcal{L}_{\text{critic}} = -\left(\mathbb{E}_{y \sim P_{\text{real}}}[C_\psi(x, \hat{v})] - \mathbb{E}_{\hat{y} \sim P_{\text{gen}}}[C_\psi(x, \hat{y})]\right) \quad (7)$$

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.

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

```
1: while not converged do
2:   Sample minibatch  $(x, y^*)$ 
3:   // Construct soft solution distribution  $P_{\text{real}}$ 
4:    $v_{i,j} \leftarrow \text{onehot}(y_{i,j}^*)$ 
5:    $\tilde{v}_{i,j} \leftarrow v_{i,j} + \epsilon_{i,j}, \quad \epsilon_{i,j} \sim \mathcal{N}(0, \sigma^2 I)$ 
6:    $\hat{v}_{i,j} \leftarrow \text{softmax}(\tilde{v}_{i,j})$ 
7:    $y_{\text{real}} \leftarrow \{\hat{v}_{i,j}\}$ 
8:   // Generate solution from recursive reasoning model
9:    $\hat{y} \leftarrow f_\theta(x)$ 
10:  // Update critic
11:  Compute critic loss
      
$$\mathcal{L}_{\text{critic}} = -\left(\mathbb{E}_{y \sim P_{\text{real}}}[C_\psi(x, y)] - \mathbb{E}_{\hat{y} \sim P_{\text{gen}}}[C_\psi(x, \hat{y})]\right)$$

12:  Update critic parameters
      
$$\psi \leftarrow \psi - \eta \nabla_\psi \mathcal{L}_{\text{critic}}$$

13:  // Update reasoning model
14:  Compute supervised loss
      
$$\mathcal{L}_{\text{sup}} = \ell(\hat{y}, y^*)$$

15:  Compute adversarial loss
      
$$\mathcal{L}_{\text{adv}} = -\mathbb{E}_{\hat{y} \sim P_{\text{gen}}}[C_\psi(x, \hat{y})]$$

16:  Compute total loss
      
$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sup}} + \lambda \mathcal{L}_{\text{adv}}$$

17:  Update reasoning model
      
$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{total}}$$

18: end while
```

Reasoning model optimization The reasoning model is trained with both supervised learning and adversarial alignment. The adversarial loss encourages generated solutions to obtain higher critic scores:

$$\mathcal{L}_{\text{adv}} = -\mathbb{E}_{\hat{y} \sim P_{\text{gen}}}[C_\psi(x, \hat{y})]. \quad (8)$$

This loss is optimized jointly with the supervised loss, resulting in the final training objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sup}} + \lambda \mathcal{L}_{\text{adv}}. \quad (9)$$

During this step, the reasoning model parameters θ are updated while the critic parameters ψ remain fixed. The complete training procedure is summarized in Algorithm 1.

5 Experiments

5.1 Datasets, model architecture and training

Following Schwarzschild et al. [12], we evaluate DART on easy-to-hard reasoning benchmarks for Deep Thinking Systems (DTS). We additionally reformulate Sudoku [9] as an easy-to-hard benchmark and use it to evaluate DART on Tiny Recursive Models (TRM) under partial observability. Detailed dataset descriptions and implementation details are provided in Appendix A.1.

5.2 Results

We organize the results around three questions. First, does DART improve over instance-level cross-entropy (CE) as the easy-to-hard shift becomes stronger, and does this hold beyond DTS? Second,

