TRAIN FOR MANY, UPDATE WITH ONE: IMPROVING MULTI-SAMPLE REASONING VIA FENCHEL–BREGMAN POSTERIOR LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-sample reasoning measures *candidate coverage*, the probability that at least one of K candidates is correct. However, minimizing the average loss of individual attempts does not directly optimize candidate coverage. We therefore formulate *finite-K posterior learning*, which trains a distribution over solver parameters to reduce the probability that all K independently generated candidates fail. To optimize this objective efficiently, we propose a Fenchel–Bregman (FB) approach that requires only one posterior draw per update, rather than the K used by direct Monte Carlo optimization. FB first uses an exact Fenchel reformulation to separate problem-specific weights from the posterior update, enabling one-draw posterior updates. It then applies a closed-form Bregman update to track these weights through an exponential running average of observed failures, reusing the same draw. We instantiate FB with IVON, a variational optimizer for Gaussian posteriors over solver parameters, yielding **Fenchel–Bregman IVON (FBI)**. Across diverse recursive and autoregressive reasoning tasks, FBI improves candidate coverage. In a matched comparison, it also achieves comparable Pass@10 while using only 6.8% of the direct Monte Carlo baseline's measured training time per update[1].

1 INTRODUCTION

Complex reasoning problems often admit several solution paths, and a single attempt may fail even when another succeeds. Multi-sample reasoning generates K candidate solutions for the same problem (Wang et al., 2023; Cobbe et al., 2021). Its *candidate coverage*, commonly measured by Pass@K, is the probability that at least one candidate is correct (Chen et al., 2021; Brown et al., 2024). For a fixed budget K, coverage depends on the distribution that generates the candidates. This motivates learning the candidate-generating distribution rather than treating it as fixed.

We study this candidate-generation problem at the level of solver parameters. Specifically, we learn a distribution $q_\phi(\theta)$ over solver parameters, where each draw instantiates a solver that produces one candidate. Independent draws from q_ϕ form the K-candidate pool used at inference. Recent methods align training with multi-sample inference by optimizing Pass@K and other group-level objectives (Tang et al., 2025; Walder & Karkhanis, 2025). Related approaches optimize multiple moments of problem-level failure (Zhang et al., 2026), limit confidence to preserve repeated-generation coverage (Chen et al., 2025a), or adapt fine-tuning to best-of-N selection (Chow et al., 2025). These methods train a single model to generate candidates for multi-sample inference, whereas we train a distribution over solver parameters for candidate coverage under a fixed budget K.

A natural way to train q_ϕ, as in standard variational learning, is to minimize the expected loss of an individually sampled solver (Blundell et al., 2015; Shen et al., 2024). This treats each draw in isolation, while multi-sample reasoning succeeds when at least one of K draws solves the problem. We address this mismatch by training q_ϕ directly for the coverage of the K-candidate pool and call this formulation *finite-K posterior learning* (Sec. 2). Finite-K posterior learning minimizes the probability that the entire K-draw candidate pool fails. The resulting objective assigns a greater learning weight to problems that the other candidates are unlikely to solve. We call the direct Monte Carlo estimator of this failure probability *Direct-K*. It is unbiased, but requires K independent posterior draws per update.

[1]Code is available at https://anonymous.4open.science/r/fbi.

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

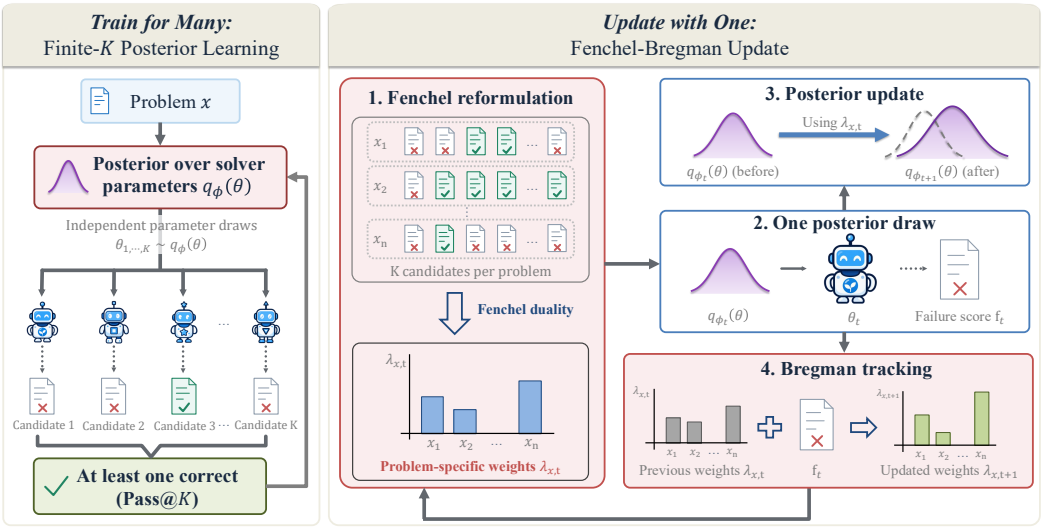


Figure 1: **Overview of finite-K posterior learning and FBI.** Left: K independent posterior draws generate a candidate pool, and training targets the probability that all K candidates fail. Right: Fenchel reformulation enables a one-draw IVON posterior update with a fixed problem-specific weight, while Bregman tracking reuses the observed failure to update the weight for the next visit.

To address this cost, we propose a **Fenchel–Bregman (FB)** optimization approach for finite-K posterior learning using one posterior draw per update. Using Fenchel duality, FB rewrites the objective so that the posterior can be updated using a weighted loss evaluated from one posterior draw. This reformulation strategy is well established in compositional optimization and learning objectives (Wang et al., 2017; Devraj & Chen, 2019; Mohri et al., 2026), and we use it here to enable one-draw posterior updates for finite-K candidate coverage. As the posterior changes, a closed-form Bregman update adjusts the weights using past failure estimates and the current solver outcome (Sec. 3.2). The same draw used for the posterior update also supplies the observation for the next weight update, avoiding additional posterior draws. We instantiate FB with IVON (Shen et al., 2024), yielding **Fenchel–Bregman IVON (FBI)** (Sec. 3.3). Fig. 1 summarizes this procedure.

We evaluate FBI across recursive reasoning benchmarks with multiple solver backbones and mathematical reasoning with an autoregressive language model. On a matched comparison, finite-K posterior learning improves coverage over matched training that optimizes individual draws. In the same setting, FBI achieves comparable coverage to Direct-K with substantially lower training computation. Together, these results support aligning posterior training with multi-sample inference using only one new posterior draw per update.

To summarize, our contributions are the following:

- We formulate finite-K posterior learning, aligning a posterior over solver parameters with the success of the finite candidate pool used at inference.
- We propose FB for one-draw posterior updates using an exact Fenchel reformulation and closed-form Bregman update for weight tracking. We instantiate FB with IVON as FBI.
- We show improved candidate coverage across recursive and autoregressive reasoning, and substantially lower training cost than Direct-K in a matched comparison.

2 Train for Many: Finite-K Posterior Learning

We now formalize the candidate-generating distribution underlying multi-sample reasoning and derive a training objective aligned with a finite candidate pool. We first describe multi-sample reasoning under posterior parameter sampling in Sec. 2.1, then formulate finite-K posterior learning in Sec. 2.2, and finally characterize the learning signal it induces in Sec. 2.3.

2.1 Multi-Sample Reasoning with Posterior Draws

Multi-sample reasoning. Complex reasoning problems can admit multiple solution paths, and success may vary across attempts. Multi-sample reasoning exploits this variation by generating K candidate solutions for the same problem, usually from one solver model with parameters θ. Candidate coverage, commonly measured by $\mathrm{Pass@}K$, is the probability that at least one of these candidates is correct.

Posterior candidate generation. For a fixed candidate budget K, coverage depends on how the candidate-generating distribution allocates probability across possible solutions. This makes the distribution over candidates a natural object to optimize. We represent this distribution through a learned distribution $q_\phi(\theta)$ over the parameters of a solver neural network. We then draw K different solvers from this distribution and one candidate from each of these solvers. That is, the candidate pool used at inference consists of a total of K outputs with one generated from each of the K parameter draws $\theta_k \sim q_\phi$. We seek to train q_ϕ for the collective success of this K-candidate pool.

2.2 Finite-K Posterior Learning

From individual draws to candidate pools. When learning q_ϕ, a natural objective is to minimize the expected loss of solvers sampled from this distribution, as for instance, a standard variational Bayesian approach would suggest. However, this treats each draw individually, whereas multi-sample inference evaluates whether at least one of K sampled solvers succeeds. To align training with this inference criterion, we propose to optimize q_ϕ directly for K-candidate coverage. We call this formulation *finite-K posterior learning*.

