System-1.5 Reasoning: Traversal in Language and Latent Spaces with Dynamic Shortcuts

Xiaoqiang Wang[1,2] **Suyuchen Wang**[1,2] **Yun Zhu** **Bang Liu**[1,2,3†]
[1]DIRO & Institut Courtois, Université de Montréal
[2]Mila - Quebec AI Institute; [3]Canada CIFAR AI Chair
{xiaoqiang.wang, suyuchen.wang, bang.liu}@umontreal.ca
gabrielzhuyun@gmail.com

Abstract

Chain-of-thought (CoT) reasoning enables large language models (LLMs) to move beyond fast System-1 responses and engage in deliberative System-2 reasoning. However, this comes at the cost of significant inefficiency due to verbose intermediate output. Recent latent-space reasoning methods improve efficiency by operating on hidden states without decoding into language, yet they treat all steps uniformly, failing to distinguish critical deductions from auxiliary steps and resulting in suboptimal use of computational resources. In this paper, we propose System-1.5 Reasoning, an adaptive reasoning framework that dynamically allocates computation across reasoning steps through shortcut paths in latent space. Specifically, System-1.5 Reasoning introduces two types of dynamic shortcuts. The model depth shortcut (DS) adaptively reasons along the vertical depth by early exiting non-critical tokens through lightweight adapter branches, while allowing critical tokens to continue through deeper Transformer layers. The step shortcut (SS) reuses hidden states across the decoding steps to skip trivial steps and reason horizontally in latent space. Training System-1.5 Reasoning involves a two-stage self-distillation process: first distilling natural language CoT into latent-space continuous thought, and then distilling full-path System-2 latent reasoning into adaptive shortcut paths (System-1.5 Reasoning). Experiments on reasoning tasks demonstrate the superior performance of our method. For example, on GSM8K, System-1.5 Reasoning achieves reasoning performance comparable to traditional CoT fine-tuning methods while accelerating inference by over 20× and reducing token generation by 91.0% on average.

1 Introduction

Foundational large language models (LLMs) (Ouyang et al., 2022; Team et al., 2023; Dubey et al., 2024; Hurst et al., 2024; Meta, 2025) has revolutionized natural language processing (Liang et al., 2022; Srivastava et al., 2023; Wang et al., 2024) and demonstrated strong potential in diverse agent applications (Liu et al., 2025), automating real-world tasks across both digital (OpenAI, 2022; Wang et al., 2023; Hong et al., 2023; Wang & Liu, 2024; Qin et al., 2025; Wang et al., 2025) and physical environments (Brohan et al., 2023; Driess et al., 2023; Zheng et al., 2024; Team et al., 2024). Recent advances in test-time scaling paradigms (Snell et al., 2024; Zhang et al., 2025) and the emergence of reasoning-oriented LLMs, also referred to as Large Reasoning Models (LRMs) (Jaech et al., 2024; Guo et al., 2025; Team, 2025; Abdin et al., 2025), have leveraged Chain-of-Thought (CoT) (Wei et al., 2022) to extend LLM reasoning capabilities (Chen et al., 2025; Wu et al., 2025). This approach facilitates a transition from fast and heuristic-driven System-1 reasoning to slower and deliberate

†Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

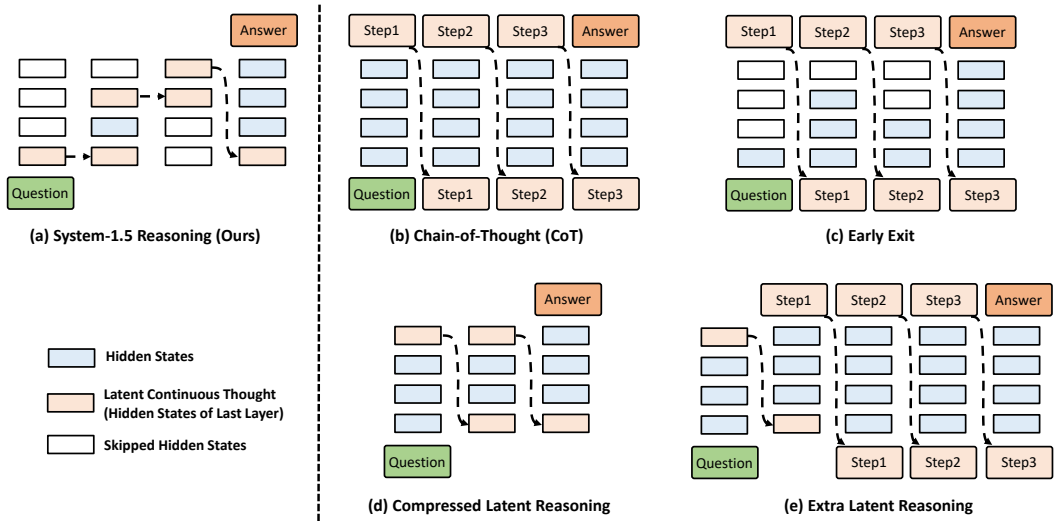

Figure 1: Comparison of (a) our proposed `System-1.5 Reasoning`, (b) chain-of-thought (CoT) reasoning, (c) early-exit (Elbayad et al., 2019; Elhoushi et al., 2024), (d) compressed latent-space reasoning (*e.g.*, Coconut (Hao et al., 2024) and CCoT (Cheng & Van Durme, 2024)), and (e) extra latent reasoning approaches that delay output (*e.g.*, *pause* token (Goyal et al., 2023) and *filler* token (Pfau et al., 2024)). `System-1.5 Reasoning` enables flexible latent reasoning along both the horizontal and vertical dimensions through dynamic shortcuts.

System-2 reasoning (Li et al., 2025b), and has significantly improved performance in competitive mathematics (Hendrycks et al., 2021; Li et al., 2024) and code generation (Jimenez et al., 2023; Jain et al., 2024).

However, CoT reasoning incurs substantial computational costs during inference due to the need to generate long reasoning chains before producing the final answer. It also suffers from overthinking phenomena (Chen et al., 2024; Team et al., 2025), resulting in redundant and verbose outputs even for simple problems, where excessive reasoning leads to unnecessary token generation and escalates inefficiency.

To address it, one line of work focuses on language-space efficiency, such as imposing token budgets during prompt input (Xu et al., 2025a) or decoding interventions (Muennighoff et al., 2025) to constrain the length or complexity (Lee et al., 2025) of generated explanations. Another direction emphasizes efficient decoding, such as speculative decoding (Leviathan et al., 2023) and optimized sampling (Sun et al., 2024b; Fu et al., 2024; Li et al., 2025a).

