iCLP: Large Language Model Reasoning with Implicit Cognition Latent Planning

Anonymous authors
Paper under double-blind review

Abstract

Large language models (LLMs), when guided by explicit textual plans, can perform reliable step-by-step reasoning during problem-solving. However, generating accurate and effective textual plans remains challenging due to LLM hallucinations and the high diversity of task-specific questions. To address this, we draw inspiration from human *Implicit Cognition* (*IC*), the subconscious process by which decisions are guided by compact, generalized patterns learned from past experiences without requiring explicit verbalization. We propose *iCLP*, a novel framework that enables LLMs to adaptively generate latent plans (LPs), which are compact encodings of effective reasoning instructions. *iCLP* first distills explicit plans from existing step-by-step reasoning trajectories. It then learns discrete representations of these plans via a vector-quantized autoencoder coupled with a codebook. Finally, by fine-tuning LLMs on paired latent plans and corresponding reasoning steps, the models learn to perform implicit planning during reasoning. Experimental results on mathematical reasoning and code generation tasks demonstrate that, with *iCLP*, LLMs can plan in latent space while reasoning in language space. This approach yields significant improvements in both accuracy and efficiency and, crucially, demonstrates strong cross-domain generalization while preserving the interpretability of chain-of-thought reasoning.

1 Introduction

Large language models (LLMs) employ a step-by-step reasoning process expressed as a chain of thought (CoT) Wei et al. (2022) to solve complex problems. Guiding thought generation with explicit plans Yao et al. (2022), which are step-wise and question-specific reasoning instructions, is crucial for improving practical usage and reliability of LLMs Huang et al. (2024; 2022). However, preparing such plans effectively for accurate reasoning is inherently challenging Valmeekam et al. (2023; 2024), particularly given the high diversity of problems across different tasks.

One preliminary approach to achieving this goal involves prompting LLMs to generate explicit plans using their internal knowledge Yao et al. (2022); Wang et al. (2023a); Sun et al. (2023). However, this method is limited by errors in the plans, which arise from the inevitable hallucinations of LLMs. While leveraging external knowledge bases can help mitigate these errors Lyu et al. (2024); Zhu et al. (2024), accessing useful information from them is time-consuming, and many tasks lack effective knowledge bases altogether. More promising recent efforts Yao et al. (2022); Jiao et al. (2024); Qiao et al. (2024b); Brahman et al. (2024) focus on fine-tuning LLMs on automatically or manually synthesized samples with explicit plans. Unfortunately, LLMs fine-tuned in this manner still struggle to achieve better performance because the plans required by problems within a single task, as well as across tasks, are vast in number and highly diverse.

We argue that these mechanisms do not align with human wisdom, known as *Implicit Cognition* (*IC*) Kihlstrom et al. (1995), through which we learn from experiences to summarize implicit patterns that shape our subconscious mind Locke & Kristof (1996), allowing us to use these patterns to solve new problems without explicitly verbalizing them. TThese patterns contain abstract rules that reflect high-level common knowledge, capable of generalizing across different problems. Additionally, our preliminary experiments on visualizing the representations of explicit plans distilled from CoTs of different questions reveal clear clustering for plans from the same task, along with a certain level of overlap indicating their reuse.

Therefore, this paper mimics IC by proposing a framework called *iCLP*, which enables any LLM to generate latent plans (LPs) in a hidden space, effectively guiding step-by-step reasoning in the language space. Our key insight for its effectiveness is that LPs, built upon summarizing experiences, are analogous to the subconscious mind in humans. Similar to how the subconscious mind serves as a flexible and adaptive guidance system in the brain for diverse problems Murphy (1963), LPs, due to their commonality and reusability for reasoning guidance, are small in scale, making them generalizable across tasks.

To construct this space, *iCLP* first prompts an off-the-shelf LLM to summarize explicit plans from a collection of effective CoT traces. Subsequently, *iCLP* borrows the encoding module of ICAE Ge et al. (2024) to map these distilled plans into a small set of memory slots that compress their semantics; it then derives generic slot representations from a codebook learned via a vector-quantized autoencoder Esser et al. (2021), trained end-to-end with plan reconstruction. By treating codebook indices as special tokens for the LLM, we directly obtain the *latent plans*, which serve as compact encodings of the plans within the learned codebook. Finally, by integrating them into the original samples, we reformulate each sample into the form: (*user*: question, *assistant*: latent plans and CoTs). Fine-tuning any LLM on these samples enables the model to internalize the intelligence of *IC*, empowering it to perform latent planning for reliable, step-wise reasoning.

We conduct evaluations on mathematical reasoning and code generation tasks. For accuracy, supervised fine-tuning of small LLMs, such as Qwen2.5-7B, with *iCLP* on datasets like `MATH` and `CodeAlpaca` yields substantial gains, achieving performance competitive with GRPO Shao et al. (2024), which relies on reinforcement learning. For efficiency, LLMs enhanced with *iCLP* reduce token cost by 10% on average compared to zero-shot CoT prompting. For generality, cross-dataset evaluations show that fine-tuned models applied to `AIME2024` and `MATH-500` achieve more than a 10% average accuracy improvement over base models. Similarly, on `HumanEval` and `MBPP`, we observe a 9% gain. Moreover, LLMs fine-tuned with *iCLP* outperform all baselines, including those trained with long CoT samples and latent CoT reasoning, while maintaining interpretability.

2 Preliminary and Motivation

2.1 Step-wise Reasoning with Planning

Given a question Q, the large language model (LLM) denoted as f with parameters $\boldsymbol{\theta}$, generates a chain of thoughts (CoTs), denoted as $\boldsymbol{c}_{1\ldots n} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n]$, where each $\boldsymbol{c}_i$ with $i \in [1, \ldots, n]$ is a textual description of the thought at the i-th reasoning step. Each thought derives formally from conditional sampling $\boldsymbol{c}_i \sim f_{\boldsymbol{\theta}}\left(\boldsymbol{c}_i | Q, \boldsymbol{c}_{1\ldots i-1}\right)$. The target of n steps reasoning is to produce the predicted solution $\widetilde{y}$ in $\boldsymbol{c}_n$ to match the ground truth y. With *explicit plans* as shown in Figure 2, denoted as $\boldsymbol{p}_{1\ldots n} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n]$, where each represents a textual instruction outlining *what to do* in a single reasoning step, the LLM can gain prior knowledge on how to organize $\boldsymbol{c}_{1\ldots n}$ for reliable problem-solving. Thus, we reformulate CoTs as $\boldsymbol{c}_{1\ldots n} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n \mid \boldsymbol{p}_{1\ldots n}]$, meaning that the i-th thought is represented as $\boldsymbol{c}_i \sim f_{\boldsymbol{\theta}}\left(\boldsymbol{c}_i \mid Q, \boldsymbol{c}_{1\ldots i-1}, \boldsymbol{p}_{1\ldots i-1}, \boldsymbol{p}_i\right)$. The guidance of $\boldsymbol{p}_i$ reduce the randomness and uncertainty of LLMs in generating the thought.