Finite-K objective. For problem x, let $F(\theta, x) \in \{0, 1\}$ denote the exact failure indicator for one draw, with $F = 1$ for failure. Write $\bar{F}_\phi(x) \coloneqq \mathbb{E}_{\theta\sim q_\phi}[F(\theta, x)]$ for the failure probability under q_ϕ. We draw $\theta_{1:K}$ independently from q_ϕ. Additional solver randomness is independent across candidates and included in all expectations. We omit it from the notation for readability. Since the K candidate draws are independent, the probability that they all fail is

$$\Pr(\text{all } K \text{ candidates fail} \mid x) = \prod_{k=1}^{K} \Pr(F(\theta_k, x) = 1) = \bar{F}_\phi(x)^K. \tag{1}$$

Since candidate coverage is the complement of the all-fail event, averaging over the empirical distribution of training problems gives $\mathrm{Pass@}K(\phi) = 1 - \mathbb{E}_x[\bar{F}_\phi(x)^K]$. Hence, maximizing coverage is equivalent to minimizing $\mathbb{E}_x[\bar{F}_\phi(x)^K]$.

Optimizing candidate coverage on the training problems alone may favor a highly concentrated parameter distribution. We therefore add KL regularization toward a fixed reference prior $p_0(\theta)$, discouraging collapse toward a point estimate while controlling the spread of q_ϕ. This yields the proposed finite-K posterior-learning objective:

$$\boxed{\min_\phi \ \mathbb{E}_x\left[\bar{F}_\phi(x)^K\right] + \tau_{\mathrm{KL}} D_{\mathrm{KL}}(q_\phi \,\|\, p_0).} \tag{2}$$

The first term minimizes the expected all-fail probability, while $\tau_{\mathrm{KL}} \geq 0$ controls regularization toward the reference prior p_0.

2.3 Understanding and Optimizing the Finite-K Objective

Problem-dependent learning signal. To see how the objective in Eq. 2 differs from an individual-draw objective, we examine the per-problem gradient of its data term. When $\bar{F}_\phi(x)$ is differentiable,

$$\nabla_\phi \bar{F}_\phi(x)^K = \underbrace{K\bar{F}_\phi(x)^{K-1}}_{\lambda_x^\star} \nabla_\phi \bar{F}_\phi(x). \tag{3}$$

We denote this multiplicative factor by $\lambda_x^\star$, the problem-specific gradient weight induced by the objective in Eq. 2. For $K \geq 2$, $\bar{F}_\phi(x)^{K-1}$ is the probability that the other $K-1$ candidates all fail. Problems that the remaining candidates are unlikely to solve therefore receive a larger gradient weight, whereas problems already likely to be solved by another candidate receive a smaller one. When $K = 1$, $\lambda_x^\star = 1$, so this problem-dependent reweighting disappears.

Naive Direct-K estimation. For each problem x, K independent posterior draws satisfy

$$\mathbb{E}_{\theta_{1:K}\sim q_\phi}\left[\prod_{k=1}^{K} F(\theta_k, x)\right] = \bar{F}_\phi(x)^K.$$

Thus, $\prod_{k=1}^{K} F(\theta_k, x)$ is an unbiased estimator of the per-problem data term in Eq. 2. We call this direct estimator *Direct-K*. Using Direct-K to estimate the expected all-fail probability in Eq. 2 requires K posterior draws per update, increasing the computational cost of each update. App. A.1 provides the detailed derivation and the unbiasedness of this estimator. The key computational challenge is to preserve the learning signal in Eq. 3 while using only a single new posterior draw per update. In the following Sec. 3 we develop an efficient one-draw update for this purpose.

3 Update with One: Fenchel–Bregman IVON

In Sec. 2 we have defined the finite-K learning signal and showed that Direct-K estimates the data term in Eq. 2 using K independent posterior draws per update. Our goal is to retain this learning signal while using only one new posterior draw per update to improve training efficiency. In the following Sec. 3.1 we will use a *Fenchel reformulation* to separate the problem-specific weight from the posterior update, making a one-draw posterior update possible once the weight is fixed. Sec. 3.2 then uses *Bregman tracking* to update this weight from the observed failure without additional posterior draws. Finally, Sec. 3.3 instantiates these updates with the variational learning algorithm IVON of Shen et al. (2024) to optimize Eq. 2 for the distribution parameters ϕ. We call our final proposed method **Fenchel–Bregman IVON (FBI)**.

3.1 Fenchel reformulation and one-draw posterior update

Why direct estimation needs K draws. Direct-K uses K independent posterior draws to estimate the gradient of $\bar{F}_\phi(x)^K$. Reusing a single draw does not preserve this quantity because, for binary F,

$$\mathbb{E}_{\theta\sim q_\phi}[F(\theta, x)^K] = \bar{F}_\phi(x),$$

which generally differs from $\bar{F}_\phi(x)^K$. Eq. 3, however, shows that the desired data gradient has the form $\lambda_x^\star \nabla_\phi \bar{F}_\phi(x)$. This suggests the possibility of separating the problem-specific weight from the posterior update so that the posterior can be updated from a single draw once the weight is fixed.

Fenchel reformulation. Fenchel conjugacy (Rockafellar, 1970) provides this separation by expressing the convex function r^K as a maximization over weighted linear functions of $r \in [0, 1]$, which we set to $r = \bar{F}_\phi(x)$. For $K \geq 2$, we have the following representation,

$$r^K = \max_{\lambda\in[0,K]} \lambda r - \psi_K(\lambda), \qquad \lambda^\star(r) = K r^{K-1}, \tag{4}$$

where $\psi_K(\lambda) = (K-1)(\lambda/K)^{K/(K-1)}$ defined on $[0, K]$ is the so-called Fenchel conjugate of r^K. Such representations exist for any convex function (Rockafellar, 1970) and for completeness, we recall in App. A.2 that the identity is exact and that $\lambda^\star(r)$ is its unique maximizer.

Applying Eq. 4 to $r = \bar{F}_\phi(x)$ in Eq. 2 introduces one weight $\lambda_x \in [0, K]$ for each training problem as a dual variable that acts as a problem-specific weight, giving the equivalent saddle-point objective

$$\min_\phi \max_{\lambda_x\in[0,K]} \mathbb{E}_x\left[\mathbb{E}_{\theta\sim q_\phi}[\lambda_x F(\theta, x)] - \psi_K(\lambda_x)\right] + \tau_{\mathrm{KL}} D_{\mathrm{KL}}(q_\phi \,\|\, p_0). \tag{5}$$

Eq. 5 separates ϕ and λ_x into different optimization variables. Holding λ_x fixed therefore defines a posterior-update subproblem whose ϕ-dependent data term is simply $\mathbb{E}_{\theta\sim q_\phi}[\lambda_x F(\theta, x)]$, while the KL term is unchanged. The maximizing weight is $\lambda_x^\star = K\bar{F}_\phi(x)^{K-1}$, exactly the problem-specific gradient weight in Eq. 3. For $\tau_{\mathrm{KL}} > 0$, this fixed-weight posterior subproblem has the generalized Bayesian variational form of Bissiri et al. (2016) which can be solved, for example, with the IVON optimizer of Shen et al. (2024).

Fixed-weight one-draw posterior update. Since λ_x is a separate variable in Eq. 5, we can hold it fixed while taking a posterior step in ϕ. For fixed λ_x,

$$\nabla_\phi \mathbb{E}_{\theta\sim q_\phi}[\lambda_x F(\theta, x)] = \lambda_x \nabla_\phi \bar{F}_\phi(x), \qquad \lambda_x = \lambda_x^\star \quad \Longrightarrow \quad \nabla_\phi \bar{F}_\phi(x)^K.$$

Under standard conditions, one posterior draw gives an unbiased estimate of this fixed-weight gradient, and at $\lambda_x = \lambda_x^\star$ it recovers the finite-K data gradient in Eq. 3; see App. A.3. The posterior step can therefore use one draw. The remaining problem is to obtain a suitable λ_x without additional draws.

3.2 Bregman update for the dual weight

Why the weight must be tracked. The optimal weight $\lambda_x^\star = K\bar{F}_\phi(x)^{K-1}$ changes as q_ϕ changes. Re-estimating the current $\bar{F}_\phi(x)$ from multiple additional posterior draws at every update would remove the one-draw advantage. At update step t, let $\lambda_{x,t}$ denote the weight used for problem x and let $F_t = F(\theta_t, x)$ denote the failure observed from the single draw θ_t. We seek an update from $\lambda_{x,t}$ to $\lambda_{x,t+1}$ using F_t, rather than recomputing $\lambda_x^\star$ from additional draws.

Bregman tracking. To update the dual weight λ_x, we use a Bregman update generated by the convex function ψ_K in the Fenchel representation (Bregman, 1967). This Bregman update can equivalently be viewed as a mirror-ascent step with ψ_K as the mirror potential. App. A.4 gives the proximal objective and its derivation. The resulting update reduces to a simple exponential running average of the observed failures:

$$\widehat{F}_{x,t+1} = \beta \widehat{F}_{x,t} + (1-\beta)F_t, \qquad \lambda_{x,t+1} = K\widehat{F}_{x,t+1}^{K-1}, \tag{6}$$

where $\beta \in (0,1)$ controls how strongly past failure estimates are retained in the exponential running average. The updated running estimate $\widehat{F}_{x,t+1}$ aggregates observed failures, and $K\widehat{F}_{x,t+1}^{K-1}$ gives the next problem-specific weight. The update is closed form and uses no additional posterior draw.

