Skip-Thinking: Chunk-wise Chain-of-Thought Distillation Enable Smaller Language Models to Reason Better and Faster

Anonymous ACL submission

Abstract

Chain-of-thought (CoT) distillation allows a large language model (LLM) to guide a small language model (SLM) in reasoning tasks. Existing methods train the SLM to learn the long rationale in one iteration, resulting in two issues: 1) Long rationales lead to a large token-level batch size during training, resulting in over-smoothed gradients of core reasoning tokens (i.e., the token will directly affect the correctness of subsequent reasoning) as they contribute a tiny fraction of the rationale. As a result, the SLM converges to sharp minima where it fails to grasp the reasoning logic. 2) The answer response is slow, as the SLM must generate a long rationale before reaching the answer. Therefore, we propose chunk-wise training (CWT), which uses a heuristic search based on the SLM loss to divide the rationale into semantically coherent chunks and focuses the SLM on learning from one chunk per iteration. Since each chunk contains fewer tokens, the gradients of core reasoning tokens in the chunk receive greater weight during backpropagation. On the basis of CWT, skip-thinking training (STT) is proposed. STT makes the SLM skip several medium reasoning chunks to reach the answer, improving reasoning speed while maintaining accuracy. We validate our approach on a variety of SLMs and multiple reasoning tasks.[1]

1 Introduction

Chain of Thought (CoT) (Chu et al., 2023) distillation enables small language models (SLMs) (Radford et al., 2019; Raffel et al., 2020) to replicate the reasoning patterns of large language models (LLMs) (Ouyang et al., 2022; Touvron et al., 2023; Dubey et al., 2024), enhancing their reasoning abilities for domain-specific tasks. The training procedure for mainstream CoT distillation methods (Ho et al., 2023; Magister et al., 2022; Ren and Zhu, 2022) is shown in the top box of Figure 1. It requires the SLM to learn a long reasoning process (rationale) from the LLM for a given task in a single training iteration, which leads to two problems.

1) Superficial understanding. The training loss for the SLM is computed as the average value over all target tokens. Consequently, the token-level batch size corresponds to the number of training tokens within a mini-batch. Since the rationale is long, the token-level batch size remains large even with a batch size of 1. Large batch size typically causes gradient over-smoothing during backpropagation (Jastrzębski et al., 2018; Keskar et al., 2017; Gao and Zhong, 2020), thereby leading to a generalization gap. Specifically, the model updates with the average gradient of the tokens in the batch. As the batch size increases gradually—consider an extreme case where a single batch encompasses the entire training dataset—the gradients across batches become more similar, causing the model loss to decrease rapidly along the similar gradients and converge to a sharp minimum. More critically, in CoT distillation, the core reasoning tokens (such as the color of the ball in Figure 1) constitute a small proportion of rationales while the prevalence of similar non-reasoning tokens (e.g., those used for foreshadowing and summarization) across different rationales, which exacerbates gradient over-smoothing, leading the SLM to converge rapidly towards learning the expressive patterns of the LLM rather than core reasoning logic.

2) Time-consuming. The SLM trained with these methods requires completing the full reasoning process to produce the final answer during testing, which results in a significantly slower answer response time.

To address the first problem, some naive approaches, such as weighting the loss of core reasoning tokens or prompt LLMs to remove redundant expressions in rationale, do not perform well (see

[1]The code will be available after the paper is officially accepted.

