LATENT THOUGHT FLOW: EFFICIENT LATENT REASONING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) increasingly rely on intermediate reasoning, yet explicit Chain-of-Thought (CoT) suffers from a linguistic space bottleneck: each thought must be decoded into tokens, causing high inference overhead. Latent reasoning moves deliberation into continuous space, but existing methods mostly learn deterministic or reward-maximizing paths, lacking a principled way to allocate probability across trajectories with different correctness and costs. We propose **Latent Thought Flow (LTF)**, which models reasoning as variable-length continuous trajectories and trains a sampler to match a reward-induced posterior over answer quality and computation cost. We instantiate this with a continuous GFlowNet using stochastic latent transitions. To handle sparse answer supervision, we introduce an Entropy-Weighted Subtrajectory Balance objective for intermediate rewards and a reference-prior regularizer to anchor exploration. Experiments under finetuning and transfer learning settings show that LTF outperforms explicit and latent reasoning baselines, improving accuracy by 9.5% while reducing reasoning length by 27.2% on average compared with strong latent reasoning methods.

1 INTRODUCTION

Large Language Models (LLMs) (Achiam et al., 2023; Singh et al., 2025; Grattafiori et al., 2024; Team et al., 2023) are increasingly tasked with complex objectives that demand a synergy of perception and deliberation, ranging from math problem solving (Lightman et al., 2024; Gao et al., 2023), embodied planning (Shinn et al., 2023; Hao et al., 2023), to code generation (Li et al., 2022; Nijkamp et al., 2022). A cornerstone of this success is the use of intermediate reasoning processes, where frameworks like Chain-of-Thought (CoT) (Wei et al., 2023; Kojima et al., 2022), Self-Consistency (Wang et al., 2022), and Tree-of-Thoughts (Yao et al., 2023) decompose intricate problems into sequential rationales. While promising, they inevitably encounter a *linguistic space bottleneck*: the requirement to decode every intermediate thought into discrete tokens. The explicit reasoning not only incurs substantial inference overhead but also constrains the model's internal computation into a potentially redundant textual form. Consequently, this reliance on explicit rationales creates an inherent accuracy-efficiency trade-off, limiting the scalability of reasoning capabilities in resource-constrained scenarios.

Latent reasoning has emerged as a promising alternative to mitigate the linguistic space bottleneck by performing multi-step computation directly within the model's continuous hidden-state space. Rather than fully externalizing deliberation as discrete text, these approaches keep intermediate reasoning internalized: Pause tokens (Goyal et al., 2023) allocate additional latent computation; stepwise internalization and compressed CoT (Deng et al., 2024; Cheng & Van Durme, 2024) densify textual rationales; and other frameworks (Hao et al., 2024; Shen et al., 2025; Xu et al., 2025a;b; Jiang et al., 2026; Yue et al., 2025; Zhao et al., 2026) like Coconut propagate continuous thought vectors or soft representations via the LLM backbone. These works demonstrate that meaningful intermediate computation need not always be expressed in linguistic words. However, most existing approaches primarily define a deterministic latent reasoning path and train it by compression, distillation, or reward maximization under a fixed reasoning budget. They do not specify how sampling probability mass should be allocated across latent reasoning trajectories with different correctness and computational costs under various questions.

We argue that this missing distributional view is central to efficient latent reasoning. For a given question, an LLM must navigate a manifold of plausible reasoning paths and only a sparse subset

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.

of which yields both logical consistency and computational efficiency. Without this distributional perspective, existing paradigms often fail to optimize the latent reasoning manifold: maximum-likelihood objectives inherit the verbosity of training rationales, while reinforcement learning tends to trigger posterior collapse onto a few high-reward modes. What is required is *a principled sampler capable of modeling the trade-offs between correctness and computational cost.* Such a framework should concentrate probability mass on concise and accurate trajectories while suppressing redundant or erroneous paths, thereby enabling the model to dynamically allocate its computational budget.

In this paper, we propose **Latent Thought Flow (LTF)**, a reward-proportional framework for learning continuous latent reasoning in LLMs. Given a question x, LTF samples a latent reasoning trajectory $\tau = (\mathbf{z}_{1:T}, \perp)$, where each $\mathbf{z}_t$ is a continuous thought state in LLM's latent space and $\perp$ denotes an adaptive stopping decision. The LLM then produces the final response conditioned on x and τ (Fig. 1). During training, the target answer defines a utility $\mathcal{R}_{x,y}(\tau)$ consisting of answer quality and computation cost, inducing the desired posterior over latent thoughts,

$$p^*(\tau \mid x, y) \propto \mathcal{R}_{x,y}(\tau). \tag{1}$$

We train the latent sampler to match this reward-induced distribution with a continuous GFlowNet objective. GFlowNets are designed to learn stochastic generative policies whose terminal samples are proportional to an unnormalized reward, which makes them well suited for preserving diverse high-utility solutions rather than only optimizing for a single maximum-reward trajectory (Bengio et al., 2021; 2023). While recent work has applied GFlowNets to LLM or VLM reasoning over discrete texts, proof, or action traces (Hu et al., 2023; Takase et al., 2024; Kang et al., 2025), LTF moves reward-proportional trajectory learning into LLM's latent space.

This formulation introduces three technical challenges. First, latent thoughts are high-dimensional continuous vectors, so discrete token-level balance equations are not directly applicable. Building on continuous GFlowNet theory (Lahlou et al., 2023), LTF parameterizes forward transitions with conditional densities and writes balance constraints over variable-length continuous trajectories. Second, answer-level supervision is sparse: the utility of a latent subtrajectory is typically observed only after the final response. We therefore use a Subtrajectory Balance objective (Malkin et al., 2022; Madan et al., 2023) with entropy-aware weighting to propagate terminal rewards to intermediate latent subtrajectories. Third, unconstrained exploration in a continuous latent space can drift away from meaningful latent states. LTF addresses this with a reference-prior regularizer that warm-starts exploration from modality-aligned anchors, while annealing the prior so that learned trajectories are shaped by the accuracy-efficiency reward. During inference, LLM samples latent thought paths and generates answers without access to gold rationales.

Experiments demonstrate that LTF improves the accuracy-efficiency frontier of reasoning and adapts latent budget to problem difficulty compared with explicit and latent reasoning baselines. Compared with latent reasoning baselines CoLaR and ReGuLaR, LTF improves accuracy by 12.9% while reducing reasoning length by 34.5% on average in fine-tuning tasks; under transfer learning, LTF improves accuracy by 6.0% while reducing reasoning length by 19.9% on average.

2 Related Work

Explicit Chain of Thought (CoT). A common approach to reasoning exposes intermediate computation as text. CoT prompting elicits rationales (Wei et al., 2023; Kojima et al., 2022), self-consistency aggregates sampled rationales (Wang et al., 2022), and Tree-of-Thoughts searches textual states (Yao et al., 2023). Multimodal variants produce explanations or grounded rationales, e.g., ScienceQA (Lu et al., 2022), Multimodal-CoT (Zhang et al., 2023), and Visual CoT (Shao et al., 2024a), while tool- and program-based systems externalize reasoning via vision experts or executable programs (Yang et al., 2023; Surís et al., 2023). However, these methods serialize visual, spatial, and perceptual computation into lengthy language, programs, or actions. LTF instead uses latent trajectories, reducing decoding overhead while retaining adaptive computation before final answer generation.

Latent Reasoning. To reduce the cost of long textual rationales, recent work explores reasoning in LLM latent spaces. Pause tokens add hidden computation before decoding (Goyal et al., 2023); explicit-to-implicit training gradually removes CoT supervision (Deng et al., 2024); and continuous-thought methods replace rationales with dense contemplation tokens, recurrent states, or soft thought tokens (Cheng & Van Durme, 2024; Hao et al., 2024; Shen et al., 2025; Xu et al., 2025a;b; Sun et al., 2026). Other methods learn abstract discrete tokens (Ramji et al., 2026) or probabilistic latent

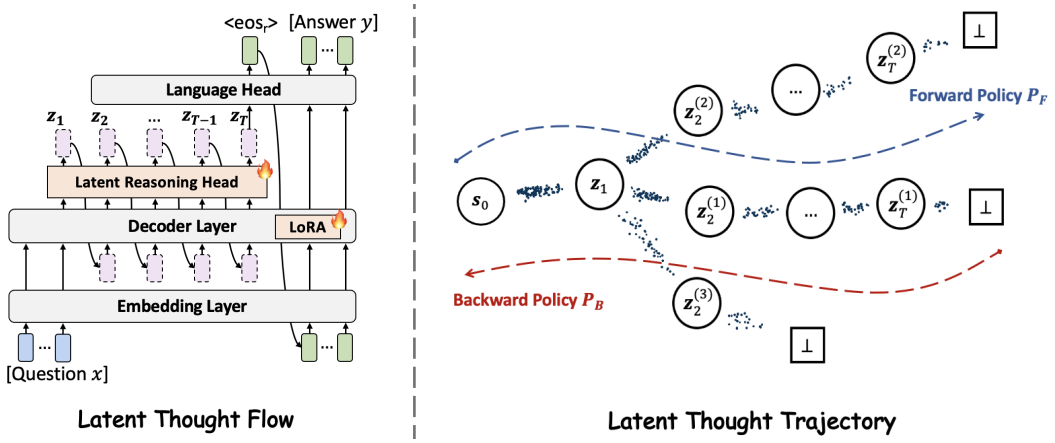

Figure 1: Illustration of our **LTF** framework. ***Left:*** Overall architecture of **LTF**. Only the LoRA module and latent reasoning head are trainable. ***Right:*** Overview of latent thought trajectories from a continuous GFlowNet perspective. A latent reasoning trajectory is represented as $\tau = (s_0, \mathbf{z}_1, \ldots, \mathbf{z}_T, \perp)$, where $\mathbf{z}_t$ denotes the latent thought state at step t. The forward policy P_F is parameterized as a variational sampling via LLM with a latent head (Eq. (3)) and trained with our entropy-weighted subtrajectory balance objective (Eq. (17)) to achieve the flow balance condition.

vectors with posterior inference (Kong et al., 2025; Jiang et al., 2026; Tan et al., 2025; Wang et al., 2026). Unlike methods that imitate or compress a single reasoning trace, LTF learns a reward-induced posterior over variable-length latent reasoning trajectories.