3.3 Posterior updates with IVON

Failure score and gradient estimation. Practical training uses a bounded failure score $f(\theta, x) \in [0,1]$, with posterior mean $\bar{f}_\phi(x) := \mathbb{E}_{\theta\sim q_\phi}[f(\theta, x)]$. The score may be a differentiable surrogate for binary failure or be estimated from sampled binary outcomes. When f differs from the binary failure indicator F, this yields a score-based surrogate of the exact finite-K objective in Eq. 2. Accordingly, the fixed-weight posterior gradient can be obtained either by backpropagation through f or by a stochastic gradient estimator. In both cases, the resulting failure observations are also used to update the running estimate $\widehat{f}$.

Algorithm 1 Fenchel–Bregman IVON (FBI) for finite-K posterior learning.

Require: Training set $\mathcal{D}$, diagonal Gaussian posterior q_{ϕ_0}, prior p_0, pool size $K \geq 2$, KL coefficient τ_{KL}, EMA coefficient $\beta \in (0,1)$, failure score f and initial failure estimate $\widehat{f}_{\mathrm{init}} \in [0,1]$

1: Initialize the IVON posterior mean and diagonal precision from ϕ_0, and $\widehat{f}_{x,0} \leftarrow \widehat{f}_{\mathrm{init}}$ for every $x \in \mathcal{D}$
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Sample a minibatch $\mathcal{B}_t$ from $\mathcal{D}$
4: **Form weight:** $\lambda_{x,t} \leftarrow K\widehat{f}_{x,t}^{K-1}$ for each $x \in \mathcal{B}_t$
5: **Draw once:** Draw $\theta_t \sim q_{\phi_t}$ and evaluate $f(\theta_t, x)$ for each $x \in \mathcal{B}_t$
6: **Estimate gradient:** Compute a gradient estimate of Eq. 7 ▷ fixed $\lambda_{x,t}$
7: **Update posterior:** Apply IVON from ϕ_t to ϕ_{t+1} using the gradient estimate and KL regularization
8: **Update failure estimate:** $\widehat{f}_{x,t+1} \leftarrow \beta\widehat{f}_{x,t} + (1-\beta)f(\theta_t, x)$ for each $x \in \mathcal{B}_t$ ▷ others unchanged
9: **end for**

IVON posterior. We instantiate the posterior update with IVON, which learns a diagonal Gaussian posterior (Shen et al., 2024). We write $q_\phi(\theta) = \mathcal{N}(\theta; \mathbf{m}, \mathrm{diag}(\boldsymbol{\sigma}^2))$, with mean $\mathbf{m}$ and diagonal precision $\boldsymbol{\sigma}^{-2}$, so IVON learns both the mean and the parameter-wise sampling scales. Under our batch-mean loss convention, IVON's effective sample size (ESS) sets the KL coefficient as $\tau_{\mathrm{KL}} = 1/\mathrm{ESS}$. ESS and weight decay jointly determine the implied Gaussian prior, so we rescale them inversely when varying the KL strength to keep the prior fixed.

One-draw IVON update. At update t, FBI first forms $\lambda_{x,t} = K\widehat{f}_{x,t}^{K-1}$ and then draws one $\theta_t \sim q_{\phi_t}$. For a minibatch $\mathcal{B}_t$, the fixed-weight data loss is given as follows:

$$\mathcal{L}_t^{\text{data}}(\theta_t) = \frac{1}{|\mathcal{B}_t|} \sum_{x \in \mathcal{B}_t} \lambda_{x,t} f(\theta_t, x). \tag{7}$$

IVON uses a gradient estimate of Eq. 7, together with KL regularization to update the posterior mean and diagonal precision. Since $\lambda_{x,t}$ is fixed during this posterior update, the Fenchel penalty $\psi_K(\lambda_{x,t})$ is constant and omitted from Eq. 7. The same failure observations then update $\widehat{f}_{x,t+1}$ through Eq. 6, so one posterior draw both updates q_ϕ and supplies the feedback used to prepare the next weight.

Overall, Alg. 1 follows a stochastic alternating primal–dual optimization for Eq. 5. With the dual weights fixed, IVON updates the posterior parameters ϕ. The same failure observations are then used in the Bregman update of the dual weights, which is equivalently a mirror-ascent step on Eq. 5.

4 Experiments

We evaluate whether FBI improves candidate coverage while reducing the training cost of finite-K posterior learning. K denotes the number of candidates at evaluation (the K in Pass@K), whereas K_{train} denotes the candidate count targeted by training. We first report benchmark results and candidate coverage across evaluation budgets for recursive reasoning models and an autoregressive language model (Secs. 4.1 and 4.2). We then analyze the training target, problem-specific weighting, failure recovery, and one-draw training cost (Sec. 4.3). Detailed protocols and hyperparameters are in App. B, and sensitivity analyses are in App. C.

4.1 Recursive reasoning models

Setup and baselines. We evaluate FBI with recursive reasoning models on the Sudoku-Extreme, Maze-Hard, ARC-AGI-1, and ARC-AGI-2 tasks. Sudoku-Extreme requires completing difficult 9×9 puzzles, while Maze-Hard requires finding optimal paths in 30×30 mazes. The ARC-AGI benchmarks require inferring a grid transformation from a few examples and applying it to a new input (Chollet, 2019; Chollet et al., 2025). We report candidate coverage and, where applicable, selected accuracy, which is defined as the exact-solution accuracy of each method's selector.

Our baselines include the Hierarchical Reasoning Model (HRM) (Wang et al., 2025), Tiny Recursive Model (TRM) (Jolicoeur-Martineau, 2025), Fixed-Point Reasoning Model (FPRM) (Movahedi et al., 2026), and Probabilistic Tiny Recursive Model (PTRM) (Sghaier et al., 2026). We also include the fixed-Gaussian weight-sampling controls W-PTRM and W-FPRM. FBI uses TRM in the main benchmark comparison and FPRM in the additional backbone comparison. The learned-posterior comparisons always use the IVON optimizer (Shen et al., 2024). Matched runs share the backbone, posterior initialization, training data, update budget, and evaluation budget. For FBI, the Acc. and Pass@10 values in Tables 1 and 2 are means over three independent training runs.

Table 1: **Selected accuracy (Acc.) and candidate coverage (Pass@K) across four recursive reasoning benchmarks.** All values are percentages. **Bold** and underlined numerical values mark the highest and second-highest values in each metric column, with ties marked equally. † denotes previously reported performance. FBI achieves the best results across all datasets and metrics.

	Sudoku-Extreme		Maze-Hard		ARC-AGI-1			ARC-AGI-2		
Method	Acc. ↑	Pass@10 ↑	Acc. ↑	Pass@10 ↑	Pass@1 ↑	Pass@2 ↑	Pass@100 ↑	Pass@1 ↑	Pass@2 ↑	Pass@100 ↑
HRM†	55.00	–	74.50	–	–	40.30	–	–	5.00	–
FPRM	94.33	–	86.60	–	–	**47.50**	–	–	6.20	–
TRM	87.28	–	87.00	–	40.00	44.38	59.50	5.00	7.50	11.53
PTRM	97.36	97.38	87.70	88.40	40.50	44.88	60.25	5.00	7.50	11.94
W-PTRM	98.83	98.84	87.50	89.80	40.37	45.25	61.25	5.00	7.08	10.69
FBI (Ours)	**99.53**	**99.55**	**88.00**	**91.30**	**40.63**	**47.50**	**64.25**	**5.42**	**8.19**	**12.78**

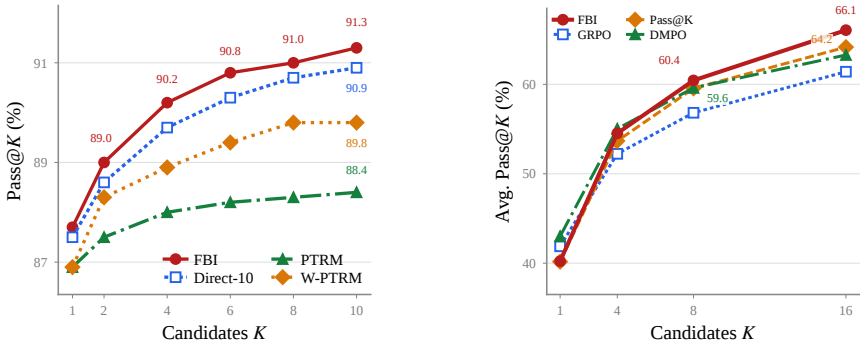
Results. FBI improves candidate coverage and selected accuracy on Sudoku-Extreme and Maze-Hard (Table 1). On Sudoku-Extreme, FBI nearly matches its Pass@10 of 99.55% with 99.53% selected accuracy. On Maze-Hard, FBI raises Pass@10 from W-PTRM's 89.80% to 91.30%, while reaching 88.00% selected accuracy. The remaining gap on Maze-Hard shows that some correct candidates are still not returned by the selector. FBI also improves candidate coverage after aggregation and ranking on ARC-AGI-1 and ARC-AGI-2. It achieves the highest Pass@K across the evaluated values of K.

Table 2: **Performance across recursive backbones on Maze-Hard.** FBI improves performance for both TRM and FPRM.