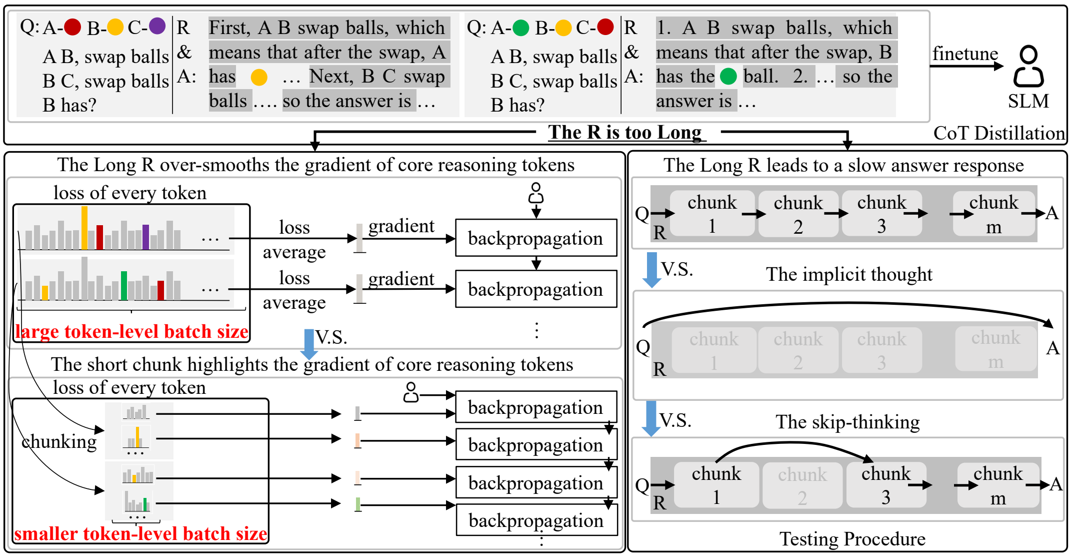

Figure 1: Illustration of CoT Distillation. The batch size is set to 1 as an illustrative example. The core reasoning token (like the color of ball in rationale R) means that its accuracy can determine the subsequent reasoning process. 1) *Superficial understanding*: The large token-level batch size will cause the gradient of the core reasoning token to be over-smoothed by plenty of other non-reasoning tokens (highlighted with a gray background in R) that are similar across different rationales during backpropagation, leading to SLMs converging to a sharp minimum where SLM often makes mistakes when generating the core reasoning token. 2) *Time-consuming*: Generating the full R takes longer than outputting the answer A directly.

Appendix B). In this work, we propose a chunk-wise training (CWT) strategy. CWT utilizes a chunking data generator to segment the rationale into a fixed number of reasoning chunks and focus SLMs on learning from one chunk per iteration. In this way, first, the token-level batch size is smaller, which mitigates gradient over-smoothing. More importantly, the relative contribution of each core reasoning token within a chunk to the gradient increases, thus driving the SLM to optimize its understanding of the core logic. In addition, since separating semantically coherent reasoning steps into independent chunks potentially disrupts the coherence of SLM reasoning, CWT introduces a heuristic search guided by model loss to optimize chunk segmentation.

For the second question, several methods (Hsieh et al., 2023; Chen et al., 2024b; Deng et al., 2023, 2024) have been proposed to enhance the response speed of the answer. Among them, internalizing the explicit reasoning process (Deng et al., 2023, 2024) into the latent space has emerged as a promising direction. However, these internalization-based methods may compromise answer accuracy due to the lack of an explicit reasoning process. Similar to these approaches, we hypothesize that language models can also encode explicit reasoning within the latent space. However, we argue that, akin to humans who externalize parts of their reasoning to maintain coherence and mitigate forgetting, language models should periodically externalize portions of their thought processes to facilitate subsequent reasoning. Therefore, we propose a CWT-based skip-thinking training (STT) strategy. Specifically, the model is trained to infer the current chunk's content from partially omitted preceding reasoning chunks, allowing it to skip certain explicit reasoning chunks while still reaching the correct conclusion. To further balance reasoning speed and accuracy, we propose a skip self-consistency (SSC) enhancement strategy during the testing phase. This strategy aggregates skip-thinking results across different spans to determine the final answer, effectively enhancing the accuracy of skip-thinking.

The key contributions are as follows:

1) To prevent a superficial understanding of SLM to the LLM's reasoning logic, we provide a theoretical analysis from the perspective of gradient updates and propose the CWT to enhance SLMs' capability in reasoning logic comprehension.

2) A skip-thinking training approach is proposed based on reasoning internalization, which not only preserves reasoning accuracy but also accelerates SLM inference.

3) Plenty of experiments are conducted across 2 different SLMs and 7 reasoning tasks to verify our proposed method.

2 Related works

CoT (Chu et al., 2023) is first introduced by Wei et al. (2022). Subsequently, CoT distillation and reasoning acceleration emerges as two critical research directions aimed at broadening the application scope of CoT.

2.1 CoT distillation

CoT distillation is first introduced in concurrent works by Ho et al. (2023), Magister et al. (2022), and Ren and Zhu (2022). They prompt the LLM to generate rationales for a given task, which is then applied as the supervised label to make the SLM mimic the reasoning logic of the LLM. Building upon these works, Scott (Wang et al., 2023) is introduced to enhance the alignment of the SLM's rationale with the answer. Li et al. (2023) proposes integrating the LoRA (Hu et al., 2022) to enhance the utilization of negative samples generated by the LLM. PaD (Zhu et al., 2023) employs an external code compiler to enhance the performance of the SLM. In addition to the aforementioned work on improving the distillation mechanism, some works have integrated CoT distillation with information retrieval (Zhao et al., 2024), table reasoning (Yang et al., 2024), thereby broadening the application scope of CoT distillation.

However, the aforementioned methods enable the SLM to learn the full rationale for the given task in a single iteration, which may cause the SLM to superficially understand the reasoning logic of LLMs.

2.2 CoT acceleration

The existing methods to accelerate the reasoning process can be roughly divided into three directions: multi-task learning, post-thinking mechanism, and latent space thought.

Multi-task learning (Hsieh et al., 2023; Chen et al., 2024b; Liu et al., 2024) utilizes distinct prefixes to differentiate between tasks. For instance, when the input task prefix is *[label]*, the SLM directly outputs the answer, whereas when the input task prefix is *[rationale]*, the SLM outputs the rationale. Since multi-task learning allows for outputting the answer directly, the answer response time can align with that of the standard fine-tuning that only applies the answer to train SLM. However, because the rationale and the answer are not within the same output sequence, the conclusion of the SLM's rationale often fails to align with the answer directly output by the SLM.