GFlowNets for Diverse Reasoning. Generative Flow Networks learn a policy whose terminal distribution is proportional to an unnormalized reward (Bengio et al., 2021; 2023), providing a distribution-matching alternative to reinforcement-learning-based latent reasoning that typically maximizes expected reward (Zhao et al., 2026; Du et al., 2025; Yue et al., 2025; Zhou et al., 2026). By preserving probability mass over multiple high-reward solutions, GFlowNets naturally support diverse reasoning. Trajectory Balance (Malkin et al., 2022) and continuous-space GFlowNets (Lahlou et al., 2023) further enable textual reasoning (Yu et al., 2024; Zhu et al., 2025) and visual rationale generation (Hu et al., 2023; Kang et al., 2025; Sun et al., 2025). However, existing GFlowNet-based methods mainly model explicit textual, symbolic, or visual traces, with limited focus on reasoning efficiency. In contrast, LTF samples in continuous latent superposition space, enabling adaptive computation and favoring trajectories that are both accurate and efficient.

3 Methodology

Overview. Latent Thought Flow (LTF) models reasoning as an adaptive flow over continuous latent thought states before generating the final answer (Fig. 1). Rather than learning a single deterministic latent chain or relying on explicit textual rationales, LTF learns a distribution over variable-length latent reasoning trajectories. The desired distribution should assign high probability to trajectories that produce correct answers with little latent computation, while retaining multiple valid reasoning paths when they exist. Learning such a distribution is challenging because 1) latent thoughts are unobserved, continuous, and variable-length; 2) supervision is sparse and available primarily through terminal answer quality; and 3) exploration in the high-dimensional latent space can be unstable. To address these challenges, we formulate latent reasoning as a continuous GFlowNet that learns a reward-proportional trajectory sampler (Bengio et al., 2021; 2023; Lahlou et al., 2023). Specifically, LTF combines an adaptive latent trajectory sampler (§3.1) with an accuracy-efficiency reward that defines the desired trajectory distribution (§3.2), an entropy-weighted continuous Subtrajectory Balance objective that propagates terminal supervision to latent subtrajectories (§3.3), and reference-prior regularization that semantically grounds exploration during early training (§3.4).

3.1 Variable-Length Latent Thought Trajectories

LTF replaces explicit textual rationales with a variable-length latent reasoning process, allowing the model to determine both *what* to compute internally and *when* to stop reasoning before generating

the final answer. Given a training set $\mathcal{D} = (x, y, r)$, where x denotes the input, y the target answer, and r an optional reference rationale used only during training, we want to train an LLM p_Θ with the latent reasoning mechanism. Let $\Theta = \{\phi, \psi\}$ denote the model parameters, where p_ϕ consists of the embedding and decoder layers that produce contextual latent states, and p_ψ denotes the language head that generates explicit tokens (Fig. 1).

Formally, given an input x, LTF samples a variable-length latent thought trajectory

$$\tau = (\mathbf{z}_{1:T}, \perp) \in \mathcal{T}, \qquad \mathbf{z}_t \in \mathbb{R}^{d_z}, \qquad 0 \le T \le T_{\max}, \tag{2}$$

where T denotes the number of continuous latent thoughts and $\perp$ denotes termination ($T = 0$ means answering without reasoning). Let $h_x = p_\phi(x)$ and $s_t = (h_x, \mathbf{z}_{1:t})$. To build a variational latent reasoning process with expressive latent superposition states, each latent thought $\mathbf{z}_t$ is sampled by a sampler q_φ consisting of a decoder layer and a latent reasoning head (Fig. 1).

At each step t, conditioned on the current context s_t, with $s_0 = h_x$, the LLM samples each latent reasoning state from its posterior distribution given the question and the previous ones, i.e., $z_{t+1} \sim q_\varphi(\cdot|s_t)$. We model the latent policy q_φ as the Gaussian distribution following prior work (Tan et al., 2025; Wang et al., 2026), and it samples the latent thoughts from a continuous density iteratively until achieving the decoded $<\text{eos}_\text{r}>$ token:

$$q_\varphi(\mathbf{z}_{t+1} \mid s_t) = \mathcal{N}\big(\boldsymbol{\mu}_\varphi(s_t), \boldsymbol{\sigma}^2_\varphi(s_t)\big), \qquad \pi^\perp(s_t) = p_\psi(< \text{eos}_\text{r} > |s_t), \tag{3}$$

with the reparameterization $\mathbf{z}_{t+1} = \boldsymbol{\mu}_\varphi(s_t) + \boldsymbol{\sigma}_\varphi(s_t) \odot \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, which allows gradients from the answer loss to pass through the latent computation (Kingma & Welling, 2013). The resulting trajectory density is

$$q_\varphi(\tau \mid x) = \left[\prod_{t=0}^{T-1} \big(1 - \pi^\perp(s_t)\big) q_\varphi(\mathbf{z}_{t+1} \mid s_t)\right] \pi^\perp(s_T), \qquad \tau = (\mathbf{z}_{1:T}, \perp). \tag{4}$$

where the product is empty when $T = 0$. We force $\pi^\perp(s_{T_{\max}}) = 1$ when the maximum budget $T_{\max}$ is reached. This formulation induces a distribution jointly over latent reasoning content and trajectory length, providing the basis for adaptive computation: the model can allocate fewer latent steps to easier inputs and additional steps when more internal computation is beneficial.

Finally, given the prefix and latent thought tokens, the answer can be generated:

$$p_\psi(y \mid x, \tau) = \prod_{m=1}^{|y|} p_\psi(y_m \mid \tau, y_{<m}). \tag{5}$$

LTF separates *latent reasoning* from *explicit answer generation*: the former is governed by a stochastic variable-length policy, while the latter decodes the final answer conditioned on the latent trajectory.

3.2 Reward-Proportional Target Distribution

The latent trajectory formulation specifies *how* LTF reasons, but not *which* trajectories should receive high probability. Since latent thoughts are unobserved and lack step-level supervision, we define their desired behavior through terminal answer quality and computational cost. For each training pair (x, y) and sampled trajectory $\tau = (\mathbf{z}_{1:T}, \perp)$, we assign the non-negative accuracy-efficiency reward

$$\mathcal{R}_{x,y}(\tau) = V_{x,y}(\tau) \exp\big(-\lambda_c C(\tau)\big), \tag{6}$$

where $V_{x,y}(\tau)$ measures answer quality, $C(\tau)$ measures latent computation, λ_c controls the cost penalty. We instantiate the quality score as

$$V_{x,y}(\tau) = \text{Ver}(y, \hat{y}_\tau) + \exp\left(1/|y| \cdot \log p_\psi(y \mid x, \tau)\right), \tag{7}$$

where $\hat{y}_\tau$ is the decoded answer and Ver denotes the task-specific accuracy. The verifier term aligns the reward with the evaluation metric, while the normalized likelihood term provides a dense signal. By default, we use a length-based cost function

$$C(\tau) = T, \tag{8}$$

which favors shorter latent trajectories unless additional computation improves answer quality.

Rather than optimizing only the single highest-reward trajectory, LTF seeks to preserve the multiplicity of valid reasoning paths by assigning probability in proportion to their rewards. Specifically, Eq. (6) induces the target distribution:

$$p^*(\tau \mid x, y) = \mathcal{R}_{x,y}(\tau)/Z_{\mathcal{R}}(x, y), \qquad Z_{\mathcal{R}}(x, y) = \int_{\mathcal{T}} \mathcal{R}_{x,y}(\tau) d\tau. \tag{9}$$

Consequently, multiple correct and efficient trajectories can retain substantial probability mass, whereas incorrect or unnecessarily long trajectories are suppressed. Because $Z_{\mathcal{R}}(x, y)$ integrates over continuous latent trajectories of all lengths, direct normalization is intractable. We therefore learn the sampler $q_\varphi(\tau \mid x)$ through a continuous GFlowNet objective that enforces local flow consistency without explicitly computing $Z_{\mathcal{R}}(x, y)$, as introduced in §3.3. The answer y defines rewards only during training; at inference, the sampler conditions solely on x.

3.3 Entropy-Weighted Continuous GFlowNet

Continuous Subtrajectory Balance. GFlowNets train a stochastic generator to sample complete trajectories in proportion to a positive reward, rather than only maximizing the expected reward of a single trajectory (Bengio et al., 2021; 2023). This property is well suited for latent reasoning, where multiple internal thought paths may lead to the same correct answer and should therefore be preserved during training. In LTF, the latent thoughts are continuous vectors, so the discrete transition probability in standard GFlowNets becomes a continuous transition density. To provide dense supervision for intermediate latent prefixes, we allow every prefix state to terminate: after stopping at a prefix, the model decodes an answer and receives a reward. This allows us to compute the flow of each prefix analytically from its immediate-stop reward and stop probability, avoiding an additional learned flow estimator.

For a subtrajectory $s_i \to s_{i+1} \to \cdots \to s_j$, the GFlowNet flow balance condition requires

$$F(s_i) \prod_{t=i}^{j-1} P_F(s_{t+1} \mid s_t) = F(s_j) \prod_{t=i}^{j-1} P_B(s_t \mid s_{t+1}), \tag{10}$$

where $F(s_i)$ is the flow at state s_i, and P_F and P_B denote the forward and backward transitions, respectively. Since our state $s_t = (h_x, \mathbf{z}_{1:t})$ stores the full latent prefix, each state has a unique parent obtained by removing the last latent vector. Thus, the backward transition is deterministic and contributes zero log-density along a valid trajectory.