Method	Acc. ↑	Pass@10 ↑
TRM-based methods		
TRM	87.00	–
PTRM	87.70	88.40
W-PTRM	87.50	89.80
FBI (Ours)	**88.00**	**91.30**
FPRM-based methods		
FPRM	86.60	–
W-FPRM	86.90	88.70
FBI (Ours)	**87.70**	**94.80**

Performance across recursive backbones. Table 2 tests whether the Maze-Hard gains extend from TRM to FPRM, using ten candidates and each method's selector. With FPRM, FBI reaches 87.70% selected accuracy and 94.80% Pass@10, compared with 86.90% and 88.70% for W-FPRM. Together with the TRM results, this shows that the gains over fixed-Gaussian weight sampling obtained using a learned posterior are not limited to one recursive backbone.

Candidate coverage. We examine coverage across candidate budgets in recursive models. Here, the coverage panels compare equal numbers of candidate attempts instead of equal wall-clock times or FLOP budgets across models. We compare FBI with Direct-10, PTRM, and W-PTRM to assess coverage under fixed evaluation budgets. FBI and Direct-10 both target $K_{\text{train}} = 10$, but use one and ten posterior draws per training update, respectively. We include Direct-10 to compare FBI's candidate coverage and training cost against a direct estimator of the same finite-K_{train} target. At test time, all four methods generate multiple candidates with the same TRM backbone and the same reasoning depth of 16. Fig. 2a compares coverage at $K \in \{1, 2, 4, 6, 8, 10\}$. FBI reaches the Pass@10 of PTRM, W-PTRM, and Direct-10 using 80%, 60%, and 20% fewer candidates, respectively. Across the budgets, the reported FBI run is 0.2–0.5 percentage points above Direct-10.



(a) **Maze-Hard.** (b) **Mathematical reasoning.**

Figure 2: **Candidate coverage across recursive and autoregressive models.** (a) Pass@K on 1,000 Maze-Hard test problems using TRM at depth 16. (b) Mean Pass@K across five mathematical-reasoning benchmarks using Qwen3-4B-Base. FBI improves candidate coverage in both settings.

4.2 Autoregressive Language Models

Setup and baselines. We use Qwen3-4B-Base to test FBI beyond recursive solvers. Following the mathematical-reasoning setup of MMPO (Zhang et al., 2026), we train on MATH7.5K (Hendrycks et al., 2021) and evaluate on MATH500 (Lightman et al., 2024), OlymMATH (Sun et al., 2026), AMC23 (MAA, 2023), AIME24 (MAA, 2024), and AIME25 (MAA, 2025). Each FBI candidate uses an independent parameter draw from the learned posterior, followed by autoregressive sampling. FBI places a diagonal IVON posterior over all actor parameters and trains with $K_{\text{train}} = 8$, using one posterior draw per optimizer update. Each update uses one posterior parameter draw shared across eight responses per prompt. App. B.4 provides the implementation details.

We compare FBI with Base and five baselines. GRPO (Shao et al., 2024) maximizes mean reward with group-relative advantages. Pass@K Training (Chen et al., 2025b) optimizes group success using

analytic response-level advantages. DMPO (Li et al., 2026) matches a reward-proportional group distribution to preserve diversity. MaxRL (Tajwar et al., 2026) uses compute-indexed objectives approaching maximum likelihood as the budget grows. MMPO (Zhang et al., 2026) minimizes failure-probability moments to target truncated time to first success.

Table 3: **Mathematical reasoning performance with Qwen3-4B-Base.** Columns report Pass@8 (%); **Avg.** is the unweighted mean across five benchmarks. **Bold** and underlined values mark the best and second-best results per column. FBI achieves the highest average Pass@8 (60.44%), 0.84 points above DMPO, and leads on AMC23 and AIME25.

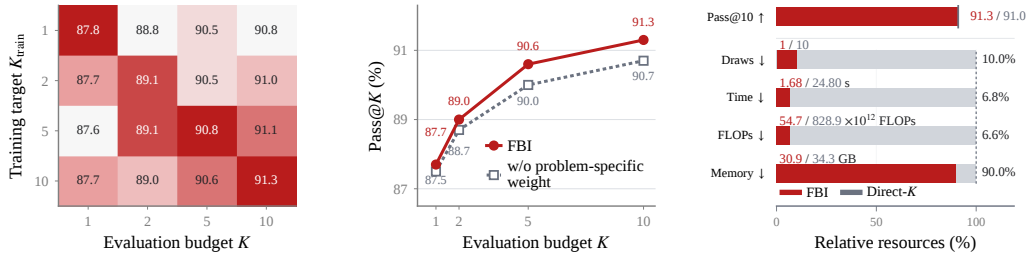
Method	MATH500	OlymMATH	AMC23	AIME24	AIME25	**Avg.**
Base	87.80	53.57	75.18	34.15	21.56	54.45
GRPO	90.17	61.96	79.07	27.96	24.95	56.82
Pass@K Training	**92.48**	62.24	82.64	34.64	25.76	59.55
DMPO	91.37	61.45	84.68	**34.65**	25.84	59.60
MaxRL	91.23	**62.26**	83.07	28.08	30.21	58.97
MMPO	89.91	58.99	81.87	26.89	25.17	56.56
FBI (Ours)	91.64	60.68	**85.20**	31.00	**33.70**	**60.44**

Mathematical reasoning performance. FBI delivers the strongest aggregate Pass@8 performance in Table 3, with an unweighted average of 60.44%. Beyond the aggregate result, FBI achieves the best scores on AMC23 and AIME25.

Candidate coverage. Fig. 2b compares FBI to GRPO, Pass@K Training, and DMPO using average Pass@K across the same five benchmarks. At $K = 1$, FBI reaches 40.22%, below DMPO at 43.01%. The gap narrows at $K = 4$, and FBI overtakes all three baselines at $K = 8$ with 60.44%, 0.84 percentage points above DMPO. At $K = 16$, FBI reaches 66.06%, 1.90 percentage points over Pass@K Training.

4.3 Analysis of finite-K posterior learning

We first examine how the training target and problem-specific weights affect candidate coverage on Maze-Hard. We then analyze failure recovery and one-draw training cost.



(a) **Training target K_{train}.** (b) **Problem-specific weights.** (c) **Training cost.**

Figure 3: **Finite-K posterior learning and training efficiency.** (a) The diagonal pattern shows that $K_{\text{train}} = K$ attains the highest reported Pass@K at each budget. Cells show Pass@K (%); darker shades indicate higher within-column coverage. (b) Problem-specific weights outperform their minibatch-mean replacement at $K_{\text{train}} = 10$. (c) At the same target, FBI achieves comparable coverage to Direct-10 using 6.8% of its one-GPU training time per update. Resources are relative to Direct-10 (100%); FLOPs are in units of 10^{12}.

Effect of K_{train} on candidate coverage. We examine how the training target $K_{\text{train}} \in \{1, 2, 5, 10\}$ affects Pass@K across evaluation budgets on Maze-Hard, retaining one posterior draw per training update (Fig. 3a). The columnwise rankings exhibit a diagonal pattern: matching K_{train} to the evaluation budget K gives the highest reported Pass@K at every tested budget. At $K = 2$, the matched target ties with $K_{\text{train}} = 5$ at 89.1%. Increasing K_{train} from 1 to 10 raises Pass@10 from

90.8% to 91.3%, while Pass@1 changes from 87.8% to 87.7%. This pattern supports choosing K_{train} according to the intended evaluation budget and shows that the training target affects candidate-pool coverage with little change in single-candidate accuracy.

Role of problem-specific weighting. We test the contribution of problem-specific weighting at $K_{\text{train}} = 10$ by replacing each problem's weight with the minibatch mean, removing differences between problems while preserving the mean weight scale (Fig. 3b). Removing these differences lowers coverage at every tested budget: Pass@1 falls from 87.7% to 87.5%, and Pass@10 from 91.3% to 90.7%. FBI's gain grows from 0.2 percentage points at $K = 1$ to 0.6 points at $K = 5$ and 10. These results support a contribution from problem-specific weighting to candidate coverage, with larger gains when evaluating multiple candidates.

Recovery after first-candidate failures. We analyze recovery among candidates 2–10 after each method's first-candidate failures on one fixed Maze-Hard seed (Figure 4). FBI recovers an exact solution in 32.5% of these cases (40/123), compared with 27.2% for Direct-10, 22.1% for W-PTRM, and 11.5% for PTRM. This recovery raises FBI's coverage from 87.7% Pass@1 to 91.7% Pass@10. FBI also has a lower repeated-output rate (56.9% versus 64.0–87.8% for the baselines), so later candidates more often change the output and recover an exact solution. The weight ablation (Figure 3b) supports problem-specific weighting's contribution to coverage, but does not isolate its effect on conditional recovery.

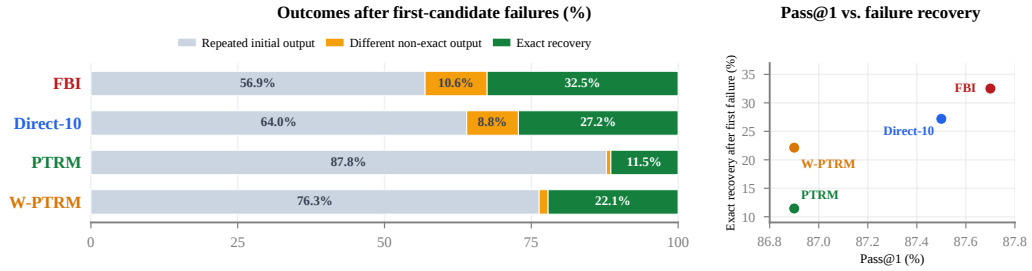


Figure 4: **Recovery after first-candidate failures on Maze-Hard.** Left: first-candidate failures are partitioned into three outcomes: only repeated initial outputs, different outputs with no exact solution, or at least one exact solution. Right: Pass@1 versus the exact-recovery rate among candidates 2–10 after each method's first candidate fails. FBI has the highest conditional exact-recovery rate of 32.5%.