Post-thinking mechanism (Chen et al., 2024a) trains the SLM to output the rationale after providing the answer, so that the answer can be generated first during the test. However, the post-thinking sacrifices the ability to decompose the task through the rationale, making it more challenging to handle tasks with higher complexity.

Training SLMs to reason in latent space has emerged as a recent research direction (Deng et al., 2023; Goyal et al., 2024; Deng et al., 2024; Hao et al., 2024). These methods propose internalizing explicit rationales into latent space, enabling implicit reasoning during forward propagation to directly generate answers. For instance, Deng et al. (2024) gradually removes reasoning steps during training to internalize rationales, while Hao et al. (2024) introduces a special token, *[thought]*, to facilitate latent reasoning. However, this approach may reduce answer accuracy in some tasks compared to explicit reasoning. We posit that this stems from the model's tendency to forget previous reasoning steps during extended reasoning in the latent space. Explicit rationales serve as a scratchpad that facilitates problem-solving (Wei et al., 2022). When discarded, the model is more likely to forget prior steps, leading to degraded reasoning capacity.

3 Preliminary

Let $D = \{(q_i, a_i)|i = 0, 1, ..., n\}$ refer to the original dataset consisting of n samples for training SLM, where q_i and a_i represent the question and answer, respectively. Based on D, CoT distillation first utilizes a zero-shot or few-shot CoT prompt to make LLM output rationale r_i for q_i. Then, the SLM is trained to maximize the generation likelihood of r_i and a_i. The training loss per training iteration of CoT distillation can be formulated as:

$$\mathbb{L} = \frac{1}{B}\sum_{b=0}^{B}\frac{1}{K-s}\sum_{k=s}^{K}\ell(f_\vartheta(x^b_{1,k},), x^b_{k+1}) \quad (1)$$

where $x = q \oplus r \oplus a$ is the input sequence whose length is K ($\oplus$ refers to the string concatenation), B refers to the training batch size, s is the start index of $r \oplus a$ in x, $f_\vartheta(\cdot)$ represents the forward calculation of SLM with parameters ϑ, and $\ell(\cdot)$ is the cross-entropy loss. After training, SLM has the ability to think before outputting answers.

Superficial understanding. Considering the parameters of SLM as a whole, during backpropa-

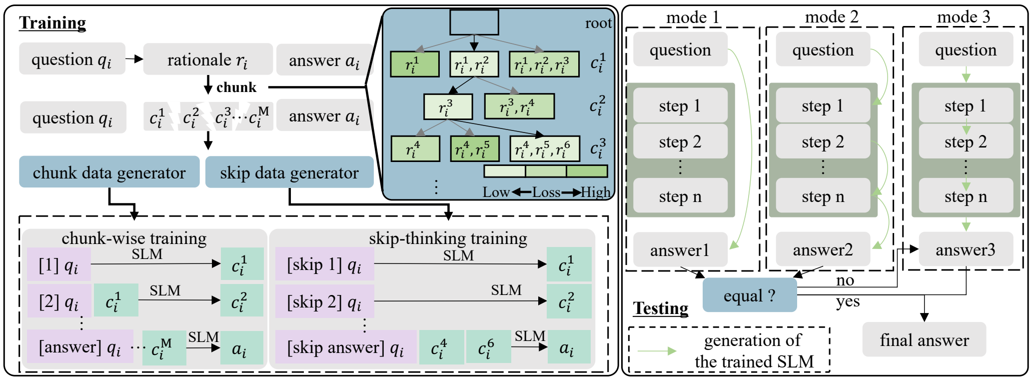

Figure 2: The illustration of the proposed methods.

gation, the gradient of ϑ can be expressed as:

$$\frac{\partial \mathbb{L}}{\partial \vartheta} = \frac{\partial}{\partial \vartheta}\left(\frac{1}{N}\sum_{i=1}^{N}\ell_i\right) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \ell_i}{\partial \vartheta} \tag{2}$$

where $N = B \times (K - s)$ represents the token-level batch size and ℓ_i is the cross-entropy loss for i_{th} training token. Assume that we divide the training tokens into two sets S_1 and S_2, where S_1 is the training token set involving the core logic in any single reasoning step and S_2 is the remaining tokens, Equation 2 can be rewritten as:

$$\frac{\partial \mathbb{L}}{\partial \vartheta} = \frac{1}{N}\sum_{i=1}^{|S_1|}\frac{\partial \ell_i}{\partial \vartheta} + \frac{1}{N}\sum_{j=1}^{|S_2|}\frac{\partial \ell_j}{\partial \vartheta} \tag{3}$$

Since $|S_2|$ is usually much larger than $|S_1|$, the gradient of the token in S_1 will be smoothed by the gradient of the token in S_2, which ultimately leads to a superficial understanding of SLM in this reasoning step.

Slow answer response. The SLM trained according to Equation 1 must first generate a rationale before providing an answer, which leads to a slower answer response compared to the SLM that directly outputs the answer.