To enable reasoning with plan guidance, a direct prompting approach, *PS*, performs sampling via $\boldsymbol{p}_{1\ldots n} \sim f_{\boldsymbol{\theta}}\left(\boldsymbol{p}_{1\ldots n} \mid \mathbf{I}\right)$, where $\mathbf{I}$ represents a customized prompt. Other approaches, such as Trajectories Synthesis, REACT Yao et al. (2022), AUTOACT Qiao et al. (2024b), and PlaSma Brahman et al. (2024), formulate the sample as that each reasoning step $\boldsymbol{c}_i$ condition on $\boldsymbol{p}_i \sim f_{\boldsymbol{\theta}}\left(\boldsymbol{p}_i \mid Q, \boldsymbol{c}_{1\ldots i-1}, \boldsymbol{p}_{1\ldots i-1}, \mathbf{I}\right)$, allowing the LLM to be fine-tuned to first plan then reason.

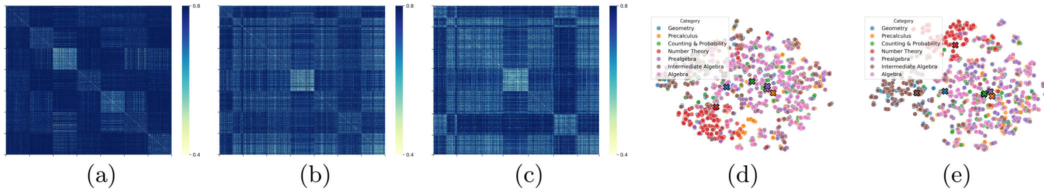


Figure 1: Illustration of relations between explicit plans of questions from different categories. We extract 200 samples from each category of the `MATH` dataset's 7 categories and prompt the LLM to decompose the answers into individual steps, followed by summarizing their explicit plans. (a)(b)(c) display the encoding distances between pairs of items: questions, explicit plans of Step 1 and Step 3, respectively. (d)(e) show the encoding clusters of the explicit plans of Step 1 and Step 3.

2.2 Explicit Plans present Commonality across Problems

Current efforts such as ReACT Yao et al. (2022) and PlaSma Brahman et al. (2024) that rely on explicit plans face a key limitation: specific and question-oriented instructions often fail to generalize well across diverse problems, and the detailed content involved is prone to errors, making it difficult for the LLM to learn effectively. Here, we introduce the idea of mapping plans from language space to latent space to capture high-level, generalizable, and concise instructions that provide conceptual-level reasoning guidance. Unlike explicit plans tailored to individual problems, their encodings are not question-specific; instead, they offer general guidance applicable across a variety of contexts. As a result, despite differing textual formulations, the encodings retain only the commonality that generalizes well across problems.

To show these benefits, we prompt DeepSeek-V3 Team (2024) to perform $\boldsymbol{c}_{1\ldots n} \sim f_{\boldsymbol{\theta}}\left(\boldsymbol{c}_{1\ldots n} \mid Q, \boldsymbol{A}, \mathbf{D}\right)$ and $\boldsymbol{p}_{1\ldots n} \sim f_{\boldsymbol{\theta}}\left(\boldsymbol{p}_{1\ldots n} \mid Q, \boldsymbol{c}_{1\ldots n}, \mathbf{S}\right)$, where $\boldsymbol{A}$ denotes the CoT answer, while $\mathbf{D}$ and $\mathbf{S}$ represent the prompts for answer decomposition and explicit plan summarization. We present the results in Figure 1. Specifically, after removing stop words from the questions and the summarized skeleton plans using NLTK, we encode them with the *all-MiniLM-L6-v2* model. We then visualize the pairwise distances with a heatmap and project the embeddings into 2D using t-SNE.

Figure 1 shows that embeddings of explicit plans exhibit a certain level of commonality and are capable of generalizing across problems. Using Figure 1a, which visualizes question similarity across seven categories, as a reference, we observe that explicit plans from reasoning Step 1 and Step 3 (Figure 1b) reveal two key trends: (1) when two questions are similar, their corresponding plans also tend to be close in embedding space; and (2) as the reasoning step increases, the degree of commonality strengthens. These trends are further supported by the clustering patterns of plans from Step 1 and Step 3 shown in Figures 1d and 1e: plans across different categories not only exhibit clear boundary separation but also significant overlap, indicating that a single plan can be applicable to questions both within and across categories.

3 Methodology

This section introduces the three components of our framework, *iCLP*: distilling explicit plans from existing answers, learning a latent plan space, and finetuning LLMs with latent plans (LPs). The overall pipeline is illustrated in Figure 2. Our core objective, motivated by Subsection 2.2, is to capture the commonalities and generalizable reasoning guidance beyond plans to support accurate reasoning across diverse tasks.

3.1 Explicit Plan Distillation

iCLP distills explicit plans, denoted as $\boldsymbol{p}_{1\ldots n}$, from existing CoT answers by prompting an off-the-shelf LLM. Specifically, for each question and its corresponding generated answer, as described in Subsection 2.2, *iCLP* first decomposes the answer into n separate reasoning

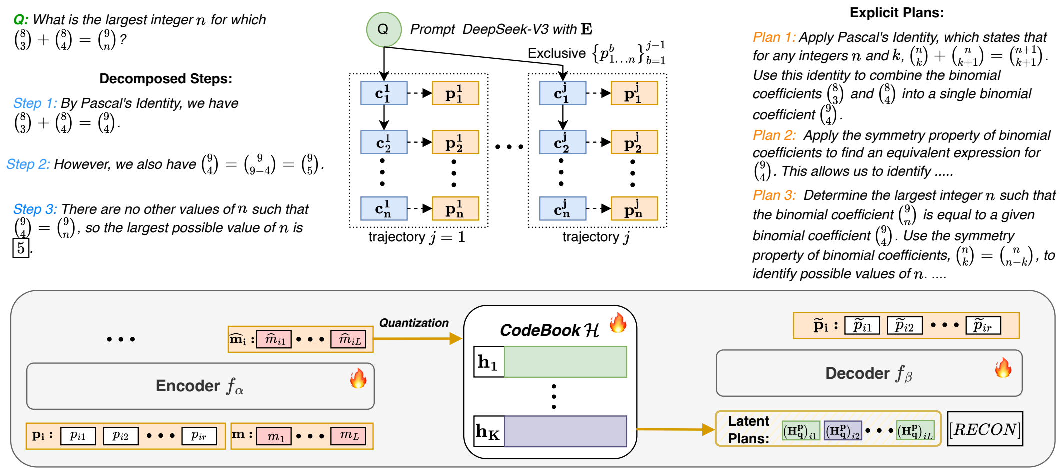


Figure 2: Illustration of the overall pipeline of *iCLP*. The upper part shows the process of Plan Distillation using a sample from the Counting & Probability category of the MATH dataset. The right part depicts the encoder - quantizer - decoder structure used for Latent Plan Generation.