However, many tokens generated during chain-of-thought (CoT) reasoning primarily serve textual fluency and coherence rather than contributing meaningfully to actual reasoning advancement (Song et al., 2025; Luo et al., 2025a). Consequently, recent approaches have explored operating reasoning in latent space. Methods such as the *pause* token (Goyal et al., 2023) and *filler* token (Pfau et al., 2024) insert special tokens and ignore their language-space outputs to allow for additional latent computation, while Coconut (Hao et al., 2024) and CCoT (Cheng & Van Durme, 2024) compress explicit CoT into hidden states and feed these hidden states, rather than generated tokens, back into the model as subsequent embeddings. By bypassing language constraints, these internal computations enable more compact and flexible reasoning within the latent space.

While these methods improve efficiency, as illustrated in Figure 1, they introduce a unidirectional optimization bias, promoting either universally fast reasoning through compressed latent-space reasoning (*e.g.*, Coconut and CCoT) for all reasoning steps or universally slow reasoning that delays outputs at every step (*e.g.*, *pause* token and *filler* token). This uniform treatment fails to differentiate between critical deductions and auxiliary steps, leading to inefficient allocation where trivial and complex steps receive nearly equal computational budgets. For example, a CoT sequence often includes distinct phases such as problem restatement, exploration, and result verification, each requiring different levels of reasoning effort (Luo et al., 2025b).

This raises a natural question: *Can we dynamically tailor computation across different reasoning steps to maximize efficiency while preserving performance?* Ideally, models should reason quickly (System-1) on non-critical steps and more carefully (System-2) on critical ones.

Motivated by this, we propose `System-1.5 Reasoning`, which adaptively combines fast and slow reasoning paths in latent space to handle varying step complexities in the language space. `System-1.5 Reasoning` introduces model depth shortcut (DS) and decoding step shortcut (SS) to adaptively allocate computation along the vertical and horizontal paths in latent space.

Specifically, the depth shortcut is implemented by augmenting each Transformer layer with a router-adapter module, which dynamically determines whether a token continues through deeper standard layers or exits early via a lightweight adapter branch. This depth shortcut mechanism enables flexible vertical computation along the model depth, allowing non-critical reasoning steps to be processed with fewer layers while critical steps continue deeper into the model. In addition, the step shortcut enables latent-space skipping across the horizontal dimension of decoding length, where early-exited hidden states at a given layer are directly copied to the next decoding step, instead of restarting latent computation from the first layer as in standard Transformers.

By supporting adaptive reasoning along both vertical and horizontal paths, `System-1.5 Reasoning` maximizes flexibility in latent-space reasoning and better mirrors human thinking, where difficult reasoning steps are handled with deliberate, reflective System-2 thinking, simple steps are processed quickly through heuristic System-1 thinking, and trivial steps are naturally skipped without thinking.

To enable latent-space shortcut reasoning, we train `System-1.5 Reasoning` via a two-stage distillation process. The first stage, language-to-latent distillation, aligns the latent-space behavior of a student model with the language-space CoT reasoning of a teacher model. This process leverages full teacher-forcing and allows efficient parallel training, avoiding the step-wise scheduling complexities of curriculum learning approaches such as iCoT (Deng et al., 2024) and Coconut (Hao et al., 2024).

The second stage, System-2 to System-1.5 distillation, further compresses full-depth reasoning trajectories into shortcut execution path by leveraging reasoning criticality estimation in the language space. Specifically, we freeze the original transformer parameters and tune only the router-adapter module using an early-exit loss that encourages non-critical tokens to exit at earlier layers and critical tokens to proceed to deeper layers, while maintaining consistency between the hidden states of the full path and those of the shortcut-exited path.

We validate our approach on challenging reasoning datasets such as GSM8K (Cobbe et al., 2021). Experiments demonstrate that `System-1.5 Reasoning` achieves reasoning performance comparable to traditional CoT fine-tuning methods while accelerating inference by over $20\times$ and reducing token generation by 91.0% on average. These results highlight the promise of `System-1.5 Reasoning` in improving the efficiency and scalability of the LLM reasoning.

2 System-1.5 Reasoning

We frame `System-1.5 Reasoning` as adaptive and dynamic latent-space reasoning guided by criticality analysis of language-space reasoning. As shown in Figure 2, the core idea is to dynamically allocate computation in latent space through two types of **dynamic shortcuts**, model depth shortcut (DS) and decoding step shortcut (SS), to handle varying step complexities in the language space. Specifically, by inserting a standard router-adapter module into each vanilla Transformer layer, the dynamic depth shortcut enables simple steps to be processed through shallow layers, while complex steps are routed through deeper layers for more extensive computation. In parallel, the dynamic step shortcut allows trivial steps to be skipped by copying hidden states at early exit points and directly reusing them as the hidden states for the next decoding step at the same layer.

To train `System-1.5 Reasoning` effectively in latent space, we employ a **two-stage distillation** process. First, we perform language-to-latent distillation by fine-tuning vanilla Transformer layers, aligning the student model's hidden states, which reason in latent space with those of the CoT-trained teacher model. Then, with the vanilla Transformer parameters frozen, we perform System-2 to System-1.5 distillation by training the router-adapter modules. Specifically, we leverage atomic thought decomposition (Teng et al., 2025) to estimate reasoning criticality step by step. We decompose the CoT into a directed acyclic graph (DAG) of self-contained subquestions. In this graph, independent nodes are labeled as non-critical, while derived nodes that require logical integration are labeled as

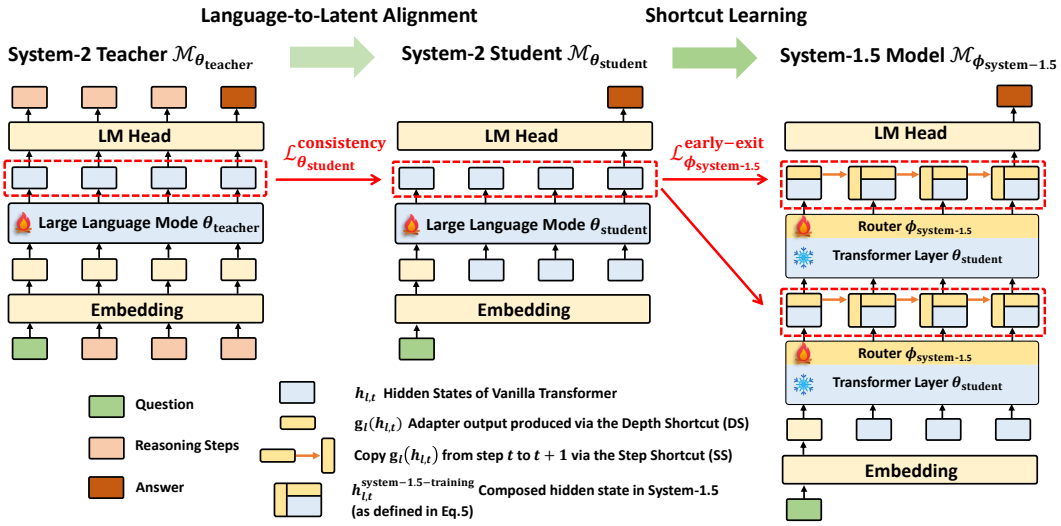

Figure 2: `System-1.5 Reasoning` is trained through a two-stage distillation process: (1) Language-to-latent alignment, where the model learns to reason in latent space by minimizing a consistency loss between a language-space reasoning model (System-2 teacher) and a latent-space reasoning model (System-2 student); and (2) Shortcut learning, where dynamic shortcuts are trained by applying an early-exit loss that encourages non-critical steps to exit earlier and critical steps to proceed deeper. The model is simplified to a 2-layer Transformer for illustration purposes.

critical. Based on this, we apply an early-exit loss that encourages tokens of non-critical steps to exit earlier and tokens of critical steps to proceed to deeper layers.