4 Method

To address the two problems, we propose CWT and STT. Figure 2 illustrates the process. First, the LLM generates r_i for q_i. Details on obtaining r_i are in Appendix A.2. The chunk and skip data generators then sequentially generate data for CWT and STT. Finally, we introduce the Skip Self-Consistency enhancement (SSC) to balance response speed and accuracy.

4.1 Chunk data generator

The chunk data generator divides the complete rationale into several smaller parts and learn from each part independently during a single training iteration. After the division, $|S2|$ in Equation 3 is significantly reduced, allowing the SLM to concentrate on comprehending the essential reasoning logic within the given part.

Division methods vary in granularity: sentence-level, reasoning step-level, and chunk-level. The first two methods lead to duplicate generation due to task-specific variations in sentence and reasoning step numbers (see Appendix C). Chunk-level division segments the rationale into M chunks. Training SLM with this data will make the SLM reach the answer after M distinct stages, thereby avoiding duplicate generation. Thus, the chunk-level division is employed in the chunk data generator.

4.1.1 Average chunking

When performing chunking, the simplest way is to divide the reasoning steps into M parts equally. Specifically, we first split the rationale by $''\backslash n''$ to obtain $r_i = \{r_i^j | j = 0, 1, ..L\}$ that has L reasoning steps. Then the reasoning steps contained in the m_{th} chunk can be formulated as:

$$c_i^m = \begin{cases} \{r_i^j \mid j \in [g \times m, g \times m + g))\} & m < M \\ \{r_i^j \mid j \in [g \times m, L]\} & m = M \end{cases} \tag{4}$$

where $g = \lfloor L/M \rfloor$ and $j \in \mathbb{Z}$. After chunking, we can convert a training sample x_i into $M+1$ training data. The first M training data can be formalized as:

$$\{[m] \oplus q_i \oplus c_i^1 \oplus c_i^2 ... \oplus c_i^m | m = 0, 1, ..., M\} \tag{5}$$

and the $M + 1$ training sample is:

$$[answer] \oplus q_i \oplus c_i^1 \oplus c_i^2 ... \oplus c_i^M \oplus a_i \tag{6}$$

The reason for adding the prefix $[m]$ and $[answer]$ is that it can tell the model what stage the current

Algorithm 1 Search-based chunking

Input: Chunk list c_{ij} of r_i and SLM ϑ_j before $(j+1)$ training epoch, q_i, a_i, threshold η, M

for m in range($M-1$) **do**
  Calculate the loss l_c for c_{ij}^m with Equation 1
  Merge c_{ij}^m and c_{ij}^{m+1} to form the list c_{temp}
  Initial: $l_{min} \leftarrow +\infty$, $index \leftarrow +\infty$
  for idx in range(len(c_{temp})) **do**
    Calculate the loss l_{idx} for $c_{temp}[:idx]$ with Equation 1
    if $l_{idx} < l_{min}$ **then**
      $l_{min} \leftarrow l_{idx}$, $index \leftarrow idx$
    end if
  end for
  if $l_c - l_{min} > \eta$ **then**
    $c_{ij}^m \leftarrow c_{temp}[:index]$
    $c_{ij}^{m+1} \leftarrow c_{temp}[index:]$
  end if
end for

Output: Chunk list $c_{i(j+1)}$ of r_i

reasoning is at, thereby reducing the difficulty of reasoning. And the s in the Equation 1 is the start index of c_i^m and a_i in these data at this time.

4.1.2 Search-based chunking

Since the average chunking (AC) may divide multiple semantically coherent reasoning steps into different chunks, the reasoning fluency of the SLM may degrade after training. In addition, the combinatorial space for allocating L reasoning steps to M chunks is vast. Therefore, we propose a search-based chunking (SBC) that applies the loss of SLM as heuristic information to efficiently identify a better chunking result.

The detailed process of SBC is outlined in Algorithm 1. The initial chunking result c_i^0 is obtained through AC. Algorithm 1 is executed before each training epoch. In general, the loss of the language model on the target token sequence indicates the language model's understanding of the content within the target token sequence (Wan et al., 2024). Based on this point, in Algorithm 1, we progressively increase the number of reasoning steps allocated to the current searching chunk and compute the SLM loss for it. As the loss decreases, we infer that the reasoning steps allocated to this chunk are more comprehensible to the SLM, aiding its understanding of the information in the current reasoning stage. Thus, we utilize this loss comparison as heuristic information to iteratively adjust the chunk division with a greedy strategy, reducing suboptimal results from unreasonable division during training.

4.2 Skip data generator

To accelerate reasoning, we employ a skip data generator for STT. STT is essentially to internalize the rationale. However, unlike Deng et al. (2024), STT still requires the explicit output of the SLM to provide a clear intermediate basis for subsequent reasoning.

Specifically, the skip data generator randomly removes one or more chunks preceding the target sequences c_i^m and a_i in Formula 5 and 6, respectively. Subsequently, it applies the corresponding prefixes $[skip\ m]$ and $[skip\ answer]$ to the corrupt sequences, explicitly instructing the SLM to skip some chunks to the chunk c_i^m and to output the answer based on incomplete explicit rationale respectively. The skip data generator will process every sample generated by the chunk data generator, and the newly generated data are incorporated with the original data to jointly train the SLM. Training the SLM with such data compels it to rely on the information from the unremoved chunks as the basis for reasoning, directly skipping the removed intermediate reasoning steps and outputting the target chunk. In other words, the removed reasoning steps are internalized within the latent space for reasoning (Deng et al., 2024; Hao et al., 2024).