steps, marking the i-th step with the phrase “Step i”. It then summarizes a plan for each step using zero temperature decoding, resulting in n plans $\boldsymbol{p}_{1\ldots n}$, one for each step.

To further increase the diversity of plans, in addition to using the provided answer for each sample, we prompt the LLM to generate U CoT reasoning trajectories per question. To prevent the LLM from reusing existing plans, for any reasoning trajectory indexed by $j \in [2, \ldots, U]$, we use a prompting formulation given by $\boldsymbol{c}_{1\ldots n} \sim f_{\boldsymbol{\theta}}\left(\boldsymbol{c}_{1\ldots n} \mid Q, \mathbf{E}, \{\boldsymbol{p}^b_{1\ldots n}\}_{b=1}^{j-1}\right)$. Here, $\mathbf{E}$ is a textual prompt that instructs the LLM not to reuse any previously used chains of plans $\{\boldsymbol{p}^b_{1\ldots n}\}_{b=1}^{j-1}$ during reasoning. Importantly, $\mathbf{E}$ permits the repetition of individual plans but not of entire chains $\boldsymbol{p}^b_{1\ldots n}$ because one chain can encode specific reasoning logic. After filtering out CoT trajectories that lead to incorrect solutions, we retain $\boldsymbol{p}^{U'}_{1\ldots n}$ for each question. Note that while the value of n varies across questions, we use fixed notation here for simplicity.

3.2 Latent Plan Generation

Although explicit plans present clear instructions and generalize among similar questions, they remain susceptible to token-level errors, particularly when LLMs hallucinate. To address this, we propose learning a latent plan space, which serves as a hidden representation of the plans. By conducting planning in this latent space, we mitigate the negative impact of token-level errors and further enhance the LLM's generalization ability, as the reasoning no longer depends on explicit and specific textual instructions.

To construct this latent space, we first follow the ICAE Ge et al. (2024) to introduce few memory tokens, denoted as $\boldsymbol{m} = [m_1, \ldots, m_L]$, where L is the length. Then, we introduce a vector-quantized autoencoder comprising an encoder $f_{\boldsymbol{\alpha}}$, a discrete codebook $\mathcal{H} = \{\boldsymbol{h}\}_{k=1}^{K} \subset \mathbb{R}^{d_h}$ and a decoder $f_{\boldsymbol{\beta}}$, where d_h represents the dimensionality of the code. Similar to the pipeline in VQ-VAE van den Oord et al. (2017) and VQ-GAN Esser et al. (2021), we approximate a given plan $\boldsymbol{p}$ by $\widetilde{\boldsymbol{p}} = f_{\boldsymbol{\beta}}\left(\boldsymbol{H}^{\boldsymbol{p}}_{\boldsymbol{q}}\right)$, where $\boldsymbol{H}^{\boldsymbol{p}}_{\boldsymbol{q}} \in \mathbb{R}^{d_h \times |\boldsymbol{p}|}$ is a spatial collection of codebook entries representing $\boldsymbol{p}$. Specifically, we set the $f_{\boldsymbol{\alpha}}$ to be an encoder-only transformer and $f_{\boldsymbol{\beta}}$ to be a decoder-only transformer. Thus, we obtain the $\boldsymbol{H}^{\boldsymbol{p}}_{\boldsymbol{q}}$ by performing element-wise quantization $\boldsymbol{q}\left(\widehat{\boldsymbol{H}^{\boldsymbol{p}}}\right)$, where each spatial code $\widehat{\boldsymbol{h}}_L \in \mathbb{R}^{d_h}$ from only the encoded memory representation $\widehat{\boldsymbol{H}^{\boldsymbol{p}}} = f_{\boldsymbol{\alpha}}\left([\boldsymbol{p}, \boldsymbol{m}]\right) \in \mathbb{R}^{d_h \times |\boldsymbol{m}|}$ is mapped to its nearest codebook entry $\boldsymbol{h}_k$, where $e \in [1, \ldots, |\boldsymbol{m}|]$ here is the token index of the memory token. This process is formulated as follows:

$$\boldsymbol{H}^{\boldsymbol{p}}_{\boldsymbol{q}} = \boldsymbol{q}\left(\widehat{\boldsymbol{H}^{\boldsymbol{p}}}\right) := \underset{\boldsymbol{h}_k \in \mathcal{H}}{\arg\min} \, ||\boldsymbol{h}_k - \widehat{\boldsymbol{h}}_e||, \quad \widetilde{\boldsymbol{p}} = f_{\boldsymbol{\beta}}\left(\boldsymbol{q}\left(f_{\boldsymbol{\alpha}}\left([\boldsymbol{p}, \boldsymbol{m}]\right)\right)\right), \forall e \in [1, \ldots, |\boldsymbol{m}|]. \tag{1}$$

For the reconstruction task, where $\widetilde{\boldsymbol{p}} \approx \boldsymbol{p}$, we employ the completion loss, as that presented in ICAE Ge et al. (2024). Given the quantized representation $\boldsymbol{H}^{\boldsymbol{p}}_{\boldsymbol{q}}$ and a special reconstruction indicator token '$[RECON]$', the decoder predicts the input $\boldsymbol{p}$ via next-token prediction. To optimize $\boldsymbol{\alpha}, \mathcal{H}, \boldsymbol{\beta}$, we propagate gradients from the decoder to the encoder, avoiding the non-differentiable quantization. The combination of the cross-entropy loss and commitment loss is computed as: $-\frac{1}{T}\sum_{t=1}^{T} \log P_\theta(x_t \mid x_{<t}) + ||sg\left[f_{\boldsymbol{\alpha}}(\boldsymbol{p})\right] - \boldsymbol{H}^{\boldsymbol{p}}_{\boldsymbol{q}}||^2 + ||sg\left[\boldsymbol{H}^{\boldsymbol{p}}_{\boldsymbol{q}}\right] - f_{\boldsymbol{\alpha}}(\boldsymbol{p})||$. With the trained $\boldsymbol{\alpha}, \mathcal{H}$, we can obtain the **latent plan** for any plan by representing it with $\boldsymbol{H}^{\boldsymbol{p}}_{\boldsymbol{q}}$.