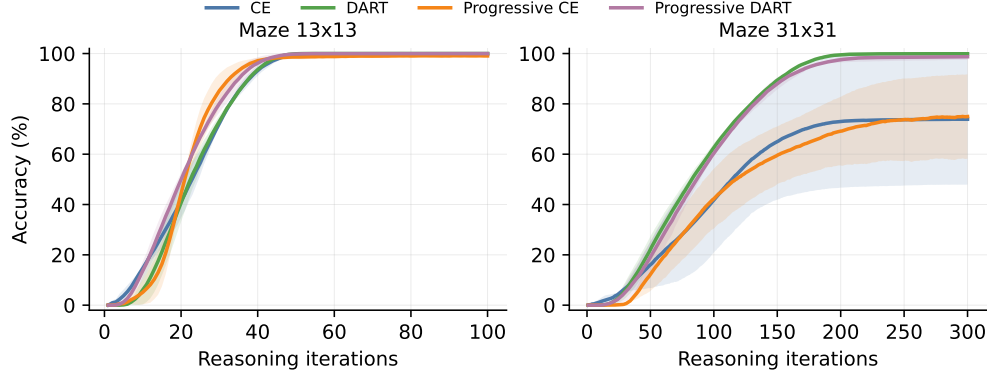


Figure 1: Maze accuracy as the evaluation size increases from 13×13 to 31×31 . The CE failure mode is mostly absent at the smaller size but appears under the harder spatial extrapolation setting.

Table 1: Maze scaling results for standard and progressive training (mean $\pm$ std over seeds).

Method	13×13 Peak	13×13 Final (100)	31×31 Peak	31×31 Final (300)
CE	99.88 ± 0.00	99.85 ± 0.03	73.92 ± 26.01	73.88 ± 25.98
Progressive CE	99.17 ± 0.59	99.06 ± 0.63	75.01 ± 16.69	74.96 ± 16.75
DART	100.00 ± 0.00	100.00 ± 0.00	99.91 ± 0.13	99.91 ± 0.13
Progressive DART	99.98 ± 0.01	99.98 ± 0.01	98.75 ± 1.25	98.73 ± 1.26

how does DART interact with progressive training, which was designed to address long-horizon overthinking? Third, can the gains be explained by simpler target augmentations such as label smoothing or softened targets without adversarial alignment?

5.2.1 DART improves over instance-level CE

Maze exposes how the CE failure mode depends on the severity of the easy-to-hard shift. Models are trained on 9×9 mazes and evaluated at larger sizes. At 13×13 , the shift is mild and CE already reaches $99.88 \pm 0.00\%$ peak accuracy and $99.85 \pm 0.03\%$ final accuracy (Table 1). In this regime, the search space is only moderately larger than the training distribution, so point-wise supervision is still sufficient for the recurrent process to recover the correct path.

The same objective becomes unreliable at 31×31 . CE reaches only $73.92 \pm 26.01\%$ peak accuracy, with a bimodal outcome across seeds: one run nearly solves the task, while another saturates around 47.9%. This indicates that the difficulty is not merely larger input size, but the rapid growth of locally plausible yet globally invalid path configurations. A one-hot target rewards only the exact final path and gives little guidance about nearby partial solutions or how to remain on the feasible path manifold over many recurrent updates. DART removes most of this sensitivity, reaching $99.91 \pm 0.13\%$ peak and final accuracy at 31×31 . The local soft target provides a non-brittle neighborhood around the ground-truth path, while the input-conditioned critic penalizes generated outputs that drift away from maze structure.

Chess shows a different CE failure mode. The model is trained on the easiest 500K puzzles and evaluated on harder puzzles with indices from 600K to 700K. CE reaches a moderate peak of $80.81 \pm 0.37\%$ (Table 4), but by iteration 50 collapses to $0.92 \pm 0.08\%$. Thus CE can briefly find useful move representations, but repeated recurrent refinement destroys them. DART improves peak

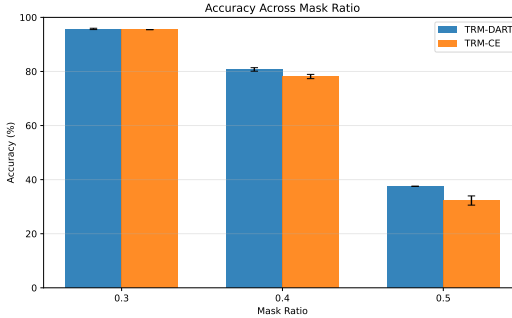


Figure 2: Sudoku accuracy under increasing masking ratios. DART consistently outperforms CE, with larger gains under stronger distribution shift.

Table 2: Test accuracy (%) on Sudoku-Extreme under different masking ratios.