4.3 Skipping self-consistency enhancement

After training the SLM with the above data, three distinct reasoning modes emerge:

1) Mode 1 involves directly skipping the all rationale to obtain the answer. The input for SLM is $[skip\ answer] \oplus q_i$

2) Mode 2 refers to skipping the middle part of the rationale. The initial input $[skip\ m] \oplus q_i$ is fed into SLM to obtain the c_i^m without outputting the middle thought. Then a new input $[skip\ d] \oplus q_i \oplus c_i^m$ $(m < d \leq M)$ is input into SLM to get the c_i^d. This process is iterated until c_i^M is output or we decide to output the answer. At this point, we feed $[skip\ answer] \oplus q_i \oplus c_i^m \oplus c_i^d ...$ into the model to get the final Answer. Obviously, mode 2 requires manual control of the skipping process. In our work, we skip only a chunk each time until the final answer is obtained.

3) Mode 3 is full thinking, which means explicitly outputting full rationales before getting the

answer. The initial input is $[m] \oplus q_i$. Similar to mode 2, we will iteratively obtain the final answer.

Among the three modes, mode 3 achieves the strongest overall performance, while mode 1 exhibits the fastest reasoning speed. To combine these three reasoning modes, we propose the SSC that achieves a more balanced trade-off between speed and performance. In SSC, the three modes reason in parallel. When the answers obtained by mode 1 and mode 2 are consistent, the corresponding answer is taken as the final result, and all reasoning processes are terminated. Otherwise, the result of mode 3 is taken as the final answer.

5 Experiments

We first introduce the detailed experimental settings, followed by a series of experiments to validate the following aspects. **Q1:** The effect of each proposed module on the model's answer accuracy. **Q2:** Comparison between the proposed method and the state-of-the-art method. **Q3:** Can CWT indeed mitigate the superficial understanding issue in SLM? **Q4:** The distinction between skip-thinking and full-thinking. **Q5:** Differences in SSC's inference acceleration across various tasks.

5.1 Experimental setting

Seven reasoning benchmarks, categorized into four distinct types: arithmetic, symbolic, common sense, and other logical reasoning, are employed to evaluate our method. Detailed information about the datasets can be found in Appendix A.1. For conciseness, we denote each dataset using abbreviations derived from their concatenated initials.

Unless otherwise stated, LLM in this section refers to *text-davinci-002* 175B, developed based on InstructGPT (Ouyang et al., 2022) and accessible via the OpenAI API. As for the student SLM, we employ GPT-2 (ranging from the base to large model) (Radford et al., 2019) and T5 (ranging from the small to large model) (Raffel et al., 2020) to evaluate the effectiveness of the proposed methods. The detailed generation parameters for LLMs and SLMs are given in Appendix A.2. More training details are available in the Appendix A.4.

5.2 Ablation experiments for Q1

First, we conduct a series of comprehensive experiments to assess the effectiveness of each proposed strategy. The results are presented in Table 1. Further experiments involving SLMs with varied

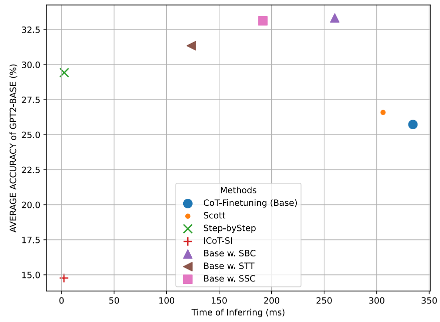

Figure 3: A comprehensive comparison of the average inference speed and performance across different methods on all datasets

parameters as student models are detailed in the Appendix D. The baseline model adopts the full-thinking training approach proposed by Ho et al. (2023).

It is evident that when chunks are partitioned using the AC, the performance of the SLM improves relative to the baseline across most tasks. But in a few tasks, the model's performance declines. We attribute this to the AC dividing coherent reasoning steps into separate chunks, thereby reducing SLM reasoning coherence. Thus, when a more optimal SBC is applied for chunking, the SLM exhibits improved performance across all tasks.

Building upon the SBC, we additionally apply STT to train the SLM. Subsequently, the trained SLM is required to skip one reasoning chunk each time during testing. As a result, the performance of the SLM is nearly identical to that of the SLM trained with SBC and generating full rationale during testing, while the reasoning speed experienced a significant improvement (see Figure 3). Finally, we incorporates the SSC based on skip thinking. In this case, the model's inference speed lies between that of full-thinking and skip-thinking, but its overall performance outperforms among the three.