2.1 Dynamic Shortcut Architecture

Formally, given an input text sequence $X = \langle x_1, \cdots, x_t, \cdots, x_T \rangle$, we denote the vanilla hidden states at the l-th layer as $H_l = \langle h_{l,1}, \cdots, h_{l,t}, \cdots, h_{l,T} \rangle$, where T is the sequence length and $l \in \{0, 1, \cdots, L\}$ with L being the total number of Transformer layers. The initial hidden states are given by $H_0 = \text{Embed}(X)$, corresponding to the embedding layer. For $l \geq 1$, denoting $f_l(\cdot)$ as the operation of a single Transformer layer, the hidden state at layer l is updated as $H_l = f_l(H_{l-1})$.

Depth shortcut. To adaptively determine how many of the initial Transformer layers are needed for each token, *i.e.*, when to trigger early exit, we employ a lightweight router-adapter module, as shown in Figure 2. At each Transformer layer l, the router module $\mathcal{R}_l$ dynamically decides whether a token should continue through the standard Transformer layer f_l for deeper processing, or exit early through an adapter branch g_l. Formally, during training, the output at layer l for step t is expressed as a weighted combination of the adapter and Transformer outputs, where the weight is determined by the binary router $\mathcal{R}_l$, implemented as a feed-forward network (FFN) layer followed by a sigmoid activation:

$$h^{\text{system-1.5-ds-training}}_{l,t} = g_{l-1}(h_{l-1,t}) * w + f_l(h_{l-1,t}) * (1 - w) \tag{1}$$

$$w = \mathcal{R}_l(h_{l-1,t}) \tag{2}$$

During inference, the output of the router $\mathcal{R}l$ serves as a confidence score for early exit, which is compared against a predefined depth exit threshold λ_{depth} to determine whether to halt computation at the current layer for the given decoding step.

$$h^{\text{system-1.5-ds-inference}}_{l,t} = \begin{cases} g_{l-1}(h_{l-1,t}), & \text{if } \mathcal{R}_l(h_{l-1,t}) > \lambda_{\text{depth}}, \\ f_l(h_{l-1,t}), & \text{otherwise.} \end{cases} \tag{3}$$

Step shortcut. The step shortcut is motivated by the observation that, in standard decoding, each step still forces the model to process from the first layer. In other words, while depth shortcuts allow adaptive reasoning along the vertical path in latent space, enabling exits at intermediate layers rather than always reaching the final layer, the model must still sequentially process every decoding step. To address this, we introduce dynamic shortcuts along the decoding steps, allowing the model

to adaptively skip steps and reason horizontally in latent space. By supporting adaptive reasoning along both vertical and horizontal paths, this design maximizes flexibility in latent-space reasoning and enables `System-1.5 Reasoning` to better mirror human thinking, where trivial steps are often skipped (step shortcut), difficult steps are handled with deliberate and reflective reasoning, and simple steps are processed quickly through heuristic reasoning (depth shortcut). More importantly, since `System-1.5 Reasoning` operates entirely in latent space, where new decoding steps rely solely on the hidden states from the previous step without introducing new input language tokens, the practice of step shortcut avoids the problem of partial observation of the input context (*i.e.*, skipping token input at the current step) in token compression methods. This allows `System-1.5 Reasoning` to achieve high computational efficiency while preserving all crucial information and thinking steps.

Formally, similar to the depth shortcut, during training, the output at layer l and step t is computed as a weighted combination of the hidden state from step $t-1$ at the same layer (note that the step shortcut is not applied when $t = 0$) and the hidden state at step t from the vanilla Transformer:

$$h_{l,t}^{\text{system-1.5-ss-training}} = g_l(h_{l,t-1}) * w + f_l(h_{l-1,t}) * (1 - w) \tag{4}$$

where the weight w is determined by the router $\mathcal{R}_l$ as Eq. 2. By merging Eq. 1 and Eq. 4, we obtain:

$$h_{l,t}^{\text{system-1.5-training}} = \Big(\underbrace{g_{l-1}(h_{l-1,t})}_{\text{depth shortcut}} + \underbrace{g_l(h_{l,t-1})}_{\text{step shortcut}} \Big) * w + \underbrace{f_l(h_{l-1,t})}_{\text{vanilla path}} * (1 - w) \tag{5}$$

During inference, as determined by Eq. 3, if a decoding step halts computation at an intermediate layer, the hidden state at that layer is directly passed to the next decoding step within the same layer to continue reasoning. If computation halts at the final layer (*i.e.*, after traversing all the Transformer layers via dynamic shortcuts, termed as a reasoning *cycle*), we either (1) feed the final hidden state back for the next reasoning cycle or (2) generate the final output. Borrowing the latent reasoning setting from (Hao et al., 2024), we apply a fixed number of latent reasoning steps, denoted as the decoding step constant λ_{step}. By combining depth exit threshold λ_{depth} in Eq. 3 and decoding step constant λ_{step} to control computation along both the model depth and decoding steps, this approach provides a flexible computational budget for test-time scaling (see Section 3.2).

2.2 Two-Stage Distillation: Language-to-Latent Alignment and Shortcut Learning

Training a model to reason in latent space with shortcut paths poses two challenges: the latent property and the adaptive property. (1) Vanilla Transformer layers in pre-trained LLMs are optimized for next-token prediction in the language space and are not inherently capable of reasoning in latent space. How can we align latent-space reasoning with language-space CoT? (2) Adaptive reasoning needs the model to dynamically decide varying reasoning paths in the latent space. How can we potentially leverage the characteristics of language-space reasoning steps (*e.g.* reasoning step criticality) to guide the model toward learning adaptive reasoning via shortcut paths?

As shown in Figure 2, for the latent property challenge, we employ **language-to-latent alignment** through hidden-state distillation. Specifically, we align the reasoning processes between a language-space teacher model and a latent-space student model. The teacher model, denoted as $\mathcal{M}_{\theta_{\text{teacher}}}$, is trained with standard CoT fine-tuning to generate both intermediate steps and final answers in the language space. The student model, denoted as $\mathcal{M}_{\theta_{\text{student}}}$, learns to reason directly in latent space.