3.3 Fine-tuning LLMs for Latent Planning

We synthesize new samples to fine-tune the LLM for planning in the latent space to enhance reasoning in the language space. Specifically, for a sample $q, \boldsymbol{c}_{1\ldots n}$ with distilled plans $\boldsymbol{p}_{1\ldots n}$, we first compute each plan encoding as $f_{\boldsymbol{\alpha}}([\boldsymbol{p}_i, \boldsymbol{m}])$, then map each memory token encoding to its closest codebook entry in $\mathcal{H}$. Thus, we have the matching indexes denoted as $\mathcal{I}_{\boldsymbol{p}} = \{\mathcal{I}_{\boldsymbol{p}_i}\}_{i=1}^{n}$, where for each $\boldsymbol{p}_i$, $\mathcal{I}_{\boldsymbol{m}_{ie}} = k$ corresponds to $\left(\boldsymbol{H}^{\boldsymbol{p}}_{\boldsymbol{q}}\right)_{ie} = \boldsymbol{h}_k$.

For any LLM, we extend the vocabulary size to include the size of the codebook K by adding special tokens denoted as '$[LP\{idx\}]$' where {idx} corresponds to the row index in the codebook. Consequently, we reformulate the sample into the form (*user*: question, *assistant*: latent plans and CoTs). This is obtained by first expressing the sample as (*user*: question, *assistant*: $\boldsymbol{p}_1, \boldsymbol{c}_1, \ldots, \boldsymbol{p}_n, \boldsymbol{c}_n$) following by replacing the token IDs of each $\boldsymbol{p}_i$ with $\mathcal{I}_{\boldsymbol{m}_i}$. As a result, with supervised fine-tuning, we train the extended vocabulary embeddings and fine-tune the LLM using the completion loss.

4 Experiments

Datasets. For the mathematical task, we use the `MATH` Hendrycks et al. (2021) and `GSM8K` Cobbe et al. (2021) datasets for model fine-tuning, while the `AIME2024` MAA Committees and `MATH-500` Hendrycks et al. (2021) datasets are used exclusively for evaluation. For the code generation task, we use the `CodeContests` Li et al. (2022) (3760 training, 165 test) dataset for fine-tuning, and the `HumanEval` Chen et al. (2021) and `MBPP` Austin et al. (2021) datasets for evaluation.

Learning Settings. We use Qwen2.5 Yang et al. (2024) in different sizes including 0.5B, 3B, 7B as the base models for fine-tuning toward latent planning. For all cases, to generate new CoT answers or extract the explicit plans, we prompt DeepSeek-V3 with a zero temperature and 0.3 temperature, respectively. For supervised fine-tuning, the learning rates for the 0.5B, 0.6B, 1.7B, 3B, and 7B models are set to 2e-5, 2e-5, 1e-5, 8e-6, and 5e-6, respectively. We employ the AdamW optimizer with a cosine learning rate scheduler. For *iCLP*, the encoder is the all-MiniLM-L6-v1 model from sentence-transformers, while the decoder is Qwen2.5-3B with the extended vocabulary, as discussed in subsection 3.3. For the quantizer, the size of the codebook is 2048, the dimension is 512, and β is 0.3. The training of *iCLP* uses LoRA Hu et al. (2021), with a batch size of 16 for 2 epochs. For the Plan Distillation, we set the U to be 20. Throughout the experiment, we set the number of memory tokens to $L = 6$ due to that the explicit plan is generally a short sentence.

Baselines. In addition to comparisons with base LLMs, *iCLP* is evaluated against state-of-the-art (SOTA) fine-tuning (FT) methods, including Learn Planning and Reasoning Jiao et al. (2024), PS+/PS Wang et al. (2023b), ReAct Yao et al. (2022), PlaSma Brahman et al. (2024), and Coconut Hao et al. (2024). However, fully reproducing these methods is relatively infeasible due to their strong dependence on specific task settings. Instead, we implement their core ideas within the context of our work, which involves step-wise explicit plans. These approaches can be viewed as instances of fine-tuning (FT) large language models on plan-based reasoning samples synthesized from existing datasets. We refer to this setting as

Table 1: Evaluating the reasoning performance of a series of Qwen2.5 models with different methods. The abbreviates of the datasets `MATH`, `MATH-500`, `AIME2024`, `GSM8K`, `CodeContests`, and `HumanEval` are `M`, `M-500`, `AM`, `G8`, `CC`, `HE`, and `MBPP`, respectively. The × indicates that, under the corresponding fine-tuning method for LLMs, training fails to converge. The → indicates cross-dataset evaluation, as described in the 'Metrics' part of the experimental settings. Here, the base refers to LLMs using zero-shot CoT prompting.

Qwen2.5	Methods	*Normal Mode*			*Cross Mode*					
		M	G8	CC	M→AM	G8→AM	M→M-500	G8→M-500	CC→HE	CC→MBPP
0.5B	Base	19.5	41.6	1.2	0	0	15.8	15.8	30.5	39.3
	GRPO	**49.6**	**54.5**	×	0	0	**34.4**	17.9	×	×
	FT-E	23.1	43.8	5.5	0	0	18.8	15.8	32.9	43.6
	FT-S	31.1	48	8.5	0	0	27.2	18.4	36.6	47
	FT-LE	13.3	38.3	1.8	0	0	11.2	9.6	19	21.4
	iCLP	36.7	51.5	**11.5**	0	0	28.1	**20.8**	**39.6**	**51**
3B	Base	42.6	79.1	9	0	0	39.6	39.6	42.1	57.1
	GRPO	**68.5**	**86.6**	×	10	0	**60.8**	40.4	×	×
	FT-E	48.2	81.4	10.9	0	0	43	39.8	44.5	61.2
	FT-S	54.9	83.2	13.9	7.6	0	48.4	40	47	65.2
	FT-LE	×	×	12.1	×	0	×	×	44.5	62
	iCLP	60.1	85	**18.8**	**10**	0	55.4	**44.2**	**54**	**73**
7B	Base	49.8	85.4	15.8	3.3	3.3	42.4	42.4	53	74.9
	GRPO	**83.7**	**91.7**	19.2	20	3.3	**78.2**	45.6	53.2	76
	FT-E	57.4	87.3	19.4	10	3.3	52	43.8	56.5	75.6
	FT-S	65.8	88.9	22.4	16.7	3.3	58.4	44	59.5	78.9
	FT-LE	×	×	×	×	×	×	×	×	×
	iCLP	74.3	90.1	**26.7**	**20**	**3.3**	68.6	**46.2**	**66.5**	**86.9**