It is noteworthy that we observed a significant performance decline of the skip-thinking on the LLC dataset. We attribute this to the fact that the LLC dataset requires parallel reasoning rather than sequential reasoning, where each reasoning step is independent with no context dependence between them. Therefore, when using the skip-thinking, the SLM fails to consider the skipped reasoning steps, which leads to a decline in performance. Based on this observation, we recommend using skip thinking only for tasks that require sequential reasoning.

Then, we verify the impact of different chunk numbers M on SBC. In Figure 4, we can observe that for tasks with relatively fixed reasoning methods and steps, such as common sense and symbolic

Methods	GPT2-base							T5-small						
	SE	AD	MA	Svamp	TSO	LLC	SQA	SE	AD	MA	Svamp	TSO	LLC	SQA
Base	8.55	10.08	14.44	10.66	56.88	21.33	58.22	3.94	8.40	8.66	9.00	60.00	45.33	56.04
Base w. AC	7.89	11.01	16.11	8.66	97.32	24.66	58.74	3.94	7.56	8.66	10.00	65.33	46.66	57.20
Base w. SBC	8.55	11.01	17.77	11.33	**100.00**	**25.33**	59.38	4.60	8.40	9.33	**10.55**	**99.55**	**48.66**	57.78
Base w. Skip	8.55	10.16	17.22	11.00	**100.00**	12.66	**59.91**	3.94	8.40	7.33	9.44	**99.55**	26.66	56.38
Based w. SSC	**9.21**	**12.60**	**18.88**	**12.00**	**100.00**	19.33	**59.91**	**5.92**	**9.24**	**10.00**	**10.55**	**99.55**	27.33	**58.51**

Table 1: The accuracy of various methods across different datasets. Refer to the Appendix D for additional ablation experiments using various student SLMs.

Methods	SE	AD	MA	Svamp	TSO	LLC	SQA	SE	AD	MA	Svamp	TSO	LLC	SQA
Text-davinci-002 (175B)	81.50	76.71	78.79	64.20	53.20	57.71	53.45	81.5	76.71	78.79	64.20	53.20	57.71	53.45
	GPT2-base (124M)							T5-small(60M)						
Standard finetune	8.55	10.08	14.44	10.66	56.88	21.33	58.22	3.94	8.40	8.66	9.00	60.00	45.33	56.04
CoT-Finetuing	8.55	10.08	14.44	10.66	56.88	21.33	58.22	3.94	8.40	8.66	9.00	60.00	45.33	56.04
Scott *	**9.21**	9.24	**22.22**	11.33	56.44	22	55.74	5.26	7.56	10.00	10.33	70.22	46.00	58.36
Step-by-Step	7.89	**12.60**	17.22	10.00	94.66	4.00	59.67	2.63	8.40	**10.55**	8.33	99.11	25.33	58.36
MMI	-	-	-	-	-	-	-	3.28	7.56	10.00	10.33	**99.55**	25.33	57.78
ICoT-SI	2.63	4.20	4.33	3.88	36.00	0.00	52.40	-	-	-	-	-	-	-
Ours	**9.21**	**12.60**	18.88	**12.00**	**100.00**	**25.33**	**59.91**	**5.92**	**9.24**	10.00	**10.55**	**99.55**	**48.66**	**58.51**

Table 2: A comparison of our methods with other approaches. A dash (-) indicates that the official code of the method is not implemented on the corresponding SLM. An asterisk (*) indicates that Scott requires the complete logits of each output token for implementation; thus, the rationales used in Scott are collected from the open-source model LLama3.1-70b-instruction (Dubey et al., 2024).

Method	Token type	AD(%)	TSO(%)
Base	core reasoning tokens	87.37	89.25
	other tokens	89.64	95.18
Base w. SBC	core reasoning tokens	88.88	92.73
	other tokens	89.80	95.17

Table 3: Confident score of GPT2-base for different tokens.

reasoning, the SLM works best when M is close to the average number of reasoning steps L. For mathematical reasoning, which has a large variation in reasoning methods and steps, setting M greater than L helps the SLM learn more solutions, thereby improving the performance of SLMs.

Third, the comparison of chunking result between AC and SBC are shown in Appendix E.2, which intuitively proves that SBC can better make the reasoning steps within a chunk more coherent.

5.3 Comparison with Other Methods for Q2

The comparison methods include standard finetuning (using only answers as label), few-shot prompting for LLMs (specific prompts can be found in the Ho et al. (2023)), full-thinking CoT distillation (CoT-Finetuning (Ho et al., 2023), Scott(Wang et al., 2023)), and distillation methods that accelerate SLM inference via multi-task learning(step by step (Hsieh et al., 2023), MMI (Chen et al., 2024b)) and internalized chains of thought (ICoT-SI (Deng et al., 2024). As shown in Table 2, our proposed method outperforms the other distillation approaches and achieves performance close to that of LLMs on certain tasks. Although the inference speed remains slower than that of multi-task learning and internalized chains of thought, it strikes a balance between performance and inference speed (see Figure 3).

5.4 Validate CWT for Q3

First, we show the performance of SLM as the token-level batch size changes in Figure 5. It can be seen that as the token-level batch size decreases, the performance of SLM on various reasoning tasks increases, which strongly verifies the motivation of CWT, that is, a smaller token-level batch size helps SLM converge to a flat minimum.