Under our latent thought trajectory formulation in §3.1, the forward transition density from s_t to non-terminal state s_{t+1} is given by

$$P_F(s_{t+1} \mid s_t) = \big(1 - \pi^{\perp}(s_t)\big) q_\varphi(\mathbf{z}_{t+1} \mid s_t), \tag{11}$$

where $q_\varphi(\mathbf{z}_{t+1} \mid s_t)$ is the Gaussian density defined in Eq. (3), and $\pi^{\perp}(s_t)$ is the probability of stopping at s_t. Therefore, the forward edge log-density for a non-terminal transition is

$$\ell_t^\varphi = \log q_\varphi(\mathbf{z}_{t+1} \mid s_t) + \log\big(1 - \pi^{\perp}(s_t)\big). \tag{12}$$

Since the reward $\mathcal{R}_{x,y}(s_t \to \perp)$ of terminating at prefix s_t is defined in Eq. (6), where the model stops at s_t, decodes an answer, and scores it against the training answer y, the flow $F(s_t)$ can be expressed by the terminal consistency condition

$$F(s_t)\pi^{\perp}(s_t) = \mathcal{R}_{x,y}(s_t \to \perp) \Rightarrow F(s_t) = \frac{\mathcal{R}_{x,y}(s_t \to \perp)}{\pi^{\perp}(s_t)}. \tag{13}$$

Hence, each intermediate state flow is obtained analytically from its immediate-stop reward and stopping probability. This converts sparse terminal supervision into prefix-level flow constraints while avoiding a separately parameterized flow function.

We use Sub-Trajectory Balance (Madan et al., 2023) objective to enforce this consistency over all subtrajectories due to its stable optimization and low variance property (Silva et al., 2025; Bengio et al., 2023). For $0 \le i < j \le T$, the residual is

$$\begin{aligned} \chi_{i:j} &= \log F(s_i) + \sum_{t=i}^{j-1} \ell_t^\varphi - \log F(s_j) \\ &= \log \frac{\mathcal{R}_{x,y}(s_i \to \perp) \left[\prod_{t=i}^{j-1} q_\varphi(\mathbf{z}_{t+1} \mid s_t)\big(1 - \pi^{\perp}(s_t)\big)\right] \pi^{\perp}(s_j)}{\mathcal{R}_{x,y}(s_j \to \perp)\pi^{\perp}(s_i)}. \end{aligned} \tag{14}$$

Thus, the continuous Sub-Trajectory Balance objective is

$$\mathcal{L}_{\mathrm{SubTB}}(x, y) = \mathbb{E}_{\tau \sim q_\varphi(\cdot|s_0)} \left[\sum\nolimits_{0 \le i < j \le T} \chi_{i:j}^2 \right]. \tag{15}$$

Minimizing this loss encourages the latent sampler to assign larger density to reasoning trajectories whose prefixes and completions obtain higher answer rewards, while preserving multiple valid reasoning paths instead of collapsing to a single mode.

Entropy-Weighted Subtrajectory Balance. Standard SubTB assigns equal optimization weight to all sampled subtrajectories, irrespective of the uncertainty of their latent transitions. This can be inefficient in continuous latent reasoning: when the transition policy has high conditional entropy, probability mass is spread over a broader region of the latent space, making the corresponding local flow constraint more difficult to learn reliably from sparse terminal feedback. We therefore introduce *Entropy-Weighted Subtrajectory Balance* (EW-SubTB), which allocates greater optimization weight to relatively uncertain subtrajectories while leaving the underlying balance condition unchanged.

Given S trajectories sampled for the same input x, let $s \in \{1, \dots, S\}$ index a sampled trajectory. We define the entropy for the latent $\mathbf{z}_{t+1}^{(s)}$ as $h_t^{(s)} = \mathcal{H}[q_\varphi(\mathbf{z}_{t+1}^{(s)} \mid s_t^{(s)})]$ where $\mathcal{H}(\cdot)$ denotes differential entropy. For a subtrajectory $s_i \to \cdots \to s_j$, the length-normalized entropy is given by $\bar{h}_{i:j}^{(s)} = \frac{1}{j-i} \sum_{t=i}^{j-1} h_t^{(s)}$. Thus, the entropy-aware weight is

$$\omega_{i:j}^{(s)} = \mathrm{sg}\left[\exp(\bar{h}_{i:j}^{(s)}) / \big(1/|\mathcal{S}_{i:j}| \cdot \sum\nolimits_{r \in \mathcal{S}_{i:j}} \exp(\bar{h}_{i:j}^{(r)})\big) \right], \tag{16}$$

where $\mathcal{S}_{i:j}$ denotes the set of subtrajectories between indices i and j in S sampled trajectories, and $\mathrm{sg}[\cdot]$ denotes the gradient stopping operation. This weighting keeps the average scale of the loss stable while assigning larger weights to relatively high-entropy latent reasoning subtrajectories.

Using the sample-specific residual $\chi_{i:j}^{(s)}$ defined by Eq. (14), the empirical EW-SubTB objective is

$$\mathcal{L}_{\mathrm{flow}}(x, y) = \frac{1}{S} \sum\nolimits_{k=1}^{S} \sum\nolimits_{0 \le i < j \le T^{(k)}} \omega_{i:j}^{(k)} \left(\chi_{i:j}^{(k)} \right)^2. \tag{17}$$

Since $\omega_{i:j}^{(s)}$ only reweights the squared residual, EW-SubTB changes the optimization emphasis but preserves the same reward-proportional target distribution. Our proposed objective is validated via both empirical results in §4 and analysis in Appendix C.5.

3.4 Reference-Prior Regularization

EW-SubTB determines how training signal is allocated across sampled trajectories, but it does not by itself prevent the sampler from exploring semantically uninformative regions early in training. This issue is particularly pronounced in a high-dimensional continuous latent space, where the reward signal is initially sparse and sampled latent states may have little connection to meaningful reasoning. When reference rationales are available, we therefore introduce a reference prior that provides semantic guidance for early latent-space exploration (Zhang et al., 2023; Shao et al., 2024a). The prior anchors latent thoughts to meaningful regions of latent space, and the GFlowNet reward later reshapes the sampler toward efficient and correct trajectories.

Given a reference latent rationale r for input x, we introduce a reference branch after the decoder layer similar to the latent reasoning head. We define the reference transition density $p_{\theta'}^{\mathrm{ref}}$ as:

$$p_{\theta'}^{\mathrm{ref}}(\mathbf{z}_t \mid s_{t-1}) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_{\theta'}(s_{t-1}), \boldsymbol{\sigma}_{\theta'}^2(s_{t-1})\big). \tag{18}$$

The reference branch shares the underlying representation space with the latent reasoning model and is trained from teacher-generated rationales, thereby encouraging the shared trainable representations to remain anchored to semantically meaningful reasoning regions. Its training objective is

$$\mathcal{L}_{\mathrm{prior}} = -\mathbb{E}_{(x,y,r)\sim\mathcal{D}} \left[\log p_{\theta'}^{\mathrm{ref}}(r \mid x) \right]. \tag{19}$$

Thus, early training benefits from semantically grounded latent states with stable exploration, while later training is governed by the accuracy-efficiency reward. If no rationale is available, this term is omitted and LTF trains only from answer supervision and the reward.

3.5 Training Objective and Inference

The training objective consists of complementary components for latent thought sampling and answer prediction. The flow loss trains the sampler to allocate probability mass according to reward, while the cross entropy loss ensures that sampled thoughts remain effective for deriving the correct answer. Given an input-answer pair (x, y), we define the answer loss as

$$\mathcal{L}_{\text{ans}}(x, y) = -\mathbb{E}_{\tau \sim q_\varphi(\cdot|h_x)} \left[\log p_\psi(y \mid \tau)\right]. \tag{20}$$

When computing the flow loss, we stop gradients through the scalar reward $\mathcal{R}_{x,y}(\tau)$; this prevents the decoder from changing the reward scale to satisfy balance and leaves answer learning to Eq. (20). The overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{flow}} + \lambda_{\text{ans}}\mathcal{L}_{\text{ans}} + \lambda_{\text{prior}}\mathcal{L}_{\text{prior}}, \tag{21}$$

where λ_{ans} and λ_{prior} are weighting hyperparameters. During inference, given the input x, LLM samples latent thoughts via Eq. (3) and decodes the answer. By default, LTF uses a single trajectory for efficiency. With larger test-time budgets, it can adaptively sample multiple latent trajectories via tree of latent thoughts (Yao et al., 2023) without decoding long explicit reasoning chains.

4 Experiments

Tasks, Metrics & Baselines. We evaluate LTF across multiple LLM backbones, including LLaMA-3.2 1B/3B (Grattafiori et al., 2024), LLaMA-3.1 8B (Grattafiori et al., 2024), and DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025). For arithmetic and symbolic reasoning, we train and evaluate on GSM8K-Aug (Deng et al., 2023), ASDiv-Aug (Xu et al., 2025a), and DU (Srivastava et al., 2023), covering mathematical reasoning, word-problem solving, and data understanding. For tasks requiring longer answers and more structured reasoning, we train and evaluate on PlanBench-Blocksworld (Valmeekam et al., 2023) and MBPP (Austin et al., 2021), which involve multi-step planning and code generation, respectively. To test arithmetic generalization and difficulty scalability, we further evaluate on out-of-domain math datasets, including GSM-Hard (Gao et al., 2023), SVAMP (Patel et al., 2021), MultiArith (Roy & Roth, 2015), AQUA-RAT (Ling et al., 2017), and MATH (Hendrycks et al., 2021). Following prior work, we report three primary metrics: task performance, measured by Accuracy (Acc.), success rate, or pass@1 depending on the task; Reasoning Length (# L), defined as the number of reasoning steps used per question; and Latency (Lat.), measuring the average inference time (ms) per query. We compare LTF with explicit reasoning methods, such as Zero-Shot CoT (Wei et al., 2023) and Assist-CoT (Xu et al., 2025a), as well as latent reasoning methods, such as Coconut (Hao et al., 2024), SoftCoT++ (Xu et al., 2025b), and ReGuLaR (Wang et al., 2026). All baselines follow their default configurations for fair comparison.