For the adaptive property challenge involving dynamic shortcut decisions, we leverage atom-of-thought (Teng et al., 2025) to decompose the original CoT into a directed acyclic graph (DAG) of self-contained subquestions. Independent nodes are identified as non-critical steps (requiring less computation), while derived nodes requiring logical integration are identified as critical steps (requiring deeper reasoning). Based on this, we introduce **shortcut learning**: We first initialize the System-1.5 model, denoted as $\mathcal{M}_{\phi_{\text{system-1.5}}}$, by inheriting the parameters from $\mathcal{M}_{\theta_{\text{student}}}$. We then freeze the Transformer parameters and insert an adapter-router module into each Transformer layer to enable fine-tuning for adaptive reasoning. Lastly, we apply an early-exit loss that encourages non-critical steps to exit at shallower layers while allowing critical steps to proceed deeper into the model.

Language-to-latent alignment. During training, we extract the last-layer hidden states from the teacher model, apply stop-gradient, and use them as ground-truth features for the student model. These

features are then fed into the student model together with the question input, enabling teacher-forcing and ensuring training efficiency.

Formally, given an input sequence $X = \langle x_1, \ldots, x_T \rangle$, we further represent it as the concatenation of three segments, the question Q, intermediate reasoning steps R, and the answer A, denoted as $X = \langle Q : R : A \rangle = \langle q_1, \ldots, q_{T_Q}, r_1, \ldots, r_{T_R}, a_1, \ldots, a_{T_A} \rangle$, where T_Q, T_R, and T_A correspond to the lengths of each segment.

The last-layer hidden states from the System-2 teacher and student models, denoted by $h_{L,t}^{\text{teacher}}$ and $h_{L,t}^{\text{student}}$ respectively, are aligned by minimizing the following mean squared error (MSE) loss:

$$\mathcal{L}_{\theta_{\text{student}}}^{\text{consistency}} = \frac{1}{T_A} \sum_{t=T_Q+T_R+1}^{T_Q+T_R+T_A} \text{MSE}(\text{sg}[h_{L,t}^{\text{teacher}}], h_{L,t}^{\text{student}}) \tag{6}$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operation applied to the teacher model.

In parallel, the System-2 teacher model is supervised by the standard negative log-likelihood (NLL) loss over both intermediate reasoning steps and final answers, while the System-2 student model is supervised by NLL loss over final answer generation only (*i.e.*, performing latent reasoning for intermediate steps):

$$\mathcal{L}_{\theta_{\text{teacher}}}^{LM} = -\sum_{t=T_Q+1}^{T_Q+T_R} \log \mathcal{M}_{\theta_{\text{teacher}}}(r_t \mid r_{<t}, Q) - \sum_{t=T_Q+T_R+1}^{T_Q+T_R+T_A} \log \mathcal{M}_{\theta_{\text{teacher}}}(a_t \mid a_{<t}, R, Q) \tag{7}$$

$$\mathcal{L}_{\theta_{\text{student}}}^{LM} = -\sum_{t=T_Q+T_R+1}^{T_Q+T_R+T_A} \log \mathcal{M}_{\theta_{\text{student}}}\left(a_t \mid a_{<t}, H_L^{\text{teacher}}, Q\right) \tag{8}$$

where $H_L^{\text{teacher}} = \{h_{L,T_Q}^{\text{teacher}}, h_{L,T_Q+1}^{\text{teacher}}, \cdots, h_{L,T_Q+T_R-1}^{\text{teacher}}\}$ denotes the extracted last-layer hidden states of the reasoning steps, provided as input to the student model to enable teacher-forcing. The overall loss for language-to-latent alignment is thus:

$$\mathcal{L}_1 = \mathcal{L}_{\theta_{\text{teacher}}}^{LM} + \mathcal{L}_{\theta_{\text{student}}}^{LM} + \alpha \mathcal{L}_{\theta_{\text{student}}}^{\text{consistency}} \tag{9}$$

where θ_{teacher} and θ_{student} refer to the original Transformer parameters of the System-2 teacher and System-2 student models, respectively.

Shortcut learning. Given a dataset with labeled intermediate reasoning steps $R = \langle r_1, r_2, \cdots, r_{T_R} \rangle$ and estimated criticality binary labels for each step $(r_1, c_1), \ldots, (r_{T_R}, c_{T_R})$, we define an early-exit loss (Elbayad et al., 2019; Elhoushi et al., 2024) that enforces consistency between the intermediate hidden states of the System-1.5 model and the final-layer hidden states of the System-2 student model (which has been trained through language-to-latent alignment):

$$\mathcal{L}_{\phi_{\text{system-1.5}}}^{\text{early-exit}} = \sum_{l=1}^{L} \sum_{t=T_Q+1}^{T_Q+T_R} e_{l,t} \text{MSE}(\text{sg}[h_{l,t}^{\text{student}}], h_{l,t}^{\text{system-1.5-training}}) \tag{10}$$

where $e_{l,t}$ is the early-exit weight for the l-th layer at step t, designed to distinguish critical and non-critical steps. Specifically, we define $e_{l,t}$ to encourage non-critical steps ($c_t = 0$) to exit earlier, and critical steps ($c_t = 1$) to exit later, according to:

$$e_{l,t} = (1 - c_t) \frac{\sum_{i=1}^{l} i}{\sum_{j=1}^{L} j} + c_t \left(1 - \frac{\sum_{i=1}^{l} i}{\sum_{j=1}^{L} j}\right). \tag{11}$$

In parallel, the System-1.5 model is also supervised by a NLL loss over the final answer generation:

$$\mathcal{L}_{\phi_{\text{system-1.5}}}^{LM} = -\sum_{t=T_Q+T_R+1}^{T_Q+T_R+T_A} \log \mathcal{M}_{\phi_{\text{system-1.5}}}\left(a_t \mid a_{<t}, H_L^{\text{student}}, Q\right) \tag{12}$$

Table 1: Quantitative results on reasoning tasks, reported in terms of accuracy (**Acc.**), number of decoding steps before generating the final answer (**# Steps**), average FLOPs reduction rate per step relative to CoT (**FLOPs r.**), and overall inference speedup relative to CoT measured by wall-clock time. Best and second-best results are highlighted with **bold** and underline, respectively.