FT-explicit (FT-E), where each step in the synthesized data includes a corresponding explicit plan. We also introduce Coconut, which implements latent chain-of-thought reasoning. This approach is distinct compared to our latent planning method and is therefore referred to as FT-special (FT-S). Additionally, for ablation analysis, we apply *iCLP* to learn a latent representation of the explicit plans and fine-tune the model based on this latent space, denoted as the *FT-latent explicit* (*FT-LE*) setting. *iCLP* is the full version of our method, which fine-tunes LLMs using latent plans learned from explicit plans. In particular, we compare our method with the state-of-the-art (SOTA) approach GPRO Shao et al. (2024), which fine-tunes LLMs using reinforcement learning.

Metrics. We report the pass1 accuracy (in %) of the LLM evaluated in three modes: In the *normal mode*, the LLM with *iCLP* is evaluated directly on the test set from the same dataset as the train set. In the *cross mode*, the LLM with *iCLP* is evaluated on a test set from a dataset different from the one used for training, denoted as training dataset → test set. The *accumulation mode* involves collecting explicit plans from multiple datasets to train the LLM with *iCLP*, followed by evaluation on various datasets.

4.1 Main Results

Table 1 shows that LLMs with *iCLP* significantly enhance reasoning performance across three datasets, demonstrating the ability to perform **generalizable planning and achieve high accuracy** across diverse tasks. In the *normal mode* and the more challenging *cross mode*, *iCLP* consistently outperforms all baselines and closely matches the performance of the GRPO algorithm Shao et al. (2024). These results suggest that reasoning with latent planning enables LLMs to acquire reasoning abilities that are both reliable and generalize effectively across a wide range of problem-solving tasks.

Specifically, *iCLP* consistently outperforms Base in *normal mode*, with an average accuracy improvement of 11.5. Although GRPO shows better performance on `MATH` and `GSM8K`, with average gains of 8 at 0.3B, 5 at 3B, and 5.5 at 7B, *iCLP* achieves the strongest results on `CodeContests`, where GRPO fails to converge at 0.5B and 3B and falls behind by 7.5 at 7B. For math reasoning tasks on `MATH` and `GSM8K`, *iCLP* surpasses *FT-E*, *FT-S*, and *FT-LE* with average improvements of 9.4, 4.3, and 30.4 respectively. On `CodeContests`, the gains are 7.1 over *FT-E*, 4.1 over *FT-S*, and 18.4 over *FT-LE*. These results suggest that *iCLP* enables

LLMs of various sizes to perform latent-space planning as generalized guidance, resulting in substantial improvements in both mathematical reasoning and code generation.

As shown in Table 2, Our *iCLP* achieves high **token efficiency**, with an average token cost of 250.3 ± 110.9 on MATH. This cost is significantly lower than that of planning-based reasoning methods such as PS+ Wang et al. (2023b). . In particular, on the more challenging dataset TheoremQA, the token usage is only 200.7 ± 100.2 for Qwen2.5-0.5B and 370.2 ± 127.1 for Qwen2.5-7Bwhich is lower even than ZeroCoT. This efficiency arises from two factors: (1) the latent plan requires only 6 tokens, and (2) the plan provides clear guidance, enabling the LLM to avoid unnecessary exploration during reasoning.

Table 2: Evaluation of the average generation token cost across different methods on MATH and TheoremQA. We report both the average and standard deviation (mean ± std) of the total tokens used per question, including tokens used for prompting the LLMs and those generated by the models.

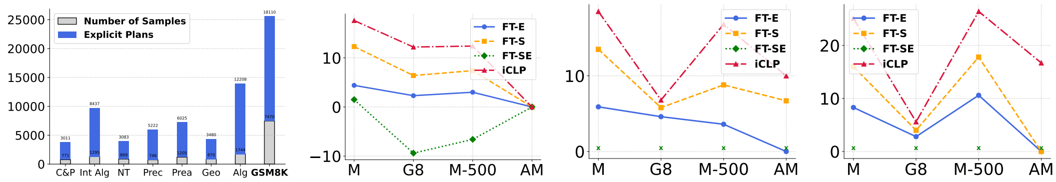
Methods	MATH	TheoremQA
0.5B w/ ZeroCoT	221.6 ± 172.5	250.3 ± 110.2
14B w/ ZeroCoT	261.8 ± 192.2	308.7 ± 137.5
0.5B w/ PS+	327.5 ± 176.7	367.5 ± 153.6
0.5B w/ Prompting	815.2 ± 356.7	993.4 ± 390.5
0.5B w/ *iCLP*	250.3 ± 110.9	**200.7 ± 100.2**
7B w/ ZeroCoT	246.9 ± 189.5	291.2 ± 160.8
7B w/ PS	357.2 ± 200.9	390.6 ± 190
7B w/ Prompting	934.5 ± 390.2	1103 ± 487.5
7B w/ *iCLP*	300.6 ± 150.8	**270.2 ± 127.1**

For the *cross mode* for generalizable planning evaluation, we evaluate Qwen2.5 models trained on latent plans derived from one dataset and tested on a different dataset. Qwen2.5 3B and 7B models with *iCLP* trained on MATH successfully solve challenging AIME2024 problems, with the 7B model achieving an accuracy of 20, only 3.3 lower than the much larger Qwen2.5-72B-Instruct. Across all model sizes (0.5B, 3B, 7B), *iCLP* significantly outperforms the Base LLM. While *iCLP* performs slightly below GRPO on M→M-500 with an average gap of 3.9, it consistently outperforms GRPO on G8→M-500, with average gains of 2.2, most notably 2.9 at 0.5B and 3.8 at 3B, indicating stronger generalization when transferring plans from G8 to M-500. In code generation, *iCLP* shows clear advantages. It improves over Base by an average of 12.4 on CC→HE and CC→MBPP. GRPO fails to converge at 0.5B and 3B on CC, while *iCLP* remains stable and effective. At 7B, where GRPO does converge, *iCLP* still outperforms it by 13.3 on CC→HE and 10.9 on CC→MBPP, demonstrating robust cross-data generalization. Moreover, compared to the variants *FT-E*, *FT-S*, and *FT-LE*, *iCLP* consistently achieves higher accuracy and more stable performance across model sizes and tasks, confirming its effectiveness in cross-data generalization for both math and code reasoning.