H	L	Depth	Standard CE			DART		
			30%	40%	50%	30%	40%	50%
3	6	42	95.5 ± 0.0	78.1 ± 0.8	32.3 ± 1.7	96.0 ± 0.0	81.4 ± 0.6	37.5 ± 0.1
4	6	56	95.5 ± 0.0	78.2 ± 0.8	32.4 ± 1.7	96.0 ± 0.0	81.5 ± 0.6	38.0 ± 0.1
5	6	70	95.5 ± 0.0	78.3 ± 0.8	32.5 ± 1.7	96.0 ± 0.0	81.6 ± 0.6	38.4 ± 0.1
6	6	84	95.5 ± 0.0	78.3 ± 0.8	32.6 ± 1.7	96.0 ± 0.0	81.6 ± 0.6	38.5 ± 0.1

solution quality to $84.83 \pm 0.03\%$, suggesting that adversarial alignment provides a more global scoring signal for a task where many moves are locally plausible but only a few are optimal. However, DART alone does not fully eliminate late-iteration overthinking on Chess, which motivates the progressive-training comparison below.

Sudoku tests whether the same principle transfers beyond DTS. We evaluate TRM [8] by training with 20% masking and testing on 10,000 puzzles with 30%, 40%, and 50% masking. Table 2 shows that DART consistently outperforms CE at every computation depth and masking ratio. The gap increases as the shift becomes harder: at depth 56, the gain is 0.5% at 30% masking, 3.3% at 40%, and 5.6% at 50%. Figure 2 shows the same trend across masking ratios. Under heavier masking, the model must reason over a larger set of plausible completions rather than recover a nearly deterministic solution. DART improves robustness by aligning predictions toward a local neighborhood around the observed solution instead of a single hard target.

Trajectory-level evidence. To inspect the mechanism behind these aggregate results, we compare representative CE and DART trajectories in Figure 3. IoU measures partial overlap with the target, while exact match identifies iterations where the full discrete solution is correct.

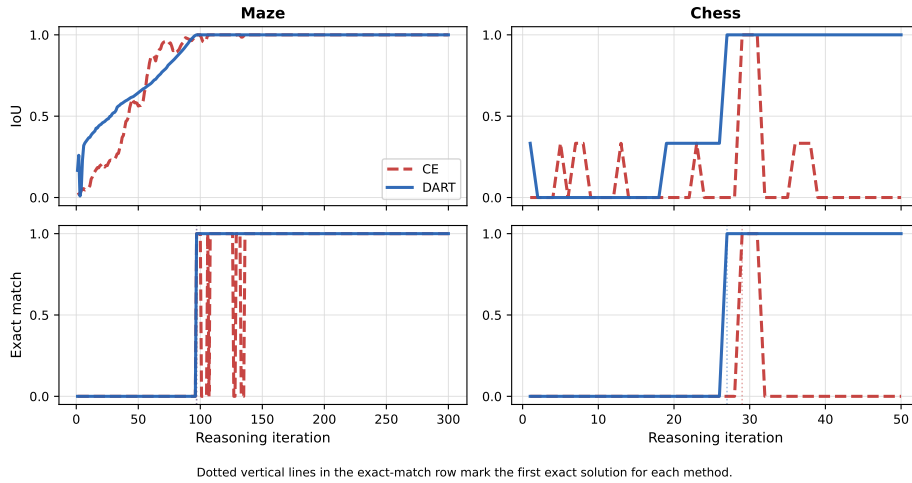


Figure 3: Sample reasoning trajectories for CE and DART on Maze and Chess. The top row shows IoU with the target solution, while the bottom row shows exact match; dotted vertical lines mark the first exact solution. In these examples, DART reaches the exact solution and remains there, whereas CE can enter the correct region transiently and then drift away.

Table 3: Aggregate trajectory statistics 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

On the representative Maze example, CE and DART both first reach exact match at iteration 97, but CE drops out of exact match 11 times afterward whereas DART remains exact for all subsequent iterations. This behavior holds across samples: DART solves and finishes all 128 evaluated 31×31 mazes, while CE solves only 51.6% at least once and finishes at 43.8% exact match (Table 3). On Chess, CE reaches exact match in many samples but almost never preserves it, ending with 0.0% final exact match and only 4.5% exact-match AUC. DART reaches exact solutions more often, reaches them earlier on average (23.1 vs. 28.0 iterations), and retains them much longer, increasing retention from 11.6% to 63.5% and exact-match AUC from 4.5% to 32.3%. This supports the hypothesis that the critic does not simply accelerate convergence, but helps keep recurrent outputs inside an input-conditioned neighborhood of successful traces.