Method	GSM8K				GSM-HARD				StrategyQA			
	Acc. (%)	# Steps	FLOPs r.	Speedup	Acc. (%)	# Steps	FLOPs r.	Speedup	Acc. (%)	# Steps	FLOPs r.	Speedup
CoT	**46.94**	26	-	-	**38.32**	26	-	-	47.62	52	-	-
LITE	44.51	28	1.84×	1.61×	37.01	28	1.56×	1.44×	46.15	42	1.96×	2.36×
LayerSkip	43.20	32	1.77×	1.45×	36.55	33	1.32×	1.03×	42.54	49	1.8×	1.86×
iCoT	32.14	2	1.02×	<u>13.45×</u>	23.17	4	1.02×	6.17×	34.42	2	1.02×	26.47×
Coconut	36.75	2	1.02×	11.98×	28.25	4	1.02×	<u>7.07×</u>	38.67	2	1.02×	<u>27.25×</u>
CODI	43.78	2	1.02×	13.37×	35.91	4	1.02×	6.93×	45.12	2	1.02×	25.96×
pause token	46.32	38	1.00×	0.8×	38.17	42	1.00×	0.89×	<u>48.14</u>	61	1.00×	0.82×
System-1.5	<u>46.66</u>	2	1.95×	**20.27**×	<u>38.28</u>	4	1.76×	**12.45**×	**48.61**	2	2.12×	**55.65**×

where H_L^{student} extracts the System-2 student model to enable teacher-forcing. The overall loss for shortcut learning is then given by:

$$\mathcal{L}_2 = \mathcal{L}^{LM}_{\phi_{\text{system-1.5}}} + \beta \mathcal{L}^{\text{early-exit}}_{\phi_{\text{system-1.5}}} \tag{13}$$

where $\phi_{\text{system-1.5}}$ refers to the parameters of the router-adapter modules. The underlying Transformer parameters are initialized from θ_{student} and are kept frozen during shortcut learning.

3 Experiments

Datasets. We evaluate the effectiveness and efficiency of `System-1.5 Reasoning` on two reasoning-intensive tasks: mathematical reasoning and common sense reasoning. For mathematical reasoning, we train on the augmented GSM8K dataset (Deng et al., 2023), which extends the original GSM8K (Cobbe et al., 2021) with a larger set of grade school-level math problems. For commonsense reasoning, we use StrategyQA (Geva et al., 2021), which contains multihop questions annotated with supporting facts. Each sample includes a question, decomposed subquestions, and a yes/no answer. We merge the annotated facts and subquestions as a coherent CoT sequence for training. For both datasets, we label each reasoning step with criticality annotations using atomic thought decomposition, as described in Section 2.2, to distinguish between critical and non-critical steps. We train models on the official training splits and evaluate performance on the respective test sets for in-domain evaluation. Additionally, for mathematical reasoning, we conduct out-of-domain evaluation on GSM-HARD (Gao et al., 2023), a dataset with increased reasoning difficulty designed to test generalization beyond the original GSM8K.

Baselines. In addition to CoT fine-tuning, we compare `System-1.5 Reasoning` against six efficient reasoning methods based on supervised fine-tuning, consisting of: (1) language-space conditional computation methods: LITE (Varshney et al., 2023) and LayerSkip (Elhoushi et al., 2024), which improve efficiency by applying early exit mechanisms; (2) Latent-space compressed reasoning: iCoT (Deng et al., 2024), Coconut (Hao et al., 2024), and CODI (Shen et al., 2025), which compress natural language CoT into compact continuous latent representations; (3) Latent-space extended reasoning methods: *pause* token (Goyal et al., 2023), which insert extra latent states to delay output and allow for extended internal reasoning. Since the official implementations of iCoT and Coconut are based on GPT-2 124M (Radford et al., 2019), while LayerSkip builds on the LLaMA 2 (Touvron et al., 2023) and LLaMA 3 (Grattafiori et al., 2024) series model, we implement two backbone versions of `System-1.5 Reasoning`, one using GPT-2 124M and another using LLaMA 3.2 1B, to ensure fair comparisons between different baselines.

3.1 Main Results

`System-1.5 Reasoning` **outperforms previous state-of-the-art methods in latent-space reasoning in both accuracy and efficiency.** As shown in Table 1, compared to iCoT, Coconut, CODI, and *pause* token, `System-1.5 Reasoning` achieves higher accuracy and greater overall speedup. In GSM8K, compared to original CoT reasoning, latent-space reasoning reduces intermediate token generation by 92.31% during inference. This reduction is even more pronounced on StrategyQA, reaching 96.15%. Additionally, compared to early-exit methods, `System-1.5 Reasoning` achieves a 1.95× reduction in average FLOPs per decoding step, due to its dynamic step shortcut mechanism,

which allows hidden states from early-exit layers to be directly copied and reused in the next decoding step. In contrast, early-exit methods still reprocess these states starting from the first layer.

By combining early exits along the model depth and shortcut copying across decoding steps, `System-1.5 Reasoning` achieves an overall inference speedup of 20.27× on GSM8K, further improving upon the approximately 10× speedups achieved by previous latent reasoning methods. On StrategyQA, it further accelerates inference by over 55×, significantly enhancing reasoning efficiency.

`System-1.5 Reasoning` **matches CoT fine-tuning on challenging mathematical reasoning tasks and outperforms it on text-rich commonsense reasoning.** On GSM8K and GSM-HARD, `System-1.5 Reasoning` achieves 46.94% and 38.32% accuracy respectively, closely matching CoT fine-tuning results of 46.67% and 38.28%. On StrategyQA, a task requiring reasoning over multiple pieces of textual evidence, `System-1.5 Reasoning` achieves 48.61% accuracy, outperforming CoT's 47.36%. Similar improvements are observed for latent-space reasoning methods such as *pause* token, highlighting the promising potential of `System-1.5 Reasoning` in reducing dependency on explicit textual tokens and enhancing the model's inherent reasoning capabilities.

3.2 In-depth Analysis

Direct distillation from language to latent is more effective for shortcut learning in `System-1.5 Reasoning`. We modify the language-to-latent alignment into a curriculum-based learning process, following the approach used in Coconut. This curriculum consists of multiple stages where natural language thoughts are progressively replaced with latent-space thoughts to train the latent reasoning model (analogous to using a fine-tuned Coconut model as the System-2 student). We then apply the same shortcut learning procedure to train `System-1.5 Reasoning`. We follow Coconut's original training schedule, using six epochs in the initial stage and three in each subsequent stage. For comparison, we also evaluate CODI, a distillation-based latent reasoning model that performs competitively in Table 1, as the System-2 student for shortcut learning.

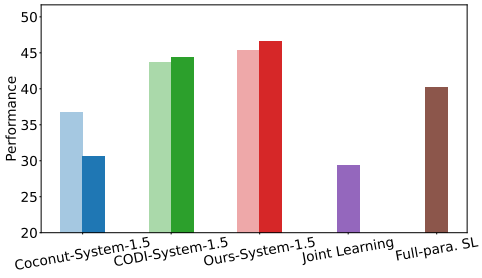

Figure 3: Ablation results on optimizing System-1.5 (shown in solid color) using different System-2 students (shown in light color), namely Coconut-System-1.5 and CODI-System-1.5, as well as alternative learning strategies: joint learning of language-to-latent alignment and shortcut learning, and full-parameter shortcut learning (SL).