We train LTF with LoRA using rank $r = 128$ and scaling factor $\alpha = 32$, and AdamW (Loshchilov & Hutter, 2017) with learning rate 1×10^{-4}, weight decay 1×10^{-2}, global batch size 256, and 100 training epochs on RTX PRO 6000 GPUs. For fairness, all methods use the same training budget and stop once either the epoch limit or time budget is reached. See more details in Appendix B.

4.1 Main Results

Finetuning Performance on Arithmetic and Symbolic Reasoning Tasks. Table 1 shows that LTF achieves superior accuracy-efficiency tradeoffs across tasks and backbones. Compared with ReGuLaR, LTF with LLaMA-1B improves average accuracy by 7.4% relatively, while reducing reasoning length by 6.4% and latency by 5.2%. On the more challenging GSM8K-Aug, it achieves a 7.3% relative accuracy gain with 9.5% shorter reasoning. This trend remains consistent for LLaMA-8B, where LTF improves average accuracy by 3.6% while reducing reasoning length by 8.9%. These results highlight that LTF learns efficient and effective latent thought trajectories, mitigating the decoding overhead associated with long reasoning chains and improving the task performance.

Finetuning Performance on Long Structured Reasoning Tasks. Table 2 shows that LTF outperforms latent reasoning methods on two long structured generation tasks: executable multi-step planning and functional program generation. Compared with ReGuLaR, LTF with LLaMA-1B improves PB success rate by 9.4% and MBPP Pass@1 by 7.2% relatively, while reducing end-to-end latency by 11.5% and 11.9%, respectively. The improvements remain consistent at the larger scale: with LLaMA-8B, LTF improves PB success rate and MBPP Pass@1 by 6.7% and 4.8%, respectively,

Table 1: Performance comparison under finetuning across different LLM backbones. We report the averaged Accuracy (Acc. %), Reasoning Length (# L), and end-to-end latency (Lat. ms/query).

Model	Method	GSM8K-Aug		ASDiv-Aug		DU		Average		
		Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Lat.(↓)
LLaMA-3.2 Instruct 1B	Zero-Shot CoT	15.70	120.37	40.85	120.73	33.18	120.01	29.91	120.37	1644.03
	Assist-CoT	16.30	119.85	42.18	119.53	35.34	119.58	31.27	119.65	1683.78
	SoftCoT++	17.04	119.73	45.72	119.48	38.18	119.70	33.65	119.64	1772.63
	CODI	13.30	6.00	43.93	6.00	38.06	6.00	31.76	6.00	274.18
	Coconut	20.50	6.00	53.18	6.00	43.97	6.00	39.22	6.00	273.89
	CoLaR	26.60	5.63	67.19	5.19	50.12	4.17	47.97	5.00	253.50
	ReGuLaR	34.58	3.69	72.19	1.24	59.92	1.20	55.56	2.04	124.56
	LTF	**37.09**	**3.34**	**75.11**	**1.22**	**66.83**	**1.18**	**59.68**	**1.91**	**118.12**
LLaMA-3.2 Instruct 3B	Zero-Shot CoT	17.10	121.39	51.08	121.67	48.90	121.75	39.03	121.60	2163.27
	Assist-CoT	17.90	120.98	52.42	120.11	50.28	120.04	40.20	120.38	2297.10
	SoftCoT++	18.59	120.77	56.39	119.97	53.31	119.85	42.76	120.20	2409.62
	Coconut	32.71	6.00	66.76	6.00	57.23	6.00	52.23	6.00	446.19
	CoLaR	42.58	5.48	73.05	5.17	61.29	4.15	58.97	4.93	375.73
	ReGuLaR	45.59	3.89	79.46	1.24	70.31	1.18	65.12	2.10	189.38
	LTF	**48.30**	**3.42**	**84.04**	**1.21**	**76.82**	**1.17**	**68.85**	**1.95**	**180.30**
LLaMA-3.1 Instruct 8B	Zero-Shot CoT	19.17	121.30	89.23	121.93	65.67	121.04	58.02	121.42	2793.88
	Assist-CoT	19.93	121.04	89.48	121.51	65.82	120.39	58.41	120.98	2904.88
	SoftCoT++	20.86	120.76	90.09	120.52	68.89	120.15	59.95	120.48	3018.54
	Coconut	36.10	6.00	91.09	6.00	73.74	6.00	66.98	6.00	772.45
	CoLaR	45.19	5.62	91.12	5.51	76.59	4.18	70.97	5.10	628.23
	ReGuLaR	50.14	3.93	92.00	1.28	81.06	**1.19**	74.40	2.13	306.83
	LTF	**53.14**	**3.37**	**94.02**	**1.24**	**84.09**	1.20	**77.08**	**1.94**	**296.77**
DeepSeek-R1 Distill-Qwen 1.5B	Zero-Shot CoT	15.94	121.57	45.70	120.70	37.02	120.09	32.89	120.79	1834.74
	Assist-CoT	17.36	120.81	46.48	119.81	38.62	119.67	34.15	120.10	1909.34
	SoftCoT++	18.19	120.64	48.51	119.93	43.04	119.83	36.58	120.13	2057.24
	Coconut	24.91	6.00	56.85	6.00	46.46	6.00	42.74	6.00	316.32
	CoLaR	27.38	5.71	69.36	5.23	50.35	4.17	49.03	5.04	287.57
	ReGuLaR	34.69	3.54	75.59	1.21	63.91	1.19	58.06	1.98	137.27
	LTF	**36.94**	**3.27**	**79.75**	**1.20**	**69.80**	**1.17**	**62.16**	**1.88**	**132.58**

Table 2: Performance comparison on structured reasoning tasks across different LLM backbones. We report task performance, Reasoning Length (# L), and end-to-end latency (Lat. ms/query).

Model	Method	PB			MBPP		
		Success(↑)	# L(↓)	Lat.(↓)	Pass@1(↑)	# L(↓)	Lat.(↓)
LLaMA-1B	Zero-Shot CoT	12.96	308.51	3561.49	38.28	281.47	2161.82
	Coconut	35.61	**6.00**	1120.14	44.32	6.00	923.59
	CoLaR	44.37	13.54	1074.87	53.70	12.19	902.42
	ReGuLaR	52.26	7.83	812.72	59.05	6.63	649.18
	LTF	**57.15**	7.41	**719.10**	**63.32**	**5.91**	**571.99**
LLaMA-8B	Zero-Shot CoT	18.68	293.29	5796.53	62.39	233.16	4064.46
	Coconut	44.96	**6.00**	1549.06	67.22	6.00	1314.18
	CoLaR	56.91	13.60	1404.77	71.16	12.25	1267.63
	ReGuLaR	62.09	7.89	1091.95	74.07	6.72	823.39
	LTF	**66.23**	7.74	**993.07**	**77.64**	**5.99**	**749.58**

while reducing latency by 9.1% and 9.0%. The latency gain reflects more efficient reasoning and demonstrates the model's ability to produce concise yet effective answers. These improvements indicate that distributional latent reasoning is particularly effective for long-horizon tasks, where it can mitigate early error propagation along extended explicit reasoning chains.

Transfer Learning Performance on Arithmetic Reasoning Tasks. Table 3 shows that LTF improves transfer performance across unseen arithmetic reasoning tasks. Compared with ReGuLaR, LTF improves average accuracy by 3.9% and 3.6% relatively on LLaMA-1B and LLaMA-8B, while further reducing average reasoning length by 2.8% and 3.6%, respectively. These results validate the reasoning generalization ability of LTF, improving broader reasoning performance while preserving strong compression efficiency. Additional experimental results are provided in Appendix C.1.

4.2 Ablation Studies

We conduct ablation studies on the key components of LTF. Due to the space limitation, we defer the detailed ablation analyses and test-time scaling to Appendix C.2 and C.4.

Table 3: Performance comparison under transfer learning setting across different LLM backbones. We report the averaged Accuracy (Acc. %), Reasoning Length (# L), and end-to-end latency (Lat. ms/query). Full results on four backbones are in Appendix C.1.

Model	Method	GSM-Hard		SVAMP		MultiArith		Average		
		Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Lat.(↓)
LLaMA-3.2 Instruct 1B	Zero-Shot CoT	2.74	120.46	27.17	120.80	28.29	120.14	19.40	120.47	1647.43
	SoftCoT++	3.77	123.56	30.96	123.42	32.13	120.66	22.29	122.55	1789.28
	CODI	2.94	6.00	21.65	6.00	19.20	6.00	14.60	6.00	274.78
	Coconut	4.87	6.00	38.90	6.00	41.30	6.00	28.36	6.00	274.51
	CoLaR	6.22	7.03	47.22	2.94	86.93	3.22	46.79	4.40	235.94
	ReGuLaR	8.25	4.10	48.19	2.10	88.97	2.29	48.47	2.83	149.26
	LTF	**8.71**	**3.97**	**52.01**	**2.11**	**90.37**	**2.17**	**50.36**	**2.75**	**146.23**
LLaMA-3.1 Instruct 8B	Zero-Shot CoT	4.51	121.66	37.43	121.56	38.82	121.84	26.92	121.69	2807.28
	SoftCoT++	5.91	124.82	42.85	123.82	43.19	121.98	30.65	123.54	3033.27
	Coconut	8.26	6.00	57.18	6.00	62.41	6.00	42.62	6.00	773.51
	CoLaR	10.97	6.53	61.30	2.83	93.80	3.14	55.36	4.17	589.40
	ReGuLaR	12.53	3.92	68.54	**2.07**	95.28	2.26	58.78	2.75	347.56
	LTF	**13.19**	**3.70**	**72.29**	2.09	**97.18**	**2.16**	**60.89**	**2.65**	**320.77**

Table 4: Ablation study of entropy weighting (EW) and reference-prior regularization (RPR) using LLaMA-3.2 1B backbone. We report the averaged Accuracy (Acc. %) and Reasoning Length (# L).