5.2.2 DART and progressive training are complementary

Progressive training and DART address related but distinct failure modes. Progressive training was introduced for DTS to reduce *overthinking*: it changes the training schedule so the same recurrent architecture experiences longer reasoning horizons and learns not to degrade when computation continues. DART instead changes the supervision signal by replacing a single one-hot target with a local soft target distribution and an input-conditioned adversarial constraint. Thus, progressive training primarily targets stability along the iteration axis for DTS, while DART targets imbalance in the valid solution distribution and can be applied at the objective level to different RRM, as shown by our DTS and TRM experiments. This distinction matters for recursive architectures such as TRM and HRM, whose inference already contains nested low-level and high-level loops; directly porting the DTS progressive schedule to such models is not a natural drop-in baseline, whereas DART only requires a structured output distribution and a critic defined on the model’s predictions.

This distinction also means that progressive training should not be expected to solve all easy-to-hard failures. Prior work on stable Deep Thinking models shows that progressive loss alone can still fail to prevent overthinking when the hidden representation is small, motivating architectural constraints on the recurrent map [18]. Our results show a complementary limitation: even when progressive training improves recurrent stability, it may not address the growth of invalid solution configurations under harder problem distributions.

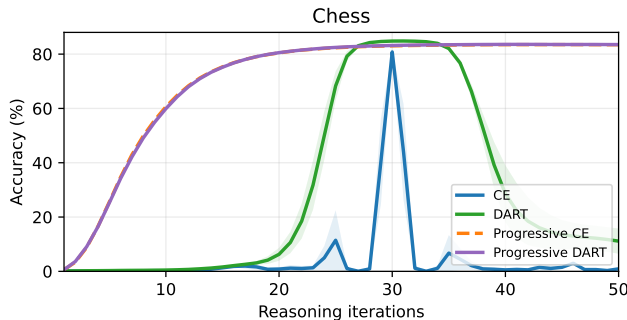


Figure 4: Progressive training baselines on Chess. Progressive training removes the late-iteration collapse of CE and DART, while DART alone reaches the highest peak accuracy.

Table 4: Progressive training baselines on Chess (mean $\pm$ std over 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	84.83 ± 0.03	31, 31	28.58 ± 10.47	11.13 ± 4.64
Progressive DART	83.59 ± 0.09	41, 50	83.58 ± 0.09	83.52 ± 0.16

This distinction is clearest on Maze. At 13×13 , all methods are near perfect because the CE failure mode has not yet appeared. At 31×31 , Progressive CE reduces variance relative to CE but reaches

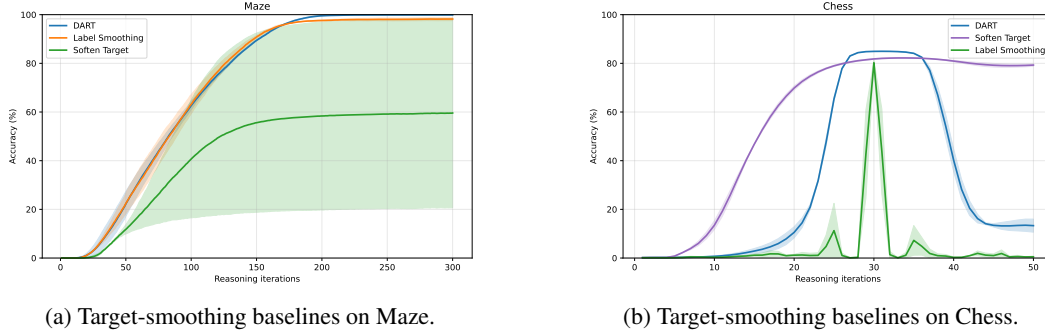


Figure 5: Accuracy versus reasoning iterations for target-smoothing baselines. Gaussian softened targets use the same local target construction as DART but remove the adversarial critic.

Table 5: Target-smoothing baselines on Maze 31×31 (mean $\pm$ std over seeds).