Figure 3 shows the results using different System-2 students to train System-1.5 model. We observe a significant accuracy drop in the final `System-1.5 Reasoning` performance when using Coconut as the System-2 student. In addition to Coconut's suboptimal performance observed in Table 1, another possible reason for the gap is that its hard curriculum schedule between language and latent-space reasoning limits the flexibility and completeness of latent-space modeling. In contrast, `System-1.5 Reasoning` leverages both "horizontal" and "vertical" shortcut across decoding steps and model depth, requiring a more flexible latent reasoning structure that hard curriculum distillation cannot effectively support.

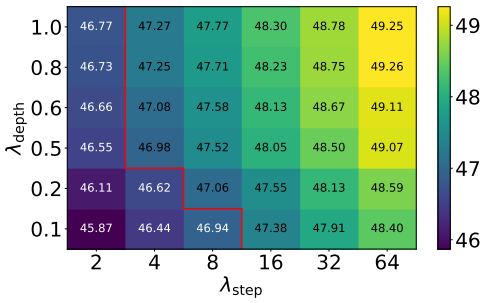

Figure 4: Controllable test-time scaling by tuning the depth exit threshold λ_{depth} and decoding step constant λ_{step} in `System-1.5 Reasoning`. The red grid-aligned Pareto boundary highlighting the frontier of configurations surpassing the CoT baseline (46.94% accuracy in Table 1).

Joint learning and full-parameter shortcut learning degrade the performance of `System-1.5 Reasoning`. We further investigate the training strategy of `System-1.5 Reasoning` by exploring two variants: (1) Joint learning, where language-to-latent alignment and shortcut learning are conducted simultaneously, *i.e.*, distilling the hidden states of natural language thought directly into the shortcut-exited hidden states; and (2) Full-

parameter shortcut learning, where instead of freezing the original Transformer parameters, all model parameters are updated during shortcut learning.

As shown in Figure 3, both joint learning and full-parameter shortcut learning degrade the final performance of `System-1.5 Reasoning`, with joint learning exhibiting a more pronounced drop. We attribute this to optimization conflicts between the two groups of parameters: Transformer parameters responsible for latent reasoning and router parameters responsible for shortcut routing. These conflicts likely hinder the model's ability to simultaneously preserve flexible latent reasoning paths and optimize dynamic shortcut decisions.

`System-1.5 Reasoning` **enables flexible budget-controllable test-time scaling.** Unlike language-space reasoning, where scaling test-time computation often requires exploring complex structural dependencies (*e.g.*, tree-like exploration) or enforcing solution consistency across multiple trajectories (Zhang et al., 2025), `System-1.5 Reasoning` allows for fine-grained control over computational budgets through simple threshold tuning during inference. Specifically, `System-1.5 Reasoning` inherently introduce two control parameters: the depth exit threshold λ_{depth}, which adjusts the adaptive computation depth for each decoding step (vertically across model layers, where $\lambda_{\text{depth}} = 1$ forces the computation to proceed to the final layer as in a standard Transformer) , and the decoding step constant λ_{step}, which determines when to halt intermediate latent thought generation and output a final answer (horizontally across decoding steps).

Figure 4 shows the performance under different configurations of λ_{depth} and λ_{step}. We observe that performance is approximately equally sensitive to adjustments along both dimensions, further validating the motivation for adaptive reasoning across both the model depth and decoding steps. Moreover, depth scaling saturates more quickly, and deeper reasoning demands significantly higher training computation, suggesting a synergistic relationship between train-time scaling and flexible test-time scaling in `System-1.5 Reasoning`.

4 Related Works

Efficient reasoning models. Efficient reasoning models aim to mitigate the inefficiencies caused by verbose outputs and the overthinking phenomenon (Chen et al., 2024; Team et al., 2025). One line of work focuses on improving language-space efficiency, using length budgeting through prompting (Lee et al., 2025) and fine-tuning (Liu et al., 2024; Yu et al., 2024; Luo et al., 2025a). For example, Chain-of-Draft (Xu et al., 2025a) applies token budget-aware prompting to guide the model toward more concise reasoning, while S1 (Muennighoff et al., 2025) introduces a budget-forcing strategy to terminate the thinking process early. Another line of work focuses on compressing language-space reasoning into latent space, exemplified by iCoT, Coconut (Hao et al., 2024), and CCoT (Cheng & Van Durme, 2024), which train models to reason through compact internal representations. *Pause* token (Goyal et al., 2023) and *filler* token (Pfau et al., 2024) further extend latent-space reasoning by inserting special tokens that delay output and allow for additional latent computation. More recently, Token Assorted (Su et al., 2025) mixes latent and language thoughts by abstracting early steps into discrete latent codes, SoftCoT (Xu et al., 2025b) mitigates catastrophic forgetting in latent reasoning via prompt tuning, and CODI (Shen et al., 2025) applies hidden-state distillation to enhance latent reasoning performance. `System-1.5 Reasoning` builds upon latent-space reasoning but further optimizes efficiency by adaptively allocating computation to handle reasoning steps with varying complexity.

Conditional computation. Conditional computation techniques aim to selectively apply heavy computation only to the important parts of an input, thereby reducing unnecessary resource usage. One line of work focuses on system or model switching (Qu et al., 2025), where inputs are routed between different reasoning systems—for example, a fast System-1 and a slower, more deliberate System-2. System-1.x (Saha et al., 2024) combines linear reasoning chains and search trees to enable fast and accurate planning in maze navigation tasks, while FaST (Sun et al., 2024a) uses switch adapters to dynamically toggle between System-1 and System-2 modes based on task complexity factors such as visual uncertainty or occlusion. Another line of work explores sparse activation via mixture-of-experts models (Jiang et al., 2024; Raposo et al., 2024), where only a subset of the model is activated during inference. Beyond expert routing, dynamic depth models adaptively apply only a portion of the Transformer layers. Early exit (EE) and skip-layer techniques are common approaches: DeeBERT (Xin et al., 2020) and CascadeBERT (Li et al., 2021) apply EE in encoder-

only architectures, while more recent decoder-only LLMs like LITE (Varshney et al., 2023) and SkipDecode (Del Corro et al., 2023) employ confidence-based exits to speed up inference. Other approaches, such as CoLT5 (Ainslie et al., 2023) and LayerSkip (Elhoushi et al., 2024), perform binary routing to skip redundant attention layers or entire blocks. `System-1.5 Reasoning` draws inspiration from this line of work and extends it to the latent reasoning space. It not only adapts computation along the model depth via early exits but also introduces dynamic shortcuts across decoding steps, enabling reasoning to proceed efficiently both vertically and horizontally.