Method	GSM8K-Aug		ASDiv-Aug		DU		Average	
	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)
LTF	37.09	3.34	75.11	1.22	66.83	1.18	59.68	1.91
w/o EW	36.64	3.31	74.80	1.20	64.74	1.19	58.73	1.90
w/o RPR	37.02	3.72	75.04	2.38	65.70	2.14	59.25	2.75

Table 5: Ablation on latent thought exploration objective using the LLaMA-3.2 1B backbone. We report the averaged Accuracy (Acc. %) and Reasoning Length (# L).

Model	GSM8K-Aug		ASDiv-Aug		DU		Average	
	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)
Latent-GRPO	25.93	13.17	66.93	12.45	49.62	11.14	47.49	12.25
DB	35.17	7.73	72.28	7.19	60.48	6.92	55.98	7.28
TB	35.84	8.01	73.05	7.39	61.52	7.14	56.80	7.51
LTF	37.09	3.34	75.11	1.22	66.83	1.18	59.68	1.91

Analysis on Optimization Modules. Table 4 shows that EW and RPR provide complementary benefits. **1)** EW improves the average accuracy by 1.6%, while leaving the reasoning length nearly unchanged. It shows that EW improves the quality of latent reasoning. **2)** RPR improves reasoning efficiency: compared with LTF w/o RPR, it reduces the average reasoning length by 30.5%.

Analysis of Latent Thought Exploration Objectives. Table 5 compares exploration objectives in LTF, with baseline details in Appendix B. **1)** Compared with Latent-GRPO, GFlowNet-based objectives achieve better accuracy-efficiency tradeoffs: DB raises average accuracy from 47.49% to 55.98% while reducing reasoning length from 12.25 to 7.28. **2)** LTF achieves the best tradeoff, with the highest accuracy (59.68%) and shortest reasoning length (1.91).

5 Conclusion

We introduce LTF, a framework for efficient latent reasoning in LLMs. LTF models reasoning as variable-length continuous latent trajectories and trains the sampler with a continuous GFlowNet objective, allocating probability mass according to answer quality and computational cost. An entropy-weighted subtrajectory balance objective and a reference-prior regularizer further improve exploration and stability. Extensive experiments highlight the superiority of LTF over explicit and latent reasoning methods, showing a path towards a better reasoning accuracy-efficiency frontier.

Limitations. Our experiments primarily focus on textual tasks involving words and symbols; extending to other modalities, such as vision and speech, remains as future work. In addition, we will further explore the generalization ability of Latent Thought Flow in theory.

AI USE STATEMENT

In this work, generative AI tools were used solely to improve the readability of the manuscript, including grammar, spelling, and word choice. They were not used to generate research ideas or synthetic data; develop theoretical models or conceptual frameworks; formulate mathematical claims or assist with proofs; propose or refine hypotheses; design or provide feedback on research methodology or experiments; implement methods; translate content; clean or reformat data; support qualitative or thematic data analysis; or interpret experimental results. All research ideas, methods, analyses, and experimental results presented in this work are original contributions of the authors. All AI-assisted edits were manually reviewed by the authors to ensure that they preserved the intended meaning and technical accuracy of the manuscript.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. 7, 15

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394, 2021. 2, 3, 5, 15

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023. 2, 3, 5

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024. 1, 2

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023. 7

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024. 1, 2

Hanwen Du, Yuxin Dong, and Xia Ning. Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts. *arXiv preprint arXiv:2509.26314*, 2025. 3

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023. 1, 7

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023. 1, 2

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 1, 7

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 7

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, 2023. 1

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024. 1, 2, 7, 14

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. *arXiv preprint arXiv:2103.03874*, 2021. 7

Edward J Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio, and Nikolay Malkin. Amortizing intractable inference in large language models. *arXiv preprint arXiv:2310.04363*, 2023. 2, 3

Cong Jiang, Xiaofeng Zhang, Fangzhi Zhu, XiaoWei Chen, Junxiong Zhu, and Zheng Zhang. Rethinking llm reasoning: From explicit trajectories to latent representations. In *The Fourteenth International Conference on Learning Representations*, 2026. 1, 3

Haoqiang Kang, Enna Sachdeva, Piyush Gupta, Sangjae Bae, and Kwonjoon Lee. Gflowvlm: Enhancing multi-step reasoning in vision-language models with generative flow networks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 3815–3825, 2025. 2, 3

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 4

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022. 1, 2

Deqian Kong, Minglu Zhao, Dehong Xu, Bo Pang, Shu Wang, Edouardo Honig, Zhangzhang Si, Chuan Li, Jianwen Xie, Sirui Xie, et al. Latent thought models with variational bayes inference-time computation. *arXiv preprint arXiv:2502.01567*, 2025. 3

Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-Garcıa, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. In *International Conference on Machine Learning*, pp. 18269–18300. PMLR, 2023. 2, 3

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022. 1

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*, volume 2024, pp. 39578–39601, 2024. 1

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 158–167, 2017. 7

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 7

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521, 2022. 2

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pp. 23467–23483. PMLR, 2023. 2, 5

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35: 5955–5967, 2022. 2, 3, 15

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022. 1

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021. 7

Keshav Ramji, Tahira Naseem, and Ramón Fernandez Astudillo. Learning efficient latent reasoning with abstract chain-of-thought. In *Workshop on Latent {\&} Implicit Thinking {\textendash} Going Beyond CoT Reasoning*, 2026. 2

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1743–1752, 2015. 7

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 37:8612–8642, 2024a. 2, 6

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024b. 15

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025. 1, 2, 14

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023. 1

Tiago Silva, Rodrigo Barreto Alves, Eliezer de Souza da Silva, Amauri H Souza, Vikas Garg, Samuel Kaski, and Diego Mesquita. When do gflownets learn the right distribution? In *The Thirteenth International Conference on Learning Representations*, 2025. 5

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025. 1

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023. 7

Guohao Sun, Hang Hua, Jian Wang, Jiebo Luo, Sohail Dianat, Majid Rabbani, Raghuveer Rao, and Zhiqiang Tao. Latent chain-of-thought for visual reasoning. *arXiv preprint arXiv:2510.23925*, 2025. 3

Lihao Sun, Hang Dong, Bo Qiao, Qingwei Lin, Dongmei Zhang, and Saravan Rajmohan. Llm reasoning as trajectories: Step-specific representation geometry and correctness signals. *arXiv preprint arXiv:2604.05655*, 2026. 2

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11888–11898, 2023. 2

Ryoichi Takase, Masaya Tsunokake, Yuta Tsuchiya, and Shota Inuzuka. Gflownet fine-tuning for diverse correct solutions in mathematical reasoning tasks. *arXiv preprint arXiv:2410.20147*, 2024. 2

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *arXiv preprint arXiv:2505.16552*, 2025. 3, 4, 14

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987, 2023. 7, 15

Fanmeng Wang, Haotian Liu, Guojiang Zhao, Hongteng Xu, and Zhifeng Gao. Regular: Variational latent reasoning guided by rendered chain-of-thought. *arXiv preprint arXiv:2601.23184*, 2026. 3, 4, 7, 14, 16

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. 1, 2

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, 2023. URL https://arxiv.org/abs/2201.11903. 1, 2, 7, 14

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, 2025a. 1, 2, 7, 14

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *arXiv preprint arXiv:2505.11484*, 2025b. 1, 2, 7, 14

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023. 2

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023. 1, 2, 7

Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. Flow of reasoning: Training llms for divergent reasoning with minimal examples. *arXiv preprint arXiv:2406.05673*, 2024. 3

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. Hybrid latent reasoning via reinforcement learning. *arXiv preprint arXiv:2505.18454*, 2025. 1, 3

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023. 2, 6

Yiran Zhao, Yuhui Xu, Doyen Sahoo, Caiming Xiong, and Junnan Li. Learning to reason over continuous tokens with reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. 1, 3

Yuyan Zhou, Jiarui Yu, Hande Dong, Zhezheng Hao, Hong Wang, Jianqing Zhang, and Qiang Lin. Lepo: Latent reasoning policy optimization for large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14416–14427, 2026. 3

Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Hengli Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, et al. Flowrl: Matching reward distributions for llm reasoning. *arXiv preprint arXiv:2509.15207*, 2025. 3

A Notation Definitions

The major notations used in this paper are listed in Table 6.

Table 6: Major Notations.

Notation	Definition
x	Input question.
y	Target answer.
h_x	Embedding of the input x produced by the backbone encoder, i.e., $h_x = p_\phi(x)$.
$\mathbf{z}_t$	Continuous latent thought sampled at step t.
s_t	Context state $(h_x, z_{1:t})$.
T	Number of latent thoughts in τ
$\perp$	Stop token indicating termination of the latent trajectory.
$\tau = (\mathbf{z}_{1:T}, \perp)$	Variable-length latent thought trajectory ending with stop token $\perp$.
$q_\varphi(\mathbf{z}_{t+1} \mid s_t)$	Continuous latent thought transition density.
$p_\psi(< \text{eos}_\text{r} >\mid s_t)$	Stopping probability of latent reasoning at prefix state s_t.
$\pi^\perp(s_t)$	Shorthand for $p_\psi(< \text{eos}_\text{r} >\mid s_t)$.
$F(s_t)$	Flow assigned to state s_t.
$\mathcal{R}_{x,y}(s_t \to \perp)$	Terminal reward for stopping from state s_t given the input x and target y.
$\chi_{i:j}$	Continuous SubTB residual over subtrajectories $s_i \to \cdots \to s_j$.