4.2 Continuous Learning with Latent Planning

Figure 3 demonstrates that in the *accumulation mode*, plans can be progressively accumulated across datasets to enhance the performance of *iCLP* in improving LLM reasoning, thus eavling the continuously learning ability of the LLMs with the latent plan space. Specifically, we first distill plans from the MATH and GSM8K datasets, then merge them and eliminate duplicates based on identical reasoning indices. Using this expanded set of plans, we train the codebook and fine-tune the LLM via *iCLP*. As shown in Figure 3, the resulting model is evaluated on four datasets.

Figure 3a highlights the significant difference between the number of explicit plans and the number of questions. In the seven categories of the MATH dataset, the number of explicit plans is not significantly larger than the number of questions, especially in some challenging categories such as *C&P*. Besides, LLMs fine-tuned with explicit plans achieve better accuracy, as shown in Table 1, indicating that explicit plans effectively capture the commonality and generalizable reasoning patterns across questions. Despite the small scale of explicit plans, such as 18,100 in GSM8K, learning latent plans results in their representation by only a few clusters. As shown in Figure 5, latent plans for questions within the same categories exhibit significant clustering while showing overlap across categories. More importantly, **fine-tuning**



(a) (b) (c) (d)

Figure 3: Illustration of the number of distilled plans and the *accumulation mode* performance of LLMs with *iCLP*. (a) shows the number of explicit plans that can be distilled from each category of the `MATH` and `GSM8K` datasets. The x-axis labels represent the abbreviations of the seven category names (see appendix). (b)(c)(d) show the accuracy gain ('y-axis') over the base model after fine-tuning the LLM with plans accumulated from the `MATH` and `GSM8K` datasets. Accuracy is measured across four datasets, with abbreviated names provided in Table 1. (b)(c)(d) correspond to Qwen2.5 models with 0.5B, 3B, and 7B parameters, respectively.

LLMs using these latent plans achieves optimal accuracy, as show in Table 1, which approaches SOTA RL-based method GRPO.

Thus, by accumulating plans, we are able to integrate **generalizable planning knowledge from different datasets**, enabling LLMs trained with *iCLP* to achieve better performance, as shown in Figure 3b, 3c, and 3d. Across all four evaluation datasets, Qwen2.5 models fine-tuned with latent ones (i.e., *iCLP*) show significant accuracy improvements compared to the base models. These improvements consistently exceed those achieved using explicit plans and *FT-S*. As the Qwen2.5 model size increases from 0.5B to 7B, the performance gains from *iCLP* become more evident, particularly with improvements greater than 20% on `MATH` and 10% on `AIME2024`.

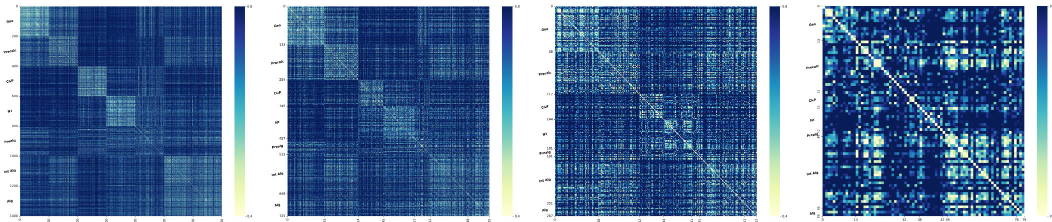
4.3 Ablation Study and Qualitative Results

Table 3 presents the impact of different settings for the encoding dimension and the size of the codebook $\mathcal{H}$ on the performance of our *iCLP*. Increasing d_h from 256 to 512 leads to a significant improvement in accuracy, while further increasing it to 1024 offers no additional benefit. Based on this, we set $d_h = 512$ and increase the codebook size K from from 1024 to 2048, which yields accuracy gains of 2.9 and 1.2 points on `MATH` and `CodeContests`, respectively. However, further enlarging K does not lead to additional performance improvements. Therefore, based on these results, especially the cross-data evaluation, we argue that the codebook size K plays a more critical role than the dimensionality in improving latent planning performance.

Table 3: Comparison of different dimensions (d_h and sizes (K)) of the coodbook $\mathcal{H}$.

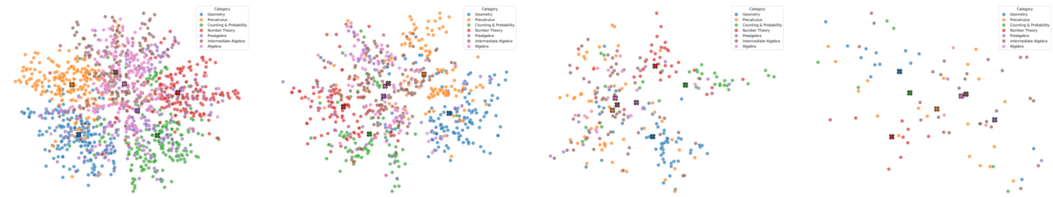
Qwen2.5	d_h	K	M	CC	M→AM	CC→HE
3B	256	1024	53.7	14.5	0	47.6
	512	1024	56.9	17	10	49.4
	1024	1024	57.2	17.6	10	51.2
	512	2048	60.1	18.8	10	54
	12	4096	60.1	19.4	10	54.3

To better understand and visualize the latent space of the latent plans, we perform reasoning with Qwen2.5-7B using *iCLP* on the `MATH` dataset and collect the encodings of latent plans across different reasoning steps. For each latent plan, we compute the average encodings by applying mean pooling over its token encodings. We then present the pairwise distance relationships between plans from different steps in Figure 4, and the corresponding encoding structures in 2D space using t-SNE in Figure 5. To make it easier to understand how the LLM with *iCLP* performs latent planning during reasoning, we provide a demonstration in Figure 6. It is evident that the LLM only needs to perform latent planning by generating a few special tokens before each reasoning step to achieve a reliable reasoning process. More importantly, compared to existing latent reasoning methods such as Coconut Hao et al. (2024), *iCLP* enables planning in the latent space while maintaining CoT reasoning in the



(a) (b) (c) (d)

Figure 4: Illustration of the relations of the encoding distances between pairwise latent plans from different reasoning steps (1, 2, 3, and 4). We randomly sample 200 questions from the test set of the `MATH` dataset and extract the encodings of latent plans generated by Qwen2.5-7B with *iCLP* during problem solving. Subfigures (a), (b), (c), and (d) present the results for the 1st, 2nd, 3rd, and 4th latent plans, respectively.