Subsequently, we further verify whether CWT helps SLM learn the core reasoning logic. Specifically, mathematical expressions (in AD) and key exchange results (in TSO) are identified and extracted as core reasoning tokens. Then, we counted the average confidence score of the core reasoning tokens and the non-reasoning tokens when the trained SLM output rationale. One can observe that compared with the base model, the gap between the confidence score of the core reasoning tokens and that of the common tokens is smaller after us-

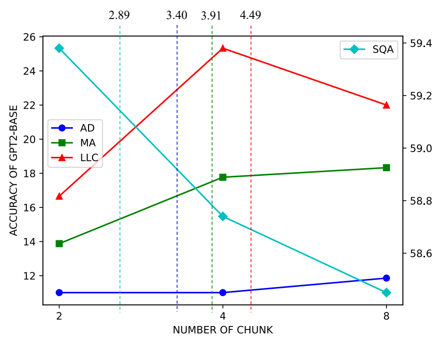

Figure 4: SLM performance trend when the number of chunks changes. The vertical dotted line refers to the average number of reasoning steps.

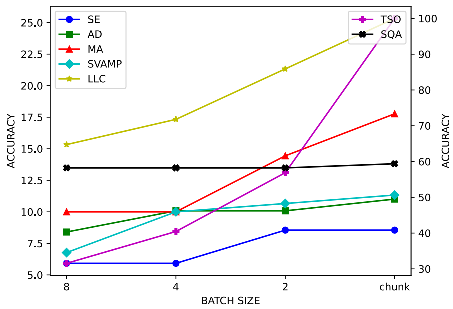

Figure 5: GPT2-base's performance trend when the batch size changes. Batch size is proportional to token-level batch size. Chunk means using CWT with SBC.

ing CWT, which means that the SLM with CWT is more confident when outputting the core reasoning tokens, i.e., it better understands the core reasoning logic of the current task.

Then, we show the cases (Appendix E.1) where the correct answer is inferred after using CWT compared to base because the core reasoning token is predicted correctly. This also proves that CWT helps SLMs comprehend the core reasoning logic.

Finally, the reasoning speed of the SLM trained with CWT based on SBC is faster than that of the baseline, which can be observed in Figure 3. We argue that this improvement stems from the SLM trained with the former focusing more on the correctness of the reasoning logic and exhibiting greater conciseness in its reasoning expressions. This conciseness is reflected in the length of the generated rationale. The average number of words in the rationale generated by the former across all tasks is 50, while the latter generates 56 words.

5.5 Validate Skip-thinking for Q4

In addition to verifying the speed-accuracy trade-off of skip-thinking shown in Figure 3, we conduct two additional experiments.

1) The impact of different skipping lengths. Since skip-thinking requires manually specified

Dataset	1	2	3
TSO	100.00	99.11	-
SQA	59.91	59.38	-
AD	10.16	10.08	9.24

Table 4: The trend of GPT2-base's performance changes caused by different skipping lengths. - means that the skip length is longer than M in the dataset

	SE	AD	SVAMP	MA	TSO	LLC	SQA
Mode 3 / SSC	1.16	1.24	1.34	1.37	1.97	1.41	1.64

Table 5: Reasoning speedup ratio of SSC compared to mode3 (full-thinking) on GPT2-base. TSO, LLC and SQA include answer options, whereas others do not.

skipping lengths, Table 4 shows SLM's performance under varying skipping lengths. Although overall performance remains similar, it generally decreases as skipping lengths increase. However, longer skipping lengths lead to shorter reasoning times, allowing for flexibility in choosing the appropriate length based on specific needs.

2) Case study. The Appendix E.3 presents a case study demonstrating the advantage of skip-thinking over full-thinking. By omitting intermediate reasoning steps, skip-thinking is less susceptible to model output hallucinations.

5.6 Validate SSC for Q5

In the Table 5, we summarize the reasoning speedup ratio of SSC relative to full-thinking (mode 3) across various tasks. The table demonstrates that when the answer space is constrained, SSC exhibits higher consistency between mode 1 and mode 2, leading to accelerated reasoning speed better. For tasks lacking answer options, SSC frequently resorts to mode 3 for reasoning, which consequently reduces its reasoning speed.

6 Conclusion

When using full rationale for CoT distillation, SLM faces two challenges: superficial understanding and slow response times. To address the two problems, we first propose CWT to reduce the token-level batch size, enhancing SLM's reasoning by mitigating gradient over-smoothing. To maintain coherence, a chunking method based on heuristic search to divide rationale into semantically coherent blocks is introduced. Building on CWT, STT trains SLM to infer from incomplete previous reasoning chunks, enabling skip-thinking during testing. Leveraging CWT and STT, the SLM achieves faster and more accurate reasoning.

Limitations

There are several possible limitations to using our method. 1) Skip thinking necessitates predefined skipping rules, which prevents it from adaptively determining whether to skip a reasoning chunk based on its importance. 2) Due to the addition of prefixes during training, the model requires iterative reasoning, thereby limiting the full utilization of the acceleration benefits provided by the K-V cache. 3) SBA employs a greedy search strategy, which may result in identifying only locally optimal chunk modes rather than globally optimal ones.