Method	Peak	Acc. @ iter 250	Final (300)
CE	73.92 ± 26.01	73.71 ± 26.17	73.88 ± 25.98
Label smoothing ($\epsilon = 0.3$)	98.22 ± 0.64	98.06 ± 0.57	98.21 ± 0.65
Gaussian softened target, no critic	59.58 ± 38.86	59.14 ± 38.80	59.58 ± 38.86
DART	99.91 ± 0.13	99.89 ± 0.15	99.91 ± 0.13

only $75.01 \pm 16.69\%$ peak accuracy, far below DART’s $99.91 \pm 0.13\%$ (Table 1). This shows that making a DTS less erratic over longer horizons does not by itself solve the larger spatial extrapolation problem: the model still receives point-wise supervision in a solution space where invalid paths grow much faster than valid ones. Progressive DART remains strong at $98.75 \pm 1.25\%$, confirming that the two objectives are compatible, but DART alone already addresses the dominant Maze failure mode.

On Chess, the roles reverse. The main failure is late-iteration collapse: CE falls from an $80.81 \pm 0.37\%$ peak to $0.92 \pm 0.08\%$ at iteration 50, and DART, while reaching the best peak accuracy, still drops to $11.13 \pm 4.64\%$. Figure 4 and Table 4 show that Progressive CE largely removes this overthinking behavior and reaches $83.46 \pm 0.02\%$ at iteration 50. Progressive DART inherits the same long-horizon stability while retaining a distributional objective, achieving the best final accuracy, $83.52 \pm 0.16\%$. The results therefore support a complementary view: DART improves the quality of the solution distribution reached by the recurrent process, and progressive training keeps that quality from degrading as computation continues.

5.2.3 DART versus simple target augmentations

Label smoothing and Gaussian softened targets isolate whether DART’s gains can be explained by simpler target augmentation. Label smoothing mixes the one-hot target with a uniform distribution, reducing overconfidence but ignoring the input-specific structure of valid solutions. The Gaussian softened-target baseline uses the same local target construction as DART, but removes the adversarial critic.

On Maze 31×31 , label smoothing helps substantially, reaching $98.22 \pm 0.64\%$ peak accuracy (Table 5). This confirms that softening brittle one-hot supervision can help CE under spatial extrapolation. However, it remains below DART, which reaches $99.91 \pm 0.13\%$ with much tighter trajectories. More importantly, Gaussian softened targets without the critic are highly unstable, reaching only $59.58 \pm 38.86\%$. Thus, simply constructing a local target neighborhood is not sufficient on Maze; the adversarial critic is needed to make that neighborhood act as a reliable, input-conditioned constraint on generated paths.

On Chess, the pattern is complementary. Label smoothing performs poorly, peaking at $79.29 \pm 0.04\%$ and collapsing almost completely by iteration 50 (Table 6). Uniformly spreading probability mass over incorrect move classes does not capture the structured ambiguity of chess decisions. Gaussian softened targets work better: they reach $82.22 \pm 0.09\%$ peak accuracy and preserve $79.29 \pm 0.49\%$ accuracy at iteration 50. This shows that local target softening can regularize recurrent dynamics

Table 6: Target-smoothing baselines on Chess (mean $\pm$ std over seeds).

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

and reduce collapse. Nevertheless, DART reaches a higher peak of $84.83 \pm 0.03\%$, indicating that adversarial alignment provides an additional signal beyond target smoothing alone.

Overall, the simple augmentations are useful but task-dependent. Label smoothing helps Maze but fails on Chess; Gaussian softened targets stabilize Chess but are unreliable on Maze. DART is more consistent because it combines local target relaxation with a learned input-conditioned critic, giving the model both a smoother target neighborhood and a constraint on whether its generated output distribution remains compatible with the specific problem instance.

6 Discussion

While RRM 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 stabilizing 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 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 distribution-guided adversarial alignment around the ground-truth solution. By modeling logical reasoning as a process of progressively moving predictions toward a local soft target neighborhood, DART provides a richer learning signal than standard cross-entropy and improves the stability of 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. Additional comparisons with label smoothing, Gaussian softened targets, and progressive training show that target softening alone is task-dependent, while adversarial alignment and progressive training address complementary failure modes. These benefits appear consistently across the evaluated DTS and TRM settings, suggesting that distributional training is a promising principle for improving recurrent reasoning systems.