(a) (b) (c) (d)

Figure 5: Illustration of the encodings of latent plans from different reasoning steps (1, 2, 3, and 4) in 2D space. We follow the same procedure as in Figure 4 and visualize the latent plan encodings using t-SNE.

language space, thereby guaranteeing strong interpretability, which is crucial for practical applications.

From Figure 4, we observe two key patterns. First, for questions belonging to the same category in `MATH`, their latent plans exhibit clear similarity, as evidenced by the prominently highlighted diagonal regions in Figures 4a,4b, 4c, and 4d. This indicates that the LLM with *iCLP* tends to generate similar latent plans when solving problems within the same category. Second, in the later reasoning steps—particularly in Figures 4c and 4d—latent plans become more aligned across different categories, suggesting that the model increasingly draws on similar implicit plan knowledge regardless of the specific problem type. From Figure 5, we observe two key patterns. First, latent plans from the same category form distinct clusters, while those from different categories are clearly separated, indicating strong category-specific structuring in the latent space. Second, the clusters from different categories are distributed around a common center, suggesting that the latent plans share a core of common implicit knowledge that generalizes well across problem types. Therefore, the result matches our motivation in Subsection 2.2: **enabling the LLM to plan in the latent space with generalizable reasoning guidance that is reusable and transfers well across problems, thereby supporting both the generalization and accuracy of reasoning**.

5 Related Work

Large language models (LLMs) have the capability to solve problems using **step-by-step reasoning** Kojima et al. (2022), where each step addresses a sub-problem and is described textually as a thought, thereby collectively forming a Chain of Thought (CoT) Wei et al. (2022). However, inherent hallucinations in LLMs can lead to ineffective thoughts. To address this issue, prompting methods have been proposed to extend CoT reasoning by incorporating additional searching Wang et al. (2022); Zhou et al. (2023); Fu et al. (2023); Yao et al. (2023) or self-reflection mechanisms Sijia et al. (2024); Sijia & Baochun (2024); Miao et al. (2024).

Instead of relying on resource-intensive approaches, **enhancing reasoning with step-wise planning** aims to directly guide LLMs using a plan — a textual instruction that specifies

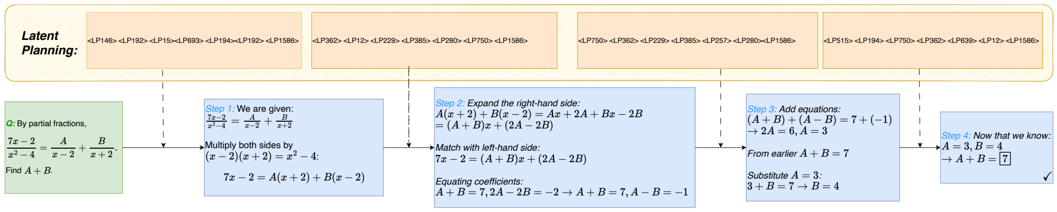


Figure 6: Illustration of the reasoning process of LLMs with latent planning. In each of the four reasoning steps, Qwen2.5-7B with *iCLP* first plans in the latent space, which then guides the generation of the next reasoning step.

what to do at each step for reliable reasoning Huang et al. (2024; 2022). Prior works Wang et al. (2023a); Sun et al. (2023); Ling et al. (2023) Prompt LLMs to generate question-specific plans or premises to guide step-wise reasoning. Building on this idea, subsequent works Yao et al. (2022); Lyu et al. (2024); Zheng et al. (2024); Zhao et al. (2023); Qiao et al. (2024a) propose to guide LLMs by synthesizing new reasoning processes in the form of planning trajectories, where each step begins with a plan and is followed by the generation of a corresponding solution step. However, as it remains difficult for LLMs to produce effective and coherent plans Valmeekam et al. (2023; 2024); Xie et al. (2025), recent methods Yao et al. (2022); Jiao et al. (2024); Qiao et al. (2024b); Brahman et al. (2024); Qiao et al. (2024a) fine-tune LLMs using synthesizes planning trajectories. However, the plans used in these approaches are still tightly coupled with specific questions and task contexts. As a result, they often lack generalizability and are susceptible to errors in detailed content. In contrast, our work focuses on latent plans, which provide high-level, concise, and generalizable instructions.

Reasoning in a latent continuous space has been increasingly explored in recent research Xu et al. (2024); Hao et al. (2024); Pagnoni et al. (2024); Su et al. (2025); team et al. (2024); Tack et al. (2025), demonstrating promising improvements in both accuracy and efficiency. Similar to our approach of learning latent plans for CoT rationales, LaRS Xu et al. (2024) proposes constructing a latent space of rationales via unsupervised learning, enabling LLMs to retrieve latent rationales for given questions. The TokenAssorted in Su et al. (2025) abstracts away initial reasoning steps using latent discrete tokens generated by VQ-VAE van den Oord et al. (2017). CoCoMix Tack et al. (2025), closely related to our work, combines discrete next-token prediction with continuous concept representations learned via a pretrained sparse autoencoder. Although the ideas presented in these methods are closely aligned with ours in blending latent representations with language tokens during reasoning, our work makes a significant advance by explicitly separating the planning and reasoning phases: planning is performed in a latent space, while reasoning is carried out in natural language.

6 Concluding Remarks

In this paper, we enabled large language models (LLMs) to emulate human-level intelligence, specifically *Implicit Cognition* (IC), by introducing a novel framework called *iCLP*. This framework allows LLMs to perform planning in a latent space in order to augment their reasoning capabilities in the language space. A central component of *iCLP* is the use of latent plans, which are crucial due to their ability to generalize across tasks. To realize this, we employ a vector quantizer autoencoder to learn a latent space, represented as a codebook, from distilled plans. We then fine-tune LLMs to generate latent plans that support reasoning across a variety of problem-solving tasks. Experimental results demonstrate that incorporating *iCLP* into LLMs leads to substantial improvements in reasoning accuracy. Furthermore, the generalizable structure of the latent plans enables LLMs fine-tuned on one dataset to transfer effectively to other datasets without requiring retraining. The latent space also supports continual learning, allowing plans distilled from multiple datasets to enhance the codebook and continuously strengthen the model's latent planning capabilities for guiding reliable reasoning. We hope this work opens a new direction that highlights the importance of enhancing reliable and generalizable reasoning through planning in a continuous latent space.