B Additional Implementation Details

In this section, we provide comprehensive details regarding baselines and training details for LLMs.

Baselines. For the main experiments, we employ various explicit and latent-based reasoning methods as baselines on multiple LLM backbones, including LLaMA-3.2 1B/3B, LLaMA-3.1 8B, and DeepSeek-R1-Distill-Qwen-1.5B. The detailed description of the LLM baselines is provided below:

Zero-Shot CoT (Wei et al., 2023): This baseline uses chain-of-thought prompting templates to elicit explicit intermediate reasoning steps before producing the final answer. It gives a contextual explicit-reasoning reference. To prevent answer degeneration, the maximum response token budget $T_{\max}$ for arithmetic and symbolic reasoning tasks is set as 256. The maximum response token budget for long structured reasoning tasks is set as 1024. We ablate the response token budget in Appendix C.3.

Assist-CoT (Xu et al., 2025a): This baseline uses a smaller assistant model to generate explicit reasoning tokens, which are then provided to the LLM as chain-of-thought prompts. The reasoning length ($\#L$) includes both soft thought tokens and intermediate reasoning steps.

SoftCoT++ (Xu et al., 2025b): This method generates multiple diverse soft thought representations through specialized initial tokens, and further encourages diversity among them with a contrastive learning objective. These soft thoughts are then used to guide the LLM's explicit reasoning process. The reasoning length ($\#L$) includes both soft thought tokens and intermediate reasoning steps.

CODI (Shen et al., 2025): This baseline uses self-distillation to align the hidden activations of latent thoughts with CoT trajectories, thereby transferring explicit reasoning patterns into the latent space.

Coconut (Hao et al., 2024): This baseline recursively feeds the last hidden state of the LLM back as a latent thought embedding, using it as the next-step input to support subsequent reasoning.

CoLaR (Tan et al., 2025): This baseline dynamically compresses reasoning chains into latent variables, where each latent state summarizes multiple consecutive reasoning tokens. It trains a latent head to autoregressively predict compressed embeddings and uses reinforcement learning to encourage diverse and concise latent reasoning paths.

ReGuLaR (Wang et al., 2026): This baseline formulates latent reasoning within a variational framework, where latent reasoning states are sampled from posterior distributions conditioned on previous states. It renders explicit CoT segments as images and uses their visual representations to regularize the latent states.

For ablation studies, we employ RL- and GFlowNet-based methods as baselines. The detailed description of the above baselines is provided below:

Latent-GRPO (Shao et al., 2024b): This baseline samples a group of outputs $\{o_1, o_2, \ldots, o_G\}$ from the old policy $p_{\Theta_{\text{old}}}$, where G denotes the group size. Each output o_i consists of a latent reasoning trajectory τ_i and a final answer. Latent-GRPO then optimizes the current policy p_Θ by minimizing the following clipped objective:

$$\mathcal{L}_{\text{Latent-GRPO}} = -\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{p_\Theta(o_i \mid x)}{p_{\Theta_{\text{old}}}(o_i \mid x)} A_i, \text{clip}\left(\frac{p_\Theta(o_i \mid x)}{p_{\Theta_{\text{old}}}(o_i \mid x)}, 1-\epsilon, 1+\epsilon\right) A_i\right). \quad (22)$$

Here, A_i is computed as a group-normalized reward:

$$A_i = \frac{r_i - \text{mean}(r_1, r_2, \ldots, r_G)}{\text{std}(r_1, r_2, \ldots, r_G)}. \quad (23)$$

We use $\mathcal{R}_{x,y}(\tau_i)$ as the reward, i.e.,

$$r_i = \mathcal{R}_{x,y}(\tau_i). \quad (24)$$

We replace the LTF objective Eq. (21) with $\mathcal{L}_{\text{Latent-GRPO}}$ and keep other training configurations the same.

Detailed Balance (DB) (Bengio et al., 2021): This variant uses the same overall objective as LTF (Eq. (21)), but replaces $\mathcal{L}_{\text{flow}}$ with a Detailed Balance loss. It can be viewed as a one-step special case of Sub-Trajectory Balance, where flow consistency is enforced only between adjacent states. Specifically,

$$\mathcal{L}_{\text{DB}} = \mathbb{E}_{\tau\sim q_\varphi(\cdot|s_0)}\left[\sum_{t=0}^{T-1} \chi^2_{t:t+1}\right], \quad (25)$$

where $\chi_{t:t+1}$ is defined in Eq. (14). All other training configurations are kept unchanged.

Trajectory Balance (TB) (Malkin et al., 2022): This variant uses the same overall objective as LTF (Eq. (21)), but replaces $\mathcal{L}_{\text{flow}}$ with a Trajectory Balance loss. Unlike DB, which enforces local one-step consistency, TB enforces flow consistency only over the complete sampled trajectory. Specifically,

$$\mathcal{L}_{\text{TB}} = \mathbb{E}_{\tau\sim q_\varphi(\cdot|s_0)}\left[\chi^2_{0:T}\right], \quad (26)$$

where $\chi_{0:T}$ is given by Eq. (14). All other training configurations are kept unchanged.

Metrics. Following prior work, we report three primary metrics: task performance, measured by Accuracy (Acc.), success rate, or pass@1 depending on the task; Reasoning Length (# L), defined as the number of reasoning steps used per question; and Latency (Lat.), measuring the average inference time (ms) per query. All results are averaged over five independent runs with different random seeds. The reported metric reasoning length ($\#L$) measures the number of reasoning steps before producing the final answer: for explicit reasoning methods, this corresponds to the number of explicit reasoning tokens; for latent reasoning approaches, it corresponds to the number of latent reasoning steps; and for implicit one-pass methods such as iCoT, $\#L$ is defined as 0.

Implementations. For all experiments, the Latent Head in LTF is implemented as a three-layer MLP, with hidden dimensions matching the hidden size of the LLM backbone. We set `rollout_n` $= 20$ and $T_{\max} = 32$ during training, meaning that the model samples 20 reasoning trajectories with maximum latent reasoning length of 32 for each prompt. For finetuning settings, the task-specific verifier $V_{x,y}(\cdot)$ is instantiated according to the corresponding correctness criterion based on tasks. For arithmetic reasoning tasks such as GSM8K-Aug and ASDiv-Aug, each problem has a determined ground-truth answer, allowing correctness to be verified by comparing the generated answer with the reference answer. In contrast, planning and code generation tasks may admit multiple valid solutions. For PB, we therefore follow the standard PlanBench protocol (Valmeekam et al., 2023) and verify each generated plan through sequential execution: a plan is considered correct only if every action is applicable and the final state satisfies the goal. For MBPP, we follow the standard execution-based evaluation protocol (Austin et al., 2021), where a generated program is considered correct if it executes successfully and passes all provided unit tests. For transfer learning settings, we use the models finetuned on GSM8K-Aug to study the out-of-domain generalization performance. In the ablation study on latent-thought exploration objectives, we use the same setting, i.e., `rollout_n` $= 20$

Table 7: Performance comparison under finetuning setting (GSM8K-Aug) and transfer learning setting (GSM-Hard, SVAMP, MultiArith) using LLaMA-3.2-1B-Instruct. We report averaged values with 95% confidence interval on Accuracy (Acc. %) and Reasoning Length (# L).

Method	GSM8K-Aug		GSM-Hard		SVAMP		MultiArith	
	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)
Zero-Shot CoT	$15.70_{\pm 0.18}$	$120.37_{\pm 0.17}$	$2.74_{\pm 0.12}$	$120.46_{\pm 0.23}$	$27.17_{\pm 0.23}$	$120.80_{\pm 0.18}$	$28.29_{\pm 0.31}$	$120.14_{\pm 0.19}$
Assist-CoT	$16.30_{\pm 0.19}$	$119.85_{\pm 0.19}$	$3.02_{\pm 0.15}$	$125.19_{\pm 0.24}$	$28.04_{\pm 0.29}$	$123.02_{\pm 0.18}$	$30.06_{\pm 0.34}$	$121.86_{\pm 0.20}$
SoftCoT++	$17.04_{\pm 0.21}$	$119.73_{\pm 0.20}$	$3.77_{\pm 0.16}$	$123.56_{\pm 0.26}$	$30.96_{\pm 0.31}$	$123.42_{\pm 0.25}$	$32.13_{\pm 0.36}$	$120.66_{\pm 0.23}$
CODI	$13.30_{\pm 0.64}$	$6.00_{\pm 0.00}$	$2.94_{\pm 0.24}$	$6.00_{\pm 0.00}$	$21.65_{\pm 0.70}$	$6.00_{\pm 0.00}$	$19.20_{\pm 0.82}$	$6.00_{\pm 0.00}$
Coconut	$20.50_{\pm 0.74}$	$6.00_{\pm 0.00}$	$4.87_{\pm 0.31}$	$6.00_{\pm 0.00}$	$38.90_{\pm 0.71}$	$6.00_{\pm 0.00}$	$41.30_{\pm 0.67}$	$6.00_{\pm 0.00}$
CoLaR	$26.60_{\pm 0.17}$	$5.63_{\pm 0.05}$	$6.22_{\pm 0.14}$	$7.03_{\pm 0.06}$	$47.22_{\pm 0.30}$	$2.94_{\pm 0.03}$	$86.93_{\pm 0.21}$	$3.22_{\pm 0.05}$
ReGuLaR	$34.58_{\pm 0.22}$	$3.69_{\pm 0.21}$	$8.25_{\pm 0.14}$	$4.10_{\pm 0.43}$	$48.19_{\pm 0.39}$	$2.10_{\pm 0.19}$	$88.97_{\pm 0.27}$	$2.29_{\pm 0.28}$
LTF	$\mathbf{37.09}_{\pm 0.19}$	$\mathbf{3.34}_{\pm 0.09}$	$\mathbf{8.71}_{\pm 0.13}$	$\mathbf{3.97}_{\pm 0.21}$	$\mathbf{52.01}_{\pm 0.27}$	$\mathbf{2.11}_{\pm 0.06}$	$\mathbf{90.37}_{\pm 0.19}$	$\mathbf{2.17}_{\pm 0.08}$

and $T_{\max} = 32$, for all compared methods, including Latent-GRPO, DB, and TB, to ensure a fair comparison. Following prior work (Wang et al., 2026), we construct the reference representation r as the embedding of golden explicit rationale in the training dataset. All hyperparameters in LTF were selected by grid search on the validation set and then fixed across all experiments. We set $\lambda_c = 0.03$ to control the cost penalty in the terminal reward. In validation experiments, LTF was stable for $\lambda_c \in (0.01, 0.04)$, and we therefore use the value 0.03 as the default setting. We set $\lambda_{\text{ans}} = 1.0$ for the answer-generation loss, so that the answer objective remains on the same scale as the main training objective. For the prior loss, we use a piecewise-linear annealing schedule for λ_{prior}. Specifically, λ_{prior} is initialized to 3.0 and linearly decayed to 0.1 over 100 epochs, with the value updated every five epochs. This schedule imposes a strong prior constraint at the beginning of training and gradually relaxes the constraint as training proceeds, allowing the reward-driven objective to have a larger influence in later stages. We found that LTF is generally robust to small variations around these settings.