References

- [1] François Chollet. On the measure of intelligence. *arXiv preprint arXiv: 1911.01547*, 2019.
- [2] François Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arcagi-2: A new challenge for frontier ai reasoning systems. *arXiv preprint arXiv: 2505.11831*, 2025.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction, 2025.
- [8] Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv: 2510.04871*, 2025.
- [9] 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.
- [10] Lukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.
- [11] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv: 1410.5401*, 2014.
- [12] 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, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6695–6706. Curran Associates, Inc., 2021.
- [13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.
- [14] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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, pages 97027–97052, 2024.

- [19] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv: 1603.08983*, 2016.
- [20] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv: 1807.03819*, 2018.
- [21] Vijay Veerabadrán, 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, editors, *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.
- [22] Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv: 2107.05407*, 2021.
- [23] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. *International Conference on Machine Learning*, 2018.
- [24] 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.
- [25] Ian Goodfellow et al. Generative adversarial nets. In *NeurIPS*, 2014.
- [26] Ling Yang, Haotian Qian, Zhilong Zhang, Jingwei Liu, and Bin Cui. Structure-guided adversarial training of diffusion models, 2024.
- [27] Manli Shu, Zuxuan Wu, Micah Goldblum, and T. Goldstein. Encoding robustness to image style via adversarial feature perturbations. *Neural Information Processing Systems*, 2020.

A Appendix

A.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 [9].

A.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.

A.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.

A.1.3 Masked Sudoku Dataset

The masked Sudoku dataset is constructed from complete Sudoku solutions provided in the Hard Sudoku benchmark [9]. Each example is represented as a 9×9 grid, where entries take values in $\{1, \dots, 9\}$. 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 the target output.

We use this dataset to define a masking-based easy-to-hard generalization benchmark. During training, 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, requiring the model to infer a larger portion of the solution from global consistency.

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.

A.2 Hyperparameters

Table 7 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 accuracy. For Sudoku, we follow a slightly different model-selection protocol: because the task difficulty is controlled by the masking ratio, we evaluate checkpoints on a validation split constructed with the same masking ratio as the target test setting.

Table 7: Training hyperparameters. 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

A.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 7.

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

A.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 8: Representative Chess training cost on one RTX 5090. GPU-hours 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.

B Additional ablations

B.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 6a 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 6b). 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 region of the valid solution manifold, allowing the model to learn nearby valid 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.

B.2 Effects of Critic Architecture

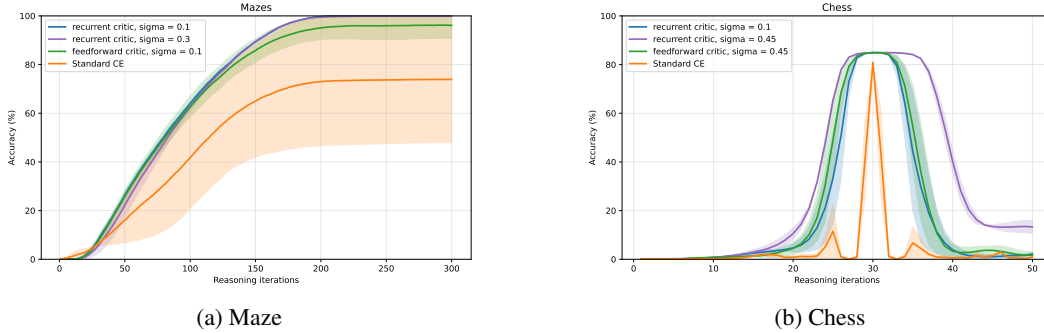


Figure 6: Effect of the target distribution width σ and the critic architecture on reasoning performance.

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 instead of applying one shared block for five iterations. Figure 6a shows that, on Maze, DART with a feedforward critic still substantially outperforms the CE baseline ($96.23 \pm 5.32\%$ vs. $73.92 \pm 26.01\%$), 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. 6b). 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 confirms that distributional alignment is robust across critic designs. 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 stabilizing 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.

B.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 3 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 solution manifold.

The aggregate trajectory statistics in Table 3 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 outputs 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 distributional-alignment interpretation of the method.