Reproducibility statement

To support reproducibility, the paper provides sufficient methodological and experimental details to enable independent replication of our results. The complete implementation, including all code, configuration files, and instructions necessary to reproduce the findings, is publicly available as a GitHub repository and the supplementary material. In keeping with current research best practices, all materials and necessary components are made publicly available on these places to enable full accessibility and reproducibility by the research community. The provided GitHub repository and our "code.zip" includes detailed instructions for reproducing not only the proposed method *iCLP* but also the main results reported in the paper. To access the source code directly, please locate the **examples/LatentPlan** directory in the supplementary material file "code.zip". Additionally, **please read the *README.md* file for step-by-step instructions on how to run the code**. The raw data of the distilled plans from `MATH` and other datasets are available on the *Hugging Face* website. Please download them directly.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *ArXiv*, abs/2108.07732, 2021. 5

Faeze Brahman, Chandra Bhagavatula, Valentina Pyatkin, Jena D. Hwang, Xiang Lorraine Li, Hirona Jacqueline Arai, Soumya Sanyal, Keisuke Sakaguchi, Xiang Ren, and Yejin Choi. Plasma: Procedural knowledge models for language-based planning and re-planning. In *International Conference on Learning Representations*, 2024. 1, 2, 3, 5, 10

Mark Chen et al. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021. URL `https://api.semanticscholar.org/CorpusID:235755472`. 5

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. 5

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proc. IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021. 2, 4

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *Proc. International Conference on Learning Representations*, 2023. 9

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *Proc. International Conference on Learning Representations*, 2024. 2, 4, 5

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *ArXiv*, 2024. 5, 8, 10

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. 5

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021. 5

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Annual Conference on Robot Learning*, 2022. 1, 10

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *ArXiv*, 2024. 1, 10

Fangkai Jiao, Chengwei Qin, Zhengyuan Liu, Nancy F. Chen, and Shafiq R. Joty. Learning planning-based reasoning by trajectories collection and process reward synthesizing. In *Conference on Empirical Methods in Natural Language Processing*, 2024. 1, 5, 10

J. F. Kihlstrom, V. A. Shames, and J. Dorfman. Intuition, incubation, and insight: Implicit cognition in problem-solving. In Geoffrey D. M. Underwood (ed.), *Implicit Cognition*, pp. 257–296. Oxford University Press, 1995. 1

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022. 9

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom, Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de, Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey, Cherepanov, James Molloy, Daniel Jaymin Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de, Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378:1092 – 1097, 2022. 5

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. In *Advances in Neural Information Processing Systems*, 2023. 10

Edwin A. Locke and Amy L. Kristof. Volitional choices in the goal achievement process. In Peter M. Gollwitzer and John A. Bargh (eds.), *The Psychology of Action: Linking Cognition and Motivation to Behavior*, pp. 365–384. Guilford, 1996. 1

Yuanjie Lyu, Zihan Niu, Zheyong Xie, Chao Zhang, Tong Xu, Yang Wang, and Enhong Chen. Retrieve-plan-generation: An iterative planning and answering framework for knowledge-intensive llm generation. In *Conference on Empirical Methods in Natural Language Processing*, 2024. 1, 10

MAA Committees. Aime problems and solutions. Art of Problem Solving. URL `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`. 5

Ning Miao, Yee Whye Teh, and Tom Rainforth. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. In *Proc. International Conference on Learning Representations*, 2024. 9

Joseph Murphy. *The Power of Your Subconscious Mind*. Prentice-Hall, 1963. 2

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke S. Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. Byte latent transformer: Patches scale better than tokens. *ArXiv*, 2024. 10

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. In *Advances in Neural Information Processing Systems*, 2024a. 10

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. Autoact: Automatic agent learning from scratch for qa via self-planning. In *Annual Meeting of the Association for Computational Linguistics*, 2024b. 1, 2, 10

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, 2024. 2, 6

Chen Sijia and Li Baochun. Toward adaptive reasoning in large language models with thought rollback. In *International Conference on Machine Learning*, 2024. 9

Chen Sijia, Li Baochun, and Di Niu. Boosting of thoughts: Trial-and-error problem solving with large language models. In *Proc. International Conference on Learning Representations*, 2024. 9

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*, 2025. 10

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in Neural Information Processing Systems*, 2023. 1, 10

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. Llm pretraining with continuous concepts. *ArXiv*, 2025. 10

DeepSeek-AI Team. Deepseek-v3 technical report. *ArXiv*, 2024. 3

The Lcm team, Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta Ruiz Costa-jussà, David Dale, Hady ElSahar, Kevin Heffernan, Joao Maria Janeiro, Tuan Tran, Christophe Ropers, Eduardo Sánchez, Robin San Roman, Alex Mourachko, Safiyyah Saleem, and Holger Schwenk. Large concept models: Language modeling in a sentence representation space. *ArXiv*, 2024. 10

Karthik Valmeekam, Sarath Sreedharan, Matthew Marquez, Alberto Olmo Hernandez, and Subbarao Kambhampati. On the planning abilities of large language models (a critical investigation with a proposed benchmark). In *Advances in neural information processing systems*, 2023. 1, 10

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. In *International Conference on Learning Representations*, 2024. 1, 10

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Neural Information Processing Systems*, 2017. 4, 10

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka Wei Lee, and Ee Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Association for Computational Linguistics*, pp. 2609–2634, 2023a. 1, 10

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proc. Annual Meeting of the Association for Computational Linguistics*, volume 1, pp. 2609–2634, 2023b. 5, 7

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. In *Proc. International Conference on Learning Representations*, 2022. 9

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. 1, 9

Jian Xie, Kexun Zhang, Jiangjie Chen, Siyu Yuan, Kai Zhang, Yikai Zhang, Lei Li, and Yanghua Xiao. Revealing the barriers of language agents in planning. In *Association for Computational Linguistics*, 2025. 10

Zifan Xu, Haozhu Wang, Dmitriy Bespalov, Xian Carrie Wu, Peter Stone, and Yanjun Qi. Lars: Latent reasoning skills for chain-of-thought reasoning. In *Conference on Empirical Methods in Natural Language Processing*, 2024. 10

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, 2024. 5

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2022. 1, 2, 3, 5, 10

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023. 9

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. 10

Zhonghua Zheng, Lizi Liao, Yang Deng, Ee-Peng Lim, Minlie Huang, and Liqiang Nie. Thoughts to target: Enhance planning for target-driven conversation. In *Conference on Empirical Methods in Natural Language Processing*, 2024. 10

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *Proc. International Conference on Learning Representations*, 2023. 9

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Ningyu Zhang, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, and Huajun Chen. Knowagent: Knowledge-augmented planning for llm-based agents. In *North American Chapter of the Association for Computational Linguistics*, 2024. 1