C Extended Experimental Results on Large Language Models

C.1 Finetuning and Transfer Learning Results

Results with 95% Confidence Interval. Table 7 shows finetuning and transfer learning performance comparison on reasoning datasets under LLaMA-3.2-1B-Instruct with 95% confidence interval ($\pm$) on Accuracy (Acc. %) and Reasoning Length (# L). These results indicate that LTF achieves better accuracy-efficiency tradeoffs across tasks. This observation can be attributed to the Entropy-Weighted SubTB objective, which encourages compact and informative latent states rather than redundant reasoning trajectories, highlighting the stability and broad applicability of LTF across model families and scales.

Transfer Learning Performance. Table 8 shows transfer learning performance comparison on arithmetic reasoning datasets under four backbones on Accuracy (Acc. %) and Reasoning Length (# L). These results validate the reasoning generalization ability of LTF, improving broader reasoning performance while preserving strong compression efficiency.

C.2 Extreme Compression and Ablation Analysis

Extreme Compression Evaluation. Although accuracy decreases under extreme compression, Table 9 shows that LTF outperforms baselines across all settings on four LLM backbones, verifying its advantage in preserving semantic information. Compared with ReGuLaR, LTF improves the average accuracy by 2.72% on MATH and 3.61% on AQUA-RAT, demonstrating its superior ability to preserve task-relevant semantic information under compression.

Analysis on Training Rollout Sample Size S. As shown in Table 10, increasing S improves performance by expanding trajectory diversity: with entropy weighting, average accuracy rises from 57.41% to 59.68% as S increases from 5 to 20, showing the scaling effect during training.

Analysis on Entropy Weighting (EW). Uniform SubTB assigns equal importance to all sampled subtrajectories, whereas EW emphasizes uncertain regions in which the sampler requires stronger

Table 8: Performance comparison under transfer learning setting across different LLM backbones. We report the averaged Accuracy (Acc. %) and Reasoning Length (# L).

Model	Method	GSM-Hard		SVAMP		MultiArith		Average	
		Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)
LLaMA-3.2 Instruct 1B	Zero-Shot CoT	2.74	120.46	27.17	120.80	28.29	120.14	19.40	120.47
	Assist-CoT	3.02	125.19	28.04	123.02	30.06	121.86	20.37	123.36
	SoftCoT++	3.77	123.56	30.96	123.42	32.13	120.66	22.29	122.55
	CODI	2.94	6.00	21.65	6.00	19.20	6.00	14.60	6.00
	Coconut	4.87	6.00	38.90	6.00	41.30	6.00	28.36	6.00
	CoLaR	6.22	7.03	47.22	2.94	86.93	3.22	46.79	4.40
	ReGuLaR	8.25	4.10	48.19	2.10	88.97	2.29	48.47	2.83
	LTF	**8.71**	**3.97**	**52.01**	**2.11**	**90.37**	**2.17**	**50.36**	**2.75**
LLaMA-3.2 Instruct 3B	Zero-Shot CoT	3.84	121.68	34.01	121.52	35.16	121.54	24.34	121.58
	Assist-CoT	4.16	125.03	35.06	123.42	37.03	121.98	25.42	123.48
	SoftCoT++	5.21	124.72	38.89	122.96	40.26	121.74	28.12	123.14
	Coconut	7.34	6.00	52.47	6.00	58.79	6.00	39.53	6.00
	CoLaR	9.73	6.46	60.70	2.72	90.10	3.04	53.51	4.07
	ReGuLaR	11.20	3.73	62.32	2.23	91.27	2.17	54.93	2.71
	LTF	**11.88**	**3.64**	**65.48**	**2.14**	**92.94**	**2.10**	**56.77**	**2.63**
LLaMA-3.1 Instruct 8B	Zero-Shot CoT	4.51	121.66	37.43	121.56	38.82	121.84	26.92	121.69
	Assist-CoT	4.75	125.59	38.62	124.62	40.60	122.64	27.99	124.28
	SoftCoT++	5.91	124.82	42.85	123.82	43.19	121.98	30.65	123.54
	Coconut	8.26	6.00	57.18	6.00	62.41	6.00	42.62	6.00
	CoLaR	10.97	6.53	61.30	2.83	93.80	3.14	55.36	4.17
	ReGuLaR	12.53	3.92	68.54	**2.07**	95.28	2.26	58.78	2.75
	LTF	**13.19**	**3.70**	**72.29**	2.09	**97.18**	**2.16**	**60.89**	**2.65**
DeepSeek-R1 Distill-Qwen 1.5B	Zero-Shot CoT	2.81	120.78	27.23	121.48	28.33	121.53	19.46	121.26
	Assist-CoT	3.07	126.03	28.12	123.76	30.14	122.70	20.44	124.16
	SoftCoT++	3.86	123.72	31.02	123.26	32.18	121.92	22.35	122.97
	Coconut	4.96	6.00	38.98	6.00	41.31	6.00	28.41	6.00
	CoLaR	6.32	7.03	47.27	2.94	87.01	3.22	46.86	4.40
	ReGuLaR	8.33	4.10	48.20	2.10	89.04	2.29	48.52	2.83
	LTF	**8.83**	**3.94**	**52.76**	**2.08**	**91.10**	**2.21**	**50.90**	**2.74**

Table 9: Extreme compression performance of LTF under finetuning setting. We report the averaged Accuracy (Acc. %) and Reasoning Length (# L).

Dataset	Method	LLaMA-1B		LLaMA-3B		LLaMA-8B		DS-1.5B		Average	
		Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)
MATH	Coconut	5.18	6.00	7.72	6.00	8.68	6.00	6.67	6.00	7.06	6.00
	CoLaR	5.31	58.29	8.35	60.44	9.13	67.19	11.13	62.20	8.48	62.03
	ReGuLaR	6.62	1.00	11.80	1.00	13.90	1.00	15.60	1.00	11.98	1.00
	LTF	**8.10**	**1.00**	**16.18**	**1.00**	**17.31**	**1.00**	**17.20**	**1.00**	**14.70**	**1.00**
AQUA-RAT	Coconut	23.17	6.00	26.31	6.00	30.94	6.00	27.81	6.00	27.06	6.00
	CoLaR	23.69	18.60	32.78	28.20	34.52	20.30	34.33	27.00	31.33	23.53
	ReGuLaR	37.28	1.00	39.12	1.00	42.41	1.00	39.07	1.00	39.47	1.00
	LTF	**39.43**	**1.00**	**43.18**	**1.00**	**46.62**	**1.00**	**43.07**	**1.00**	**43.08**	**1.00**

Table 10: Ablation on training rollout sample size S in LTF using the LLaMA-3.2 Instruct 1B backbone. We report the averaged Accuracy (Acc. %) and Reasoning Length (# L).

S	GSM8K-Aug		ASDiv-Aug		DU		Average	
	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)
5	35.19	3.33	73.42	1.23	63.61	1.19	57.41	1.92
10	36.71	3.33	74.65	1.21	65.84	1.18	59.07	1.91
20	37.09	3.34	75.11	1.22	66.83	1.18	59.68	1.91

supervision, which is analyzed from entropy-based perspective (Appendix C.5). In addition, its benefit is accordingly more pronounced under larger training-time rollout budgets.

Analysis on Reference-Prior Regularization (RPR). As shown in Table 11, RPR produces shorter trajectories. At 25%, 50%, 75%, and 100% of training, it reduces #L by 48.9%, 60.0%, 31.6%, and 10.2%, respectively. These results indicate that RPR anchors the sampling distribution to a reliable

Table 11: Performance of LTF and LTF w/o RPR at different training stages under GSM8K-Aug on LLaMA-1B. We report Accuracy (Acc. %) and Reasoning Length (# L).

Training Stage	Method	Acc.(↑)	# L(↓)
25%	LTF	11.31	**10.29**
	w/o RPR	**13.44**	20.14
50%	LTF	**23.48**	**5.12**
	w/o RPR	22.35	12.81
75%	LTF	**31.82**	**4.17**
	w/o RPR	30.23	6.10
100%	LTF	**37.09**	**3.34**
	w/o RPR	37.02	3.72

Table 12: Ablation on the maximum response token budget $T_{\max}$ using the LLaMA-3.2-Instruct-1B backbone under zero-shot CoT. We report Accuracy (Acc. %) and Reasoning Length (#L).