5 Conclusion

We introduce `System-1.5 Reasoning`, a novel reasoning framework that improves inference efficiency in large language models by enabling dynamic shortcut reasoning within latent space. `System-1.5 Reasoning` adaptively allocates computation based two forms of latent-space shortcuts: depth shortcuts, which modulate layer-wise computation, and step shortcuts, which streamline decoding steps. To support this adaptive reasoning, `System-1.5 Reasoning` is trained via a two-stage distillation process: first aligning language-space and latent-space reasoning, and then learning shortcut execution paths guided by stepwise reasoning criticality. This training paradigm ensures that the model captures both the expressiveness of System-2 reasoning and the efficiency of System-1-style shortcuts. Experimental results on challenging reasoning benchmarks demonstrate that `System-1.5 Reasoning` preserves the accuracy of traditional chain-of-thought methods while accelerating inference by more than $20\times$. These results highlight the potential of `System-1.5 Reasoning` as an effective and scalable latent-space reasoning framework for future LLM deployment.

Acknowledgements

This work is supported by the Canada CIFAR AI Chair Program and the Canada NSERC Discovery Grant (RGPIN-2021-03115).

References

Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, et al. Phi-4-reasoning technical report. *arXiv preprint arXiv:2504.21318*, 2025.

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontanon, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. CoLT5: Faster long-range transformers with conditional computation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5100, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.309. URL `https://aclanthology.org/2023.emnlp-main.309/`.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference. *arXiv preprint arXiv:2307.02628*, 2023.

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.

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.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 8469–8488, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*, 2019.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.681. URL https://aclanthology.org/2024.acl-long.681/.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. Efficiently serving llm reasoning programs with certaindex. *arXiv preprint arXiv:2412.20993*, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021. doi: 10.1162/tacl_a_00370. URL https://aclanthology.org/2021.tacl-1.21/.

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.

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.

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

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.

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.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Ayeong Lee, Ethan Che, and Tianyi Peng. How well do llms compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 19274–19286, 2023.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9, 2024.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. CascadeBERT: Accelerating inference of pre-trained language models via calibrated complete models cascade. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 475–486, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.43. URL https://aclanthology.org/2021.findings-emnlp.43/.

Peiji Li, Kai Lv, Yunfan Shao, Yichuan Ma, Linyang Li, Xiaoqing Zheng, Xipeng Qiu, and Qipeng Guo. Fastmcts: A simple sampling strategy for data synthesis. *arXiv preprint arXiv:2502.11476*, 2025a.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025b.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*, 2025.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. Can language models learn to skip steps? *arXiv preprint arXiv:2411.01855*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025a.

Yijia Luo, Yulin Song, Xingyao Zhang, Jiaheng Liu, Weixun Wang, GengRu Chen, Wenbo Su, and Bo Zheng. Deconstructing long chain-of-thought: A structured reasoning optimization framework for long cot distillation. *arXiv preprint arXiv:2503.16385*, 2025b.

Meta. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation, 2025. URL `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

OpenAI. Chatgpt: Optimizing language models for dialogue, 2022. URL `https://openai.com/blog/chatgpt/`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, et al. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*, 2025.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Swarnadeep Saha, Archiki Prasad, Justin Chih-Yao Chen, Peter Hase, Elias Stengel-Eskin, and Mohit Bansal. System-1. x: Learning to balance fast and slow planning with language models. *arXiv preprint arXiv:2407.14414*, 2024.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 17456–17472, 2022.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *arXiv preprint arXiv:2501.03124*, 2025.

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.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.

Guangyan Sun, Mingyu Jin, Zhenting Wang, Cheng-Long Wang, Siqi Ma, Qifan Wang, Tong Geng, Ying Nian Wu, Yongfeng Zhang, and Dongfang Liu. Visual agents as fast and slow thinkers. *arXiv preprint arXiv:2408.08862*, 2024a.

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290*, 2024b.

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

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL `https://qwenlm.github.io/blog/qwq-32b/`.

Fengwei Teng, Zhaoyang Yu, Quan Shi, Jiayi Zhang, Chenglin Wu, and Yuyu Luo. Atom of thoughts for markov llm test-time scaling. *arXiv preprint arXiv:2502.12018*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Neeraj Varshney, Agneet Chatterjee, Mihir Parmar, and Chitta Baral. Accelerating llama inference by enabling intermediate layer decoding via instruction tuning with lite. *arXiv preprint arXiv:2310.18581*, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Xiaoqiang Wang and Bang Liu. Oscar: Operating system control via state-aware reasoning and re-planning. *arXiv preprint arXiv:2410.18963*, 2024.

Xiaoqiang Wang, Lingfei Wu, Tengfei Ma, and Bang Liu. FAC^{2}E: Better understanding large language model capabilities by dissociating language and cognition. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13228–13243, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.734. URL `https://aclanthology.org/2024.emnlp-main.734/`.

Xiaoqiang Wang, Suyuchen Wang, Yun Zhu, and Bang Liu. R^{3}mem: Bridging memory retention and retrieval via reversible compression. *arXiv preprint arXiv:2502.15957*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Sifan Wu, Huan Zhang, Yizhan Li, Farshid Effaty, Amirreza Ataei, and Bang Liu. Seeing beyond words: Matvqa for challenging visual-scientific reasoning in materials science. *arXiv preprint arXiv:2505.18319*, 2025.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2246–2251, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.204. URL https://aclanthology.org/2020.acl-main.204/.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025a.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. *arXiv preprint arXiv:2502.12134*, 2025b.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. What, how, where, and how well? a survey on test-time scaling in large language models. *arXiv preprint arXiv:2503.24235*, 2025.

Duo Zheng, Shijia Huang, Lin Zhao, Yiwu Zhong, and Liwei Wang. Towards learning a generalist model for embodied navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13624–13634, 2024.

A Limitations and Societal Impact

Limitations. While `System-1.5 Reasoning` shows promising improvements in efficiency and performance, it comes with two main limitations. First, reasoning in latent space, despite being compact and efficient, lacks the interpretability of language-space CoT methods. Without explicit intermediate reasoning steps, it becomes difficult to understand, analyze, or verify the model's internal logic. Second, our evaluation is currently limited to medium-scale benchmarks and model sizes. Future work is needed to validate `System-1.5 Reasoning` across larger-scale settings and a broader range of tasks.

Potential Societal Impact. On the positive side, the efficiency of `System-1.5 Reasoning` could reduce the computational cost of deploying reasoning-capable LLMs, facilitating their broader adoption in real-world applications. This could contribute to more accessible and sustainable AI technologies, potentially democratizing advanced reasoning capabilities for a wider range of users and institutions. On the negative side, the lack of interpretability inherent to latent reasoning poses risks in high-stakes or safety-critical settings. Without transparent intermediate steps, it is harder to detect flawed logic or harmful behavior, which may require stronger safeguards than those typically used for explicit language-based reasoning systems.

B Experiment Setup

Datasets. We evaluate the effectiveness and efficiency of `System-1.5 Reasoning` in two reasoning-intensive tasks: mathematical reasoning and common sense reasoning.