One-draw training cost. FBI achieves comparable mean Maze-Hard Pass@10 to Direct-10 across repeated runs at $K_{\text{train}} = 10$. Fig. 3c compares their one-GPU training resources per update. FBI uses one posterior draw per update, compared with ten for Direct-10. FBI uses 6.8% of Direct-10's measured training time and 6.6% of its FLOPs per update. Peak allocated GPU memory is 30.9 GB for FBI and 34.3 GB for Direct-10. App. B.5 gives the one-GPU measurement details and an additional eight-GPU comparison.

5 Conclusion

Fenchel–Bregman IVON (FBI) uses an exact reformulation and Bregman tracking for one-draw finite-K posterior learning, improving candidate coverage across recursive and autoregressive reasoning. On matched Maze-Hard, FBI reaches comparable Pass@10 to Direct-10 using 10.0% as many posterior draws and 6.8% of the measured training time per update. On Qwen3-4B-Base, FBI attains the highest mean Pass@8 across five mathematical-reasoning benchmarks. In the failure analysis, later FBI candidates recover an exact solution after 32.5% of first-candidate failures, suggesting that the coverage gain includes recovery after initial errors. However, surrogate mismatch, stochastic gradient noise, or posterior drift may weaken failure tracking. Improving surrogate calibration where applicable and using predictive or variance-reduced tracking could make that signal more reliable. Adaptive or correlated candidate pools and selector-aware training also could better match inference, while amortized estimates for unseen problems and richer posteriors could broaden applicability.

AI USE STATEMENT

We used OpenAI GPT and Anthropic Claude to improve the clarity and readability of the manuscript, refine research ideas, and assist in developing mathematical derivations and proofs. The authors reviewed all AI-assisted material and checked the mathematical arguments step by step, including their assumptions and derivations. The authors take full responsibility for the final content of this work, including all AI-assisted text and mathematical claims.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. In Annual Meeting of the Association for Computational Linguistics (ACL), 2024. URL https://aclanthology.org/2024.acl-long.662/.

Junyeob Baek, Mingyu Jo, Minsu Kim, Mengye Ren, Yoshua Bengio, and Sungjin Ahn. Generative recursive reasoning, 2026. URL https://arxiv.org/abs/2605.19376. arXiv:2605.19376.

Pier Giovanni Bissiri, Chris C. Holmes, and Stephen G. Walker. A general framework for updating belief distributions. J. R. Stat. Soc. Ser. B Stat. Methodol., 78(5):1103–1130, 2016. doi: 10.1111/rssb.12158.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In International Conference on Machine Learning (ICML), 2015. URL https://proceedings.mlr.press/v37/blundell15.html.

Lev M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. USSR Comput. Math. Math. Phys., 7(3):200–217, 1967. doi: 10.1016/0041-5553(67)90040-7.

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, 2024. URL https://arxiv.org/abs/2407.21787. arXiv:2407.21787.

Feng Chen, Allan Raventós, Nan Cheng, Surya Ganguli, and Shaul Druckmann. Rethinking fine-tuning when scaling test-time compute: Limiting confidence improves mathematical reasoning. In Advances in Neural Information Processing Systems (NeurIPS), 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/e8f4eae0a41cab67fdead3aa6b77f083-Paper-Conference.pdf.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374. arXiv:2107.03374.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models, 2025b. URL https://arxiv.org/abs/2508.10751.

François Chollet. On the measure of intelligence, 2019. URL https://arxiv.org/abs/1911.01547. arXiv:1911.01547.

François Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier AI reasoning systems, 2025. URL https://arxiv.org/abs/2505.11831. arXiv:2505.11831.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Aviral Kumar, Rishabh Agarwal, Sridhar Thiagarajan, Craig Boutilier, and Aleksandra Faust. Inference-aware fine-tuning for best-of-N sampling in large language models. In International Conference on Learning Representations (ICLR), 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c40bed606c51c8e827c1ba75aa2da054-Paper-Conference.pdf.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168. arXiv:2110.14168.

Adithya M. Devraj and Jianshu Chen. Stochastic variance reduced primal dual algorithms for empirical composition optimization. In Advances in Neural Information Processing Systems (NeurIPS), 2019. URL https://arxiv.org/abs/1907.09150.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS), 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025. URL https://arxiv.org/abs/2510.04871. arXiv:2510.04871.

Hynek Kydlíček. Math-Verify: Math verification library. Software, 2025. URL https://github.com/huggingface/Math-Verify.

Xiaozhe Li, Yang Li, Xinyu Fang, Shengyuan Ding, Peiji Li, Yongkang Chen, Yichuan Ma, Tianyi Lyu, Linyang Li, Dahua Lin, Qipeng Guo, Qingwen Liu, and Kai Chen. Beyond mode collapse: Distribution matching for diverse reasoning. In International Conference on Machine Learning (ICML), 2026. URL https://arxiv.org/abs/2605.19461.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In International Conference on Learning Representations (ICLR), 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

MAA. 2023 AMC 12 A and B problems. Mathematical Association of America, American Mathematics Competitions, 2023. URL https://live.poshenloh.com/past-contests.

MAA. 2024 AIME I and II problems. Mathematical Association of America, American Mathematics Competitions, 2024. URL https://live.poshenloh.com/past-contests.

MAA. 2025 AIME I and II problems. Mathematical Association of America, American Mathematics Competitions, 2025. URL https://live.poshenloh.com/past-contests.

Mehryar Mohri, Jon Schneider, and Yifan Wu. Distributional alignment games for answer-level fine-tuning. In International Conference on Machine Learning (ICML), 2026. URL https://openreview.net/forum?id=orjnT28Vp6.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers, 2026. URL https://arxiv.org/abs/2606.18206. arXiv:2606.18206.

R. Tyrrell Rockafellar. Convex Analysis, volume 28 of Princeton Mathematical Series. Princeton University Press, Princeton, NJ, 1970. doi: 10.1515/9781400873173. URL https://doi.org/10.1515/9781400873173.

Amin Sghaier, Ali Parviz, and Alexia Jolicoeur-Martineau. Probabilistic tiny recursive model, 2026. URL https://arxiv.org/abs/2605.19943. arXiv:2605.19943.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Yuesong Shen, Nico Daheim, Bai Cong, Peter Nickl, Gian Maria Marconi, Bazan Clement Emile Marcel Raoul, Rio Yokota, Iryna Gurevych, Daniel Cremers, Mohammad Emtiyaz Khan, and Thomas Möllenhoff. Variational learning is effective for large deep networks. In International Conference on Machine Learning (ICML), 2024. URL https://proceedings.mlr.press/v235/shen24b.html.

Haoxiang Sun, Yingqian Min, Zhipeng Chen, Xin Zhao, and Ji-Rong Wen. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. In Annual Meeting of the Association for Computational Linguistics (ACL), 2026. URL https://aclanthology.org/2026.acl-long.792/.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning. In International Conference on Machine Learning (ICML), 2026. URL https://arxiv.org/abs/2602.02710.

Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Remi Munos. Optimizing language models for inference time objectives using reinforcement learning. In International Conference on Machine Learning (ICML), 2025. URL https://proceedings.mlr.press/v267/tang25o.html.

Christian Walder and Deep Karkhanis. Pass@K policy optimization: Solving harder reinforcement learning problems. In Advances in Neural Information Processing Systems (NeurIPS), 2025. URL https://arxiv.org/abs/2505.15201.

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

Mengdi Wang, Ethan X. Fang, and Han Liu. Stochastic compositional gradient descent: Algorithms for minimizing compositions of expected-value functions. Math. Program., 161(1–2):419–449, 2017. doi: 10.1007/s10107-016-1017-3.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In International Conference on Learning Representations (ICLR), 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yijun Zhang, Yule Xie, Jiaxin Ding, Xin Ding, Fan Xu, Haoxiang Zhang, and Luoyi Fu. Beyond the mean: Multi-moment policy optimization for LLM reasoning, 2026. URL https://arxiv.org/abs/2608.02149. arXiv:2608.02149.

A Derivations for finite-K posterior learning and Fenchel–Bregman updates

This appendix provides the derivations underlying the finite-K objective and the one-draw Fenchel–Bregman update. App. A.1 derives the binary all-fail probability and its direct K-draw Monte Carlo estimator. App. A.2 establishes the Fenchel representation and its optimal problem-specific weight, and App. A.3 shows that fixing this weight makes the posterior-dependent gradient estimable from one draw. Finally, App. A.4 derives the Bregman update for the dual weight and its closed-form EMA implementation.