GSM8K-Aug			PB			MBPP		
$T_{\max}$	Acc.(↑)	#L(↓)	$T_{\max}$	Acc.(↑)	#L(↓)	$T_{\max}$	Acc.(↑)	#L(↓)
64	3.27	62.05	512	10.14	306.93	512	38.06	282.90
128	14.33	120.41	1024	12.96	308.51	1024	38.28	281.47
256	15.69	120.39	2048	12.99	308.41	2048	38.30	281.30

semantic region during early training while producing shorter latent reasoning trajectories, leading to lower overall training overhead.

C.3 Analysis on Zero-Shot CoT

We investigate whether the performance of zero-shot CoT is constrained by the maximum response-token budget. As shown in Table 12, increasing $T_{\max}$ from 64 to 128 substantially improves accuracy on GSM8K-Aug from 3.27% to 14.33%, indicating that a budget of 64 tokens is insufficient for completing the reasoning process. By contrast, further increasing $T_{\max}$ from 128 to 256 yields only a marginal accuracy improvement, from 14.33% to 15.69%, while leaving the average reasoning length essentially unchanged (120.41 versus 120.39 tokens). The stability of the realized reasoning length despite the doubled generation budget suggests that the model naturally terminates after approximately 120 reasoning tokens on average rather than exhausting the available budget.

To further distinguish natural termination from budget-induced truncation, we examine generation-completion statistics. We define a response as reaching the generation limit if it exhausts the full `max_new_tokens` budget without emitting an EOS token. A valid final answer is an output that can be parsed by the task-specific evaluator, irrespective of whether the parsed answer is correct. With $T_{\max} = 128$, only 1.2% of responses reach the generation limit, while 98.8% produce a valid final answer. With $T_{\max} = 256$, no response reaches the generation limit, and every response produces a valid final answer. Taken together, the nearly identical accuracies and realized reasoning lengths under $T_{\max} = 128$ and $T_{\max} = 256$, along with the absence of truncated responses under the latter setting, demonstrate that zero-shot CoT is not constrained by the maximum response-token budget.

A similar pattern is observed on the longer structured reasoning tasks. Increasing $T_{\max}$ from 512 to 1024 improves performance, whereas a further increase to 2048 produces negligible changes in either accuracy or realized reasoning length. Moreover, at $T_{\max} = 1024$, no response on either PB or MBPP reaches the generation limit, and every response produces an output that can be processed by the corresponding task evaluator. These results confirm that $T_{\max} = 1024$ is sufficient for natural completion on both tasks. Accordingly, we use $T_{\max} = 256$ for arithmetic and symbolic reasoning tasks and $T_{\max} = 1024$ for long structured reasoning tasks.

Table 13: Ablation on the number of latent thought chains N during test time using the LLaMA-3.2 Instruct 1B backbone. We report the averaged Accuracy (Acc. %) and Reasoning Length (# L).

N	GSM8K-Aug		ASDiv-Aug		DU		Average	
	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)	Acc.(↑)	# L(↓)
1	37.09	3.34	75.11	1.22	66.83	1.18	59.68	1.91
5	38.13	3.38	77.62	1.24	68.03	1.20	61.26	1.94
10	38.72	3.37	78.73	1.23	68.94	1.20	62.13	1.93

C.4 Test-Time Scaling Analysis

Table 13 evaluates how the number of latent thought chains N affects inference performance. Increasing N improves accuracy across all three benchmarks, while leaving the average reasoning length nearly unchanged. Notably, raising N from 1 to 10 improves average accuracy from 59.68% to 62.13% ($+2.45$), whereas the average reasoning length changes only marginally from 1.91 to 1.93. These results demonstrate a key advantage of LTF: it supports effective test-time scaling by sampling multiple compact latent reasoning trajectories, rather than relying on substantially longer reasoning traces. We further observe that LTF can adaptively allocate different numbers of reasoning steps to problems of varying difficulty.

This trend aligns with the intended Monte Carlo inference mechanism of LTF. By sampling more latent thought chains, the model increases the probability of identifying a high-reward reasoning trajectory. At the same time, the diminishing improvement from $N = 5$ to $N = 10$ suggests that a modest number of samples is sufficient to capture most of the attainable benefit. Overall, these results provide empirical evidence that the learned reward-induced sampling posterior distribution in LTF is effective for improving inference-time performance efficiently.

C.5 Analysis of Latent Reasoning Dynamics

We provide additional analyses to better understand the latent reasoning dynamics learned by LTF. We focus on two complementary aspects: (i) the entropy of the learned latent reasoning trajectories, and (ii) the distributional behavior of the sampler under multiple rollouts and different task distributions.

Entropy of Latent Reasoning Trajectories. To better understand the latent reasoning dynamics learned by LTF, we analyze the average reasoning entropy of latent reasoning paths generated by CoLaR, ReGuLaR, LTF w/o entropy weighting (EW), and LTF under LLaMA-3.1 Instruct 8B backbone. Entropy quantifies how much information content the latent reasoning trajectory carries. A higher entropy often indicates a richer spread of information across many dimensions, reflecting diverse, less redundant features and better information preservation. Conversely, a lower entropy usually reflects concentrated eigenvalue spectra, suggesting that the latent representations may contain redundant information. Given a latent reasoning trajectory $\tau = (s_0, \mathbf{z}_{1:T})$, we define its *average reasoning entropy* as

$$\Xi(\tau) = \frac{1}{T} \sum_{t=0}^{T-1} \mathcal{H}\left[q_\varphi(\mathbf{z}_{t+1} \mid s_t)\right], \tag{27}$$

where $\mathcal{H}(\cdot)$ denotes differential entropy.

Table 14: Average reasoning entropy with 95% confidence interval of latent reasoning trajectories across different reasoning methods using the LLaMA-3.1 Instruct 8B backbone. Average reasoning entropy $\Xi(\tau)$ is calculated via Eq. (27).

Method	Average Reasoning Entropy $\Xi(\tau)$
CoLaR	0.013 ± 0.009
ReGuLaR	0.019 ± 0.002
LTF w/o EW	0.030 ± 0.006
LTF	0.024 ± 0.003

Our empirical study investigates the property of latent reasoning paths based on test samples drawn from fine-tuning tasks. As shown in Table 14, CoLaR exhibits the lowest reasoning entropy (0.013), suggesting that its latent reasoning trajectories are highly deterministic and potentially prone to trajectory collapse. ReGuLaR yields higher entropy (0.019), indicating that it encourages more stochastic latent reasoning behaviors. LTF w/o entropy weighting further increases the entropy (0.030), producing even more stochastic but less structured reasoning trajectories.

In contrast, LTF achieves lower entropy (0.024) than LTF w/o entropy weighting while still maintaining higher stochasticity than CoLaR and ReGuLaR. This pattern suggests that entropy-weighted sub-trajectories do not simply maximize stochasticity; rather, they regulate the degree of exploration in latent reasoning. By adaptively allocating training credit to sub-trajectories with appropriate uncertainty, LTF encourages reasoning paths that are both diverse and structured, which helps explain its stronger downstream performance.

These observations indicate the existence of an effective entropy regime for latent reasoning and we denote the threshold as *effective entropy threshold*. When reasoning entropy is too low and under the effective entropy threshold, latent trajectories may collapse into overly deterministic patterns, limiting information contained in the latent reasoning paths and weakening generalization. Moderate increases in entropy can improve reasoning performance by encouraging diverse latent trajectories. However, once entropy becomes excessive beyond the effective entropy threshold, the reasoning process may become overly stochastic, making the learned trajectories less reliable and less effective for downstream tasks. Thus, identifying the effective entropy regime is crucial for regulating latent reasoning dynamics and improving reasoning performance.

Distributional Behavior and Adaptive Reasoning. To further characterize LTF's distributional behavior and adaptive reasoning, we report additional Pass@K results under LLaMA-1B. We report Pass@K results on GSM8K-Aug and ASDiv-Aug using LLaMA-1B, together with the average number of latent reasoning steps per rollout.

Table 15: Pass@K performance and average latent reasoning length on GSM8K-Aug and ASDiv-Aug using LLaMA-1B. The reasoning length (#L) is reported per rollout.

Method	GSM8K-Aug				ASDiv-Aug			
	Pass@1		Pass@16		Pass@1		Pass@16	
	Acc. (↑)	#L (↓)	Acc. (↑)	#L (↓)	Acc. (↑)	#L (↓)	Acc. (↑)	#L (↓)
Coconut	20.50	6.00	26.58	6.00	53.18	6.00	60.22	6.00
CoLaR	26.60	5.63	38.71	5.59	67.19	5.19	75.16	5.21
ReGuLaR	34.58	3.69	48.90	3.71	72.19	1.24	81.27	1.25
LTF	37.09	3.35	53.41	3.34	75.11	1.22	86.92	1.22

Two observations emerge from Table 15. First, LTF exhibits stronger multi-sample scaling than ReGuLaR. On GSM8K-Aug, the accuracy advantage of LTF over ReGuLaR increases from 2.51 at Pass@1 to 4.51 at Pass@16. Similarly, on ASDiv-Aug, the advantage increases from 2.92 to 5.65. Importantly, the average reasoning length remains nearly unchanged between Pass@1 and Pass@16. The improvement therefore comes from sampling additional successful trajectories from the learned distribution rather than allocating more computation to each individual rollout.

Second, LTF exhibits substantially different reasoning budgets across the two task distributions. Its average reasoning length is 3.35 steps on GSM8K-Aug but only 1.22 steps on ASDiv-Aug, while remaining both shorter and more accurate than ReGuLaR on each dataset. This result is consistent with the intended adaptive reasoning behavior: rather than using a fixed latent computation budget, LTF learns task-dependent trajectory-length distributions and allocates more latent computation where longer reasoning is beneficial.

Overall, the entropy and Pass@K analyses provide complementary evidence for the distributional formulation of LTF. The learned sampler maintains non-trivial uncertainty over latent continuations, supports multiple successful reasoning trajectories, and adapts its reasoning budget across task distributions.