For mathematical reasoning, we utilize the augmented GSM8K dataset (Deng et al., 2023), which extends the original GSM8K dataset (Cobbe et al., 2021) to approximately 400,000 grade school-level math problems. Each problem comprises a question, a chain-of-thought (CoT) that details the intermediate reasoning steps, and a final numerical answer. To assess generalization to more challenging problems, we also perform out-of-domain evaluation using GSM-HARD (Gao et al., 2023), a dataset derived from GSM8K by replacing numerical values with larger, less common numbers, thus increasing the difficulty of the problem.

For commonsense reasoning, we employ the StrategyQA dataset , which consists of 2,780 examples requiring implicit multihop reasoning. Each example includes a concise yes/no question, a decomposition into intermediate reasoning steps, and supporting evidence paragraphs. We construct CoT sequences by merging the annotated decompositions and corresponding evidence, enabling the model to learn structured reasoning paths.

To facilitate adaptive reasoning, we annotate each CoT step in both datasets with criticality labels, identifying steps as either critical or non-critical, using atomic thought decomposition, as described in Section 2.2. For both tasks, we train `System-1.5 Reasoning` on the respective training splits and evaluate performance on the corresponding test sets to assess in-domain effectiveness. The inclusion of GSM-HARD allows us to further evaluate the model's ability to generalize to more complex mathematical reasoning scenarios.

Baselines. Beyond standard CoT fine-tuning, we compare `System-1.5 Reasoning` against six efficient supervised fine-tuning-based reasoning methods, categorized into three classes: (1) Language-space conditional computation methods: LITE (Varshney et al., 2023) and LayerSkip (Elhoushi et al., 2024), which improve efficiency by applying early exit mechanisms on decoding natural language thought sequences; (2) Latent-space compressed reasoning methods: iCoT (Deng et al., 2024), Coconut (Hao et al., 2024), and CODI (Shen et al., 2025), which compress natural language CoT into compact continuous latent representations; (3) Latent-space extended reasoning methods: *pause* token (Goyal et al., 2023), which insert extra latent states to delay output and allow for extended internal reasoning.

Since the official implementations of iCoT and Coconut are based on GPT-2 124M, while LayerSkip builds on the LLaMA 2 and LLaMA 3 series, we implement two backbone versions of `System-1.5 Reasoning`, one using GPT-2 124M and another using LLaMA 3.2 1B, to ensure fair comparisons across different baselines.

Evaluation metrics. Accuracy is evaluated by exact match between the generated final answer and the ground truth: specifically, the numerical value in the GSM8K dataset, and the yes/no label in

StrategyQA. In addition to accuracy, we assess efficiency across three dimensions: decoding steps, computation cost, and overall inference time.

Decoding steps refer to the number of tokens generated before producing the final answer. For latent reasoning models that do not generate explicit tokens, we instead count the number of latent reasoning steps applied. We unify the terminology across all methods by reporting the average number of decoding steps required on the test set.

The computation cost considers both the number of decoding steps and the computational depth traversed within the Transformer layers. That is, although different methods may require a similar number of decoding steps or generated tokens, methods like `System-1.5 Reasoning` and conditional computation models with early exits involve significantly fewer layer computations. Following the setting in previous work (Elbayad et al., 2019; Schuster et al., 2022), we estimate the computational cost using FLOPs (floating point operations), approximating the dot-product and matrix-vector operations within Transformer layers, while omitting nonlinearities, normalization, and residual connections. We compute the average FLOPs per decoding step and report the average reduction in FLOPs relative to CoT fine-tuning.

The overall inference time is measured by recording the actual wall-clock inference time on the test set. We report the average inference speedup relative to CoT fine-tuning.

Implementation Details. We developed our method using PyTorch (Paszke et al., 2019). The base models—GPT-2 124M (Radford et al., 2019) and LLaMA 3.1-1B (Grattafiori et al., 2024)—are initialized from pretrained checkpoints provided by the Hugging Face Transformers library (Wolf et al., 2020). During shortcut learning, we insert a router-adapter module into each Transformer layer. The router module is implemented as a feed-forward network (FFN) followed by a sigmoid activation. The adapter module is implemented using LoRA (Hu et al., 2021), with a scaling factor $\alpha = 32$, rank $r = 8$, and a dropout rate of 0.1. Unless otherwise noted in our test-time scaling analysis (Section 3.2), we set the default depth exit threshold λ_{depth} to 0.6 and the decoding step count λ_{step} to 2. We set the loss coefficient for the language-to-latent alignment (Eq. 9) to $\alpha = 1.0$, and similarly, the loss coefficient for shortcut learning (Eq. 13) to $\beta = 1.0$. Fine-tuning is conducted for 8 epochs using the AdamW optimizer (Loshchilov & Hutter, 2018), with a maximum learning rate of 2×10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and a warmup over 6% of total training steps. We use a batch size of 2. Training is performed on a single NVIDIA RTX A5000 (24 GB) GPU, requiring approximately 26 hours for LLaMA 3.2 1B and 5 hours for GPT-2 (124M) for an 8-epoch run. All experiments are conducted over four independent runs with different random seeds. We report the average results across these runs to ensure statistical stability.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper claims to adaptively allocate computation based on stepwise reasoning complexity, which is realized through dynamic shortcuts along model depth and decoding steps. These contributions are clearly stated in the abstract and introduction and are consistently supported throughout the method and experimental sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We present the limitations of this paper in Appendix A.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [No]

Justification: The paper primarily focuses on running computational experiments to support its motivation and empirical claims.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides all necessary details regarding the experimental setup, implementation, and hyperparameters in Section 3 and Appendix B, ensuring that the main results can be reproduced and the conclusions independently verified.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We provide all necessary details regarding the experimental setup, implementation, and hyperparameters in Section 3 and Appendix B to support reproducibility for reviewer assessment. While the code and data are not publicly available at this stage, we plan to release them upon acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all necessary details regarding the experimental setup, implementation, and hyperparameters in Section 3 and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification:Our method achieves over 20% inference speedup, demonstrating not only statistical but also practical significance. Regarding performance, our claim of comparable accuracy is supported by the consistent results observed across four independent runs with different random seeds (see Appendix B).

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide sufficient information on the compute resources used, including GPU type and training time, in Appendix B to support reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper strictly adheres to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss both the positive and negative societal impacts of our work in Appendix A.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The model is still under development and has not been released for public use. Additionally, no human evaluations involving external annotators have been conducted or disclosed, so safeguards for responsible release are not applicable at this stage.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite the corresponding papers or official sources for all datasets and tools used in our work, and have ensured that all licenses and terms of use are fully respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide a detailed introduction to our proposed efficient reasoning model in Section 2, Section 3, and Appendix B.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We do not use LLMs for the core method development in our work.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.