A.1 Binary all-fail probability and direct Monte Carlo estimation

Fix a problem x and draw $\theta_{1:K} \overset{\text{iid}}{\sim} q_\phi$. Any solver randomness ξ_k is included in each draw, independently across candidates, and suppressed in the notation. Since $F(\theta, x) \in \{0, 1\}$,

$$\Pr(\text{all } K \text{ candidates fail} \mid x) = \mathbb{E}\left[\prod_{k=1}^{K} F(\theta_k, x)\right] = \bar{F}_\phi(x)^K. \tag{8}$$

The first equality uses the indicator property, and the second uses independence. Thus, the product is an unbiased Direct-K estimator using K posterior draws, and candidate coverage is $1 - \bar{F}_\phi(x)^K$. Averaging the all-fail probability over training problems gives the data term in Eq. 2.

For correlated draws, the product $\prod_{k=1}^{K} F(\theta_k, x)$ still equals one exactly when all K candidates fail, so it remains the indicator of the all-fail event. However, without independence its expectation does not generally factor into a product of marginal failure probabilities, and hence need not equal $\bar{F}_\phi(x)^K$.

Likewise, using the same posterior draw in all K factors does not recover the finite-K objective. Because $F(\theta, x) \in \{0, 1\}$,

$$F(\theta, x)^K = F(\theta, x),$$

so

$$\mathbb{E}_{\theta \sim q_\phi}[F(\theta, x)^K] = \bar{F}_\phi(x),$$

rather than $\bar{F}_\phi(x)^K$ in general. Thus, the K-th-power objective relies on independent candidate draws, and a naive reuse of one draw cannot provide the same Monte Carlo estimator.

A.2 Derivation of the Fenchel representation

This section proves the Fenchel representation in Eq. 4 and identifies its unique maximizer. For $K \geq 2$, let $\psi_K(\lambda) = (K-1)(\lambda/K)^{K/(K-1)}$ on $[0, K]$. Its derivative and inverse are

$$\psi_K'(\lambda) = \left(\frac{\lambda}{K}\right)^{1/(K-1)}, \qquad (\psi_K')^{-1}(u) = K u^{K-1}, \quad u \in [0, 1]. \tag{9}$$

The derivative is a strictly increasing bijection from $[0, K]$ to $[0, 1]$, so ψ_K is strictly convex. For $r \in [0, 1]$, the unique maximizer of the strictly concave function $\lambda r - \psi_K(\lambda)$ is $(\psi_K')^{-1}(r)$, including the endpoints $r = 0, 1$. Hence,

$$\begin{aligned} \lambda^\star(r) &= K r^{K-1}, \\ \lambda^\star(r) r - \psi_K(\lambda^\star(r)) &= K r^K - (K-1) r^K = r^K. \end{aligned} \tag{10}$$

This proves Eq. 4. Setting $r = \bar{F}_\phi(x)$ gives Eq. 5 with the KL term unchanged and recovers the weight in Eq. 3. For $K = 1$, the data term is linear and no Fenchel step is needed.

A.3 One-draw gradient with a fixed weight

Fix a problem x and suppose q_ϕ has a differentiable density whose support does not depend on ϕ. Assume differentiation can be exchanged with integration. The binary indicator $F(\theta, x)$ itself need not be differentiable. The score-function identity gives

$$\nabla_\phi \bar{F}_\phi(x) = \mathbb{E}_{\theta \sim q_\phi}[F(\theta, x) \nabla_\phi \log q_\phi(\theta)]\,. \tag{11}$$

Let λ_x be fixed with respect to ϕ and chosen before drawing θ. Multiplying the one-draw estimator in Eq. 11 by λ_x gives an unbiased estimate of $\lambda_x \nabla_\phi \bar{F}_\phi(x)$. When $\lambda_x = \lambda_x^\star$, this estimator is unbiased for the finite-K data gradient in Eq. 3. At other fixed weights, it remains unbiased for the fixed-weight data gradient, but not generally for the finite-K data gradient. The KL term is outside this Monte Carlo argument and remains unchanged in the fixed-weight posterior subproblem.

At update t, $\lambda_{x,t}$ is determined before $\theta_t \sim q_{\phi_t}$ is drawn, from the current running estimate. Conditioning on the information available before the draw therefore makes the weight fixed, so the same unbiasedness argument applies. The observed failure F_t then updates the running estimate for the next visit.

A.4 Bregman update for the dual weight

For $K \geq 2$, define the Bregman divergence generated by ψ_K as

$$D_{\psi_K}(\lambda, \mu) = \psi_K(\lambda) - \psi_K(\mu) - \psi_K'(\mu)(\lambda - \mu).$$

Given the current weight $\lambda_{x,t} \in [0, K]$, the observed failure $F_t \in [0, 1]$, and $\eta > 0$, we consider the Bregman-proximal update

$$\lambda_{x,t+1} = \arg\max_{\lambda \in [0,K]} \left\{ \lambda F_t - \psi_K(\lambda) - \eta^{-1} D_{\psi_K}(\lambda, \lambda_{x,t}) \right\}. \tag{12}$$

The first two terms are the per-step dual objective induced by the Fenchel representation, while the Bregman term regularizes the update toward the previous weight.

Proposition 1 (Closed-form Bregman update)**.** *Let $K \geq 2$, $\eta > 0$, and suppose*

$$\lambda_{x,t} = K \widehat{F}_{x,t}^{K-1}, \qquad \widehat{F}_{x,t} \in [0, 1].$$

With $\beta = 1/(1+\eta)$, the unique solution of Eq. 12 is

$$\widehat{F}_{x,t+1} = \beta \widehat{F}_{x,t} + (1-\beta) F_t, \qquad \lambda_{x,t+1} = K \widehat{F}_{x,t+1}^{K-1},$$

which gives the update in Eq. 6.

Proof. From App. A.2,

$$\psi_K'(\lambda) = \left(\frac{\lambda}{K}\right)^{1/(K-1)}, \qquad (\psi_K')^{-1}(u) = K u^{K-1}.$$

Differentiating the objective in Eq. 12 gives the stationarity condition

$$(1+\eta)\psi_K'(\lambda_{x,t+1}) = \psi_K'(\lambda_{x,t}) + \eta F_t.$$

Since $\lambda_{x,t} = K \widehat{F}_{x,t}^{K-1}$, we have $\psi_K'(\lambda_{x,t}) = \widehat{F}_{x,t}$. Therefore,

$$\begin{aligned} \psi_K'(\lambda_{x,t+1}) &= \frac{\widehat{F}_{x,t} + \eta F_t}{1+\eta} \\ &= \beta \widehat{F}_{x,t} + (1-\beta) F_t \coloneqq \widehat{F}_{x,t+1}. \end{aligned} \tag{13}$$

Since $\widehat{F}_{x,t}, F_t \in [0, 1]$, their convex combination $\widehat{F}_{x,t+1}$ also lies in $[0, 1]$. Applying $(\psi_K')^{-1}$ then gives

$$\lambda_{x,t+1} = K \widehat{F}_{x,t+1}^{K-1} \in [0, K].$$

Finally, the objective in Eq. 12 is strictly concave in λ, so this solution is the unique maximizer. □

Thus, the exponential running average in Eq. 6 is not an independently chosen heuristic; it is the closed-form solution of the Bregman-proximal weight update in Eq. 12.

B Experimental details

In this section, we describe the evaluation protocols, baseline controls, and posterior-training settings used in Sec. 4. We then specify the autoregressive evaluation and the training-cost measurements. We use the failure score f and running estimate $\widehat{f}$ defined in Sec. 3.3. Apps. B.3 and B.4 specify how the failure signal and its gradient estimate are instantiated in our experiments.

B.1 Recursive evaluation protocols

Metrics and candidate budgets. Selected accuracy measures exact solutions returned by a method's selector, whereas Pass@K measures exact-solution coverage among its first K ordered candidates. Here K denotes the evaluation budget, while K_{train} denotes the training objective's target budget. Posterior methods use independently sampled parameters, and stochastic solvers also resample their internal randomness. Unless stated otherwise, $K = 10$, and smaller budgets use ordered prefixes of the same candidate bank. For ARC-AGI, these prefixes follow the aggregation and ranking below. We randomly hold out 10% of the original training problems for validation and use it to select noise scales, the selection margin, posterior scales, and checkpoints.

Sudoku-Extreme. Sudoku-Extreme expands 1,000 training source puzzles with digit permutations, valid row and column permutations, and optional transposition, yielding 1,001,000 rows. The failure tracker is keyed by source-puzzle ID, so all variants of a source share one $\widehat{f}_x$ state. When a minibatch contains repeated sources, we average their observed failure scores before applying one EMA update per source. We evaluate candidates at reasoning depth 64 on all 422,786 test puzzles. TRM-based methods select with the largest q_{halt} score, while GRAM uses its native selector.

Maze-Hard. Maze-Hard contains 1,000 training mazes and 1,000 test mazes. TRM training uses the eight dihedral views, while FPRM adaptation uses unaugmented mazes. TRM-based evaluations use reasoning depth 16. PTRM uses the largest q_{halt} score, and W-PTRM applies the same rule after perturbing the Q head with the other trainable parameters. FBI retains the first posterior candidate unless the highest-scoring candidate exceeds its logit by more than $\ln 2$. FBI uses this margin rule with both TRM and FPRM backbones. We also tested this rule with the baselines, but retained their existing selectors because it did not improve their performance.

ARC-AGI. We follow the augmentation-based ARC evaluation framework of TRM and PTRM. ARC-AGI-1 and ARC-AGI-2 evaluation use the public ARC-AGI-1[2] and ARC-AGI-2[3] releases. For each augmentation, we evaluate 25 fixed posterior draws at reasoning depth 16 and map their predictions back to the original grid orientation (Sghaier et al., 2026). For ARC-AGI-1, we preserve its first two outputs from conservative Q- and structure-aware aggregation and rank the remainder by row-relative Q support within a one-logit margin. For ARC-AGI-2, we additionally use row-relative Q support and structural consistency for its first two outputs. We also tested FBI's aggregation and ranking rules with the baselines but these changes did not improve their performance. Following the TRM/PTRM ARC-AGI evaluation (Sghaier et al., 2026), we compute Pass@K over the top K predictions after aggregation and ranking, rather than over raw solver outputs.

B.2 Baseline details

Sampling-based reasoning. **PTRM** keeps the model parameters fixed, injects Gaussian noise into the latent state at each deep-recursion step, and selects among the resulting trajectories.

Weight-sampling controls. **W-PTRM** and **W-FPRM** are non-learned weight-sampling baselines. Let $\theta_0 \in \mathbb{R}^d$ collect every trainable floating-point parameter of the corresponding checkpoint, including its Q head but excluding buffers and masks. For candidate j, both controls sample the isotropic Gaussian

$$\theta^{(j)} = \theta_0 + \sigma\epsilon_j, \qquad \epsilon_j \overset{\text{iid}}{\sim} \mathcal{N}(0, I_d), \qquad \sigma = m\frac{1}{d}\sum_{r=1}^{d}|\theta_{0,r}|,$$

where m is a dimensionless multiplier. Each draw perturbs the original checkpoint independently, without latent noise. Table 4 gives the selected scales. Sudoku-Extreme and Maze-Hard use scales selected on the validation splits described above and freeze them before test evaluation. ARC-AGI-1 uses the fixed multiplier 0.30. ARC-AGI-2 selects its scale by Pass@2, then Pass@1, Pass@5, and pairwise Q-score accuracy.

[2] https://huggingface.co/datasets/arcprize/arc_agi_v1_public_eval
[3] https://huggingface.co/datasets/arcprize/arc_agi_v2_public_eval

Table 4: **Fixed-Gaussian weight-sampling controls.** The variance of every sampled scalar is σ^2.

Setting	m	σ
W-PTRM, Sudoku-Extreme	0.30	0.00312003
W-PTRM, Maze-Hard	0.30	0.00202273
W-PTRM, ARC-AGI-1	0.30	0.00650025
W-PTRM, ARC-AGI-2	0.20	0.00453306
W-FPRM, Maze-Hard	0.05	0.000298018

Additional comparison with GRAM. **GRAM** models reasoning as stochastic latent trajectories and samples several trajectories in parallel (Baek et al., 2026). Table 5 compares GRAM and FBI on Sudoku-Extreme and Maze-Hard.

Table 5: **Additional recursive reasoning comparison.** Acc. is selected accuracy, and Pass@10 is coverage among ten candidates. All values are percentages. FBI exceeds GRAM on both metrics and benchmarks.

	Sudoku-Extreme		Maze-Hard	
Method	Acc. ↑	Pass@10 ↑	Acc. ↑	Pass@10 ↑
GRAM	98.48	98.49	87.30	87.30
FBI (Ours)	**99.53**	**99.55**	**88.00**	**91.30**

B.3 Recursive posterior training

Posterior optimization methods. We use the IVON posterior update described in Sec. 3.3 (Shen et al., 2024). For learned posterior methods, we initialize the posterior mean from a task checkpoint and fine-tune its mean and diagonal precision on the task training split. Direct-10 uses the same failure score and IVON posterior machinery as FBI but estimates the $K_{\text{train}} = 10$ data term with ten independent posterior draws per update. It optimizes the transformed failure score rather than the raw task loss.

Failure-score construction. Given a solver's nonnegative per-problem training loss ℓ, FBI uses $f = 1 - \exp(-\ell/T)$ with a task-specific temperature $T > 0$. The loss ℓ is produced by the solver's training objective before auxiliary selector losses are added. Write $\bar{f}_\phi(x) \coloneqq \mathbb{E}_{\theta\sim q_\phi}[f(\theta, x)]$. Replacing the binary failure probability in Eq. 2 with this posterior mean gives the score-based objective

$$\min_\phi \ \mathbb{E}_x[\bar{f}_\phi(x)^{K_{\text{train}}}] + \tau_{\text{KL}} D_{\text{KL}}(q_\phi \| p_0). \tag{14}$$

This objective equals the exact all-fail target only when $\bar{f}_\phi(x)$ equals the one-candidate failure probability; otherwise it is a surrogate. For K_{train} independent posterior draws, $\prod_{k=1}^{K_{\text{train}}} f(\theta_k, x)$ is an unbiased Monte Carlo estimator of $\bar{f}_\phi(x)^{K_{\text{train}}}$. These scores supply the fixed-weight IVON loss in Eq. 7.

Practical posterior losses. Let B_s be the minibatch at global optimizer step s, and let g_i denote the problem or augmentation-group key for row i. With the batch-averaged Q auxiliary loss denoted by $\mathcal{L}_q$, the implemented recursive FBI scalar differentiated at step s is

$$\mathcal{L}_{\text{FBI},s} = \frac{1}{|B_s|} \sum_{i\in B_s} \lambda_{g_i,s} f(\theta_s, x_i) + \alpha_q \mathcal{L}_q(\theta_s; B_s). \tag{15}$$

Each $\lambda_{g_i,s} = K_{\text{train}} \widehat{f}_{g_i,s}^{K_{\text{train}}-1}$ is computed from the tracker before drawing θ_s and is held fixed and detached throughout that posterior update. If several rows in the minibatch share a group key, their failure scores are averaged before one tracker update for that group; tracker states for absent groups remain unchanged. The Fenchel penalty is omitted from this scalar because it is constant at fixed λ. Under IVON's parameterization, ESS and weight decay jointly specify the zero-mean isotropic

Gaussian prior p_0. Their product gives its precision. We vary the KL strength through ESS while adjusting weight decay inversely when the prior is to be held fixed. For the TRM runs, $\mathcal{L}_q$ averages prefix-level, class-balanced cell-level, and smooth-min aggregate binary cross-entropy terms. This auxiliary is added after constructing f, so it is neither part of the failure score nor multiplied by λ. It contributes to both gradient and curvature estimates.

TRM training configurations. Tables 6 and 7 report the task-specific settings. All four runs use curvature initialization 3.0, IVON momentum 0.9, curvature EMA 0.99999, and posterior noise scale 1.0. FBI initializes each failure estimate as $\widehat{f}_x = 0.5$. On Sudoku-Extreme and Maze-Hard, dense parameters use IVON, puzzle embeddings use separate learning rates, and model EMA is 0.999. ARC-AGI freezes the puzzle embeddings. We use the TRM-MLP variant for Sudoku-Extreme and the TRM-Att variant for Maze-Hard, ARC-AGI-1, and ARC-AGI-2.

Table 6: **Training configurations for TRM-based FBI.** The Sudoku-Extreme and Maze-Hard runs optimize $K_{\text{train}} = 10$ with one posterior draw per update. Both runs use constant learning rates and bfloat16 forward computation. Both use batch size 128, failure-score temperature 1.0, and puzzle-embedding weight decay 0.1.

Quantity	Sudoku-Extreme	Maze-Hard
Updates	200	188
Dense learning rate	1×10^{-4}	2.8565×10^{-5}
Puzzle-embedding learning rate	1×10^{-2}	1.3323×10^{-4}
IVON effective sample size	30,000	15,000
IVON weight decay	0.225	0.5
Relative KL strength	1	2
q_{halt} loss weight	0.35	0.27760
Failure-EMA coefficient	0.99	0.9

Table 7: **ARC-AGI posterior-training configurations.** Both runs use a diagonal IVON posterior over 6,829,058 dense parameters, frozen puzzle embeddings, and one parameter draw per update. Both use $K_{\text{train}} = 10$, failure-score temperature 1.0, dense learning rate 10^{-4}, and failure-EMA coefficient 0.99.

Quantity	ARC-AGI-1	ARC-AGI-2
Updates / batch size	400 / 128	400 / 128
q_{halt} loss weight	1.0	0.5
IVON effective sample size	30,000	30,000
IVON weight decay	0.225	0.225
Evaluation posterior scale	1.5	1.5

FPRM training configuration. FBI initializes its posterior mean from the EMA weights of the released FPRM checkpoint. We initialize IVON from scratch with zero momentum and unit curvature, without a separate model EMA. Table 8 lists the settings for 2,000 updates.

Table 8: **FPRM-based FBI training on Maze-Hard.** The run uses bfloat16 forward computation, $K_{\text{train}} = 10$, and one posterior draw per update.

Quantity	Value
Global batch size / GPUs	768 / 8
Dense learning rate	10^{-4}
IVON effective sample size	100,000
IVON momentum / curvature EMA	0.9 / 0.999
IVON noise scale	1.0
Failure-score temperature / EMA coefficient	0.0625 / 0.95760
Initial failure estimate / q_{halt} loss weight	0.5 / 0.5

B.4 AUTOREGRESSIVE EXPERIMENTS

Model and evaluation. We use Qwen3-4B-Base with MATH7.5K training data and evaluate on MATH500, OlymMATH, AMC23, AIME24, and AIME25. Each FBI candidate uses an independent posterior parameter draw followed by autoregressive sampling. The sampling temperature is 0.6 and top-p is 0.95 (Zhang et al., 2026).

Evaluation protocol and metrics. We generate sixteen candidates per problem, score every candidate with Math-Verify (Kydlíček, 2025), and record binary correctness. If c of a problem's sixteen candidates are correct, we estimate Pass@8 as $1 - \binom{16-c}{8} / \binom{16}{8}$. We average the problem-level estimates within each benchmark and report the unweighted mean across the five benchmark scores. All entries in Table 3 come from these common-protocol evaluations. Table 9 reports the complete candidate-coverage comparison at $K \in \{1, 4, 8, 16\}$, including the methods omitted from Fig. 2b for readability.

Table 9: **Mathematical-reasoning candidate coverage with Qwen3-4B-Base.** Each entry is the unweighted mean Pass@K across the five benchmarks, in percent. All methods use the same evaluation protocol and sixteen candidates per problem.

Method	Pass@1	Pass@4	Pass@8	Pass@16
Base	27.07	46.73	54.45	61.10
GRPO	41.89	52.23	56.82	61.41
Pass@K Training	40.17	53.67	59.55	<u>64.16</u>
DMPO	**43.01**	**55.03**	<u>59.60</u>	63.28
MaxRL	<u>42.43</u>	54.38	58.97	62.71
MMPO	38.79	51.01	56.56	61.94
FBI (Ours)	40.22	<u>54.52</u>	**60.44**	**66.06**

FBI training. The common posterior update is given in Sec. 3.3. We place a diagonal IVON posterior over all actor parameters, initialize its mean from Qwen3-4B-Base, and set its diagonal curvature to 10^{-3}. We train for 300 steps with learning rate 0.5, effective sample size 10^9, batch size 32, and seed 42 for the data order and IVON noise. Each update draws one parameter sample shared by its eight responses per prompt and all backward passes. We estimate policy gradients with REINFORCE Leave-One-Out (RLOO) (Ahmadian et al., 2024), which uses the other responses in each group to construct a leave-one-out reward baseline. The stored failure score starts at $8^{-1/7}$ and yields the normalized $K_{\text{train}} = 8$ weight $8\hat{f}^7$ before each update. After a successful update, an EMA with coefficient 0.5 incorporates the observed failure $1 - \frac{1}{8}\sum_{j=1}^{8} r_j$, where Math-Verify supplies binary rewards r_j.

FBI evaluation. Following MMPO (Zhang et al., 2026), we use the following user prompt:

```
{problem} Let's think step by step and output the final
answer within \boxed{}.
```

We use a 4,096-token response limit. We independently sample 16 parameter vectors from the learned posterior and use each vector to generate one response per problem. We fix the posterior-draw and decoding seed schedules for reproducibility.

B.5 TRAINING COMPUTE ACCOUNTING

Resource-benchmark setup. For resource measurement in Fig. 3c, FBI and Direct-10 use the same cellwise-Q TRM backbone, initial checkpoint, dataset, seed 0, batch size 128, bfloat16 computation, IVON schedule, and one NVIDIA RTX A6000 48 GB GPU, with compilation disabled. The benchmark changes only the finite-K_{train} gradient estimator and its number of posterior draws, one for FBI and ten for Direct-10, with $K_{\text{train}} = 10$ for both. The TRM experiments estimate IVON curvature with a Price's-theorem estimator. FBI reuses a scaled finite-K_{train} gradient for curvature on most updates and computes an ordinary full-batch curvature anchor every 20 updates. Direct-10

first evaluates a detached ten-draw probe and then replays the same draws to compute the product gradient. In the one-GPU implementation, its draws execute sequentially.

Timing, FLOPs, and memory. We run deterministic 10- and 30-update jobs with the same update schedule and divide the difference in GPU-synchronized training-loop elapsed times by 20, measuring updates 11–30 including one curvature anchor. Checkpoint writes and model, optimizer, and data-loader construction are outside the timed interval. PyTorch's FLOP counter profiles ordinary and curvature-anchor updates, combined in their 19:1 frequency. It counts matrix multiplication and fused attention in forward and backward passes, but not every elementwise operation. Peak allocated GPU memory is measured across the 30-update benchmark run, with the peak counter reset before model construction. Table 10 summarizes the resource measurements used in Fig. 3c.

Table 10: **Matched one-GPU training-resource benchmark on Maze-Hard.** Time and counted FLOPs are per update. FBI uses 6.8% of Direct-10's time and 6.6% of its counted FLOPs.

Quantity	FBI	Direct-10
Posterior draws per update	1	10
Training time/update (s)	1.68	24.80
Training FLOPs/update (TFLOPs)	54.7	828.9
Peak allocated GPU memory (GB)	30.9	34.3

Eight-GPU execution. Using the same global batch of 128, model, initial checkpoint, seed, bfloat16 precision, IVON schedule, and $K_{\mathrm{train}} = 10$, we additionally measure both methods on eight NVIDIA RTX A6000 GPUs. FBI partitions the batch into 16 examples per GPU and sums the rank-local gradients. Direct-10 replicates the full batch on each GPU and distributes its ten draws across them (two GPUs process two draws each), combining probe and replay results for the IVON update. Training time divides the GPU-synchronized elapsed time for updates 11–30 by 20, including batch retrieval, GPU transfer, communication, optimizer updates, and EMA. Training FLOPs/update sums the counted FLOPs across all eight GPUs for ordinary update 19 and curvature-anchor update 20, weighted 19:1 as in the one-GPU benchmark. Table 11 reports the resulting per-update comparison. These results characterize the measured batch-parallel and draw-parallel implementations, not the best possible eight-GPU schedule for either method.

Table 11: **Matched eight-GPU training-resource benchmark on Maze-Hard.** Time is synchronized elapsed time per update, whereas FLOPs are summed across the eight GPUs. Under these parallel implementations, FBI uses 4.8% of Direct-10's time and 6.6% of its aggregated FLOPs.

Quantity	FBI	Direct-10
Parallelization	Batch-parallel	Draw-parallel
Training time/update (s)	0.256	5.311
Training FLOPs/update (TFLOPs)	54.7	828.9

C Hyperparameter sensitivity

We vary the failure-EMA coefficient and relative KL strength used by FBI on Maze-Hard, holding relative KL strength at 2 for the EMA sweep and the failure-EMA coefficient at its reported value for the relative-KL sweep. We use the same random seed for all experiments in these sweeps. For the relative-KL sweep, we vary τ_{KL} through IVON's effective sample size while adjusting weight decay inversely. This keeps the product of ESS and weight decay, and hence the implied Gaussian prior, fixed.

Fig. 5 evaluates selected accuracy and Pass@10 under the same prefix-Q selector as the main Maze-Hard result. Across the EMA sweep at relative KL strength 2, selected accuracy ranges from 87.2% to 88.1% and Pass@10 from 90.2% to 91.7%. Both metrics rise from $\beta = 0$ to the reported $\beta = 0.9$ setting, then fall at $\beta = 0.99$ and 0.999. At $\beta = 0.9$, selected accuracy is 88.1% and Pass@10 is 91.7%. The $\beta = 0$ setting is a boundary control outside the finite-η Bregman-proximal setting

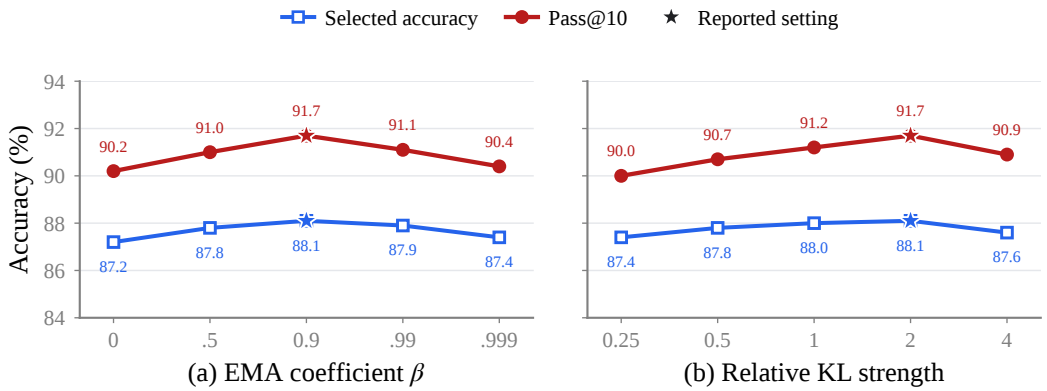


Figure 5: **EMA and KL sensitivity of FBI on Maze-Hard.** (a) varies the failure-EMA coefficient β at relative KL strength 2. (b) varies relative KL strength at $\beta = 0.9$ while keeping the implied prior fixed. Selected accuracy uses the frozen prefix-Q selector, and Pass@10 measures coverage among ten posterior candidates on test set with one fixed seed. Both metrics are highest at $\beta = 0.9$ in (a) and relative KL strength 2 in (b) among the tested settings.

analyzed above. In the relative-KL sweep, both metrics rise from 87.4%/90.0% at strength 0.25 to 88.1%/91.7% at strength 2, then fall to 87.6%/90.9% at strength 4.