For the last point, strategies such as simulated annealing can be employed to avoid local optima. Regarding the first two points, addressing these limitations will be the focus of our future research.

Ethics Statement

Given that toxicity is present in LLMs, the student SLM may inherit such toxicity during the learning of the LLM's reasoning process. To address this issue, one can apply existing toxicity reduction techniques to mitigate toxicity in LLM reasoning.

References

Xiaoshu Chen, Sihang Zhou, Ke Liang, and Xinwang Liu. 2024a. Distilling reasoning ability from large language models with adaptive thinking. *Preprint*, arXiv:2404.09170.

Xin Chen, Hanxian Huang, Yanjun Gao, Yi Wang, Jishen Zhao, and Ke Ding. 2024b. Learning to maximize mutual information for chain-of-thought distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6857–6868, Bangkok, Thailand. Association for Computational Linguistics.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2023. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*.

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

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, and Archie Sravankumar et.al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Fengli Gao and Huicai Zhong. 2020. Study on the large batch size training of neural networks based on the second order gradient. *arXiv preprint arXiv:2012.08795*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. *Preprint*, arXiv:2310.02226.

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

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882, Toronto, Canada. Association for Computational Linguistics.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. 2018. Finding flatter minima with sgd.

Nitish Shirish Keskar, Jorge Nocedal, Ping Tak Peter Tang, Dheevatsa Mudigere, and Mikhail Smelyanskiy. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. 5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017.

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

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Bin Sun, Xinglin Wang, Heda Wang, and Kan Li. 2023. Turning dust into gold: Distilling complex reasoning capabilities from llms by leveraging negative data. *arXiv preprint arXiv:2312.12832*.

Weize Liu, Guocong Li, Kai Zhang, Bang Du, Qiyuan Chen, Xuming Hu, Hongxia Xu, Jintai Chen, and Jian Wu. 2024. Mind‘s mirror: Distilling self-evaluation capability and comprehensive thinking from large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6748–6763, Mexico City, Mexico. Association for Computational Linguistics.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Siyu Ren and Kenny Zhu. 2022. Specializing pre-trained language models for better relational reasoning via network pruning. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2195–2207, Seattle, United States. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, and et al. Aditya Gupta. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615.

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

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. In *The Twelfth International Conference on Learning Representations*.

Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. 2023. SCOTT: Self-consistent chain-of-thought distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5546–5558, Toronto, Canada. Association for Computational Linguistics.

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

Bohao Yang, Chen Tang, Kun Zhao, Chenghao Xiao, and Chenghua Lin. 2024. Effective distillation of table-based reasoning ability from LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5538–5550, Torino, Italia. ELRA and ICCL.

Yichun Zhao, Shuheng Zhou, and Huijia Zhu. 2024. Probe then retrieve and reason: Distilling probing and reasoning capabilities into smaller language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13026–13032, Torino, Italia. ELRA and ICCL.

Xuekai Zhu, Biqing Qi, Kaiyan Zhang, Xingwei Long, and Bowen Zhou. 2023. Pad: Program-aided distillation specializes large models in reasoning. *arXiv preprint arXiv:2305.13888*.

A Experimental Details

A.1 Datsets

To evaluate our model, we employ seven established benchmarks spanning four categories: Arithmetic (SingleEq (Koncel-Kedziorski et al., 2015), AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2015), Svamp (Patel et al., 2021)), Symbolic (Last Letter Concatenation (Kojima et al., 2022)), Common Sense (StrategyQA (Geva et al., 2021)), and General Logical Reasoning (Track Shuffled Objects (Srivastava et al., 2023)). We implement the training-test data partitioning adhering to the methodology described by (Ho et al., 2023).

A.2 Rationale generation of *Text-davinci-002*

We utilize the prompts described in Ho et al. (2023) to generate rationales from *Text-davinci-002*. The key modification involves swapping the positions of the rationale and the answer in the few-shot exemplars, enabling the LLM to leverage the answer information during reasoning. In alignment with the methodology outlined by Ho et al. (2023), we constrain the teacher-generated rationales to a maximum sequence length of 128. Additionally, we employ temperature sampling with T=0.7 to generate diverse rationales for each sample.

A.3 Rationale generation of SLM

The student model predictions are limited to a sequence length of 1024 and greedy decoding is applied for SLM across all benchmarks.

A.4 Training datails

For SLM training, we configure a batch size of 2, an initial learning rate of 1e-5, and a total of 50 epochs. We evaluate the SLM after each epoch. The learning rate follows a cosine annealing schedule with restarts, incorporating a warm-up phase of 1200 steps. We employ the Adam optimizer with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $weight\_decay = 0.1$ to optimize the model parameters. For search-based chunking, we set $\eta = 0.1$, as this value can empirically promote stable model training. As for the number of chunks M, We assign $M = 4$ for all arithmetic reasoning tasks and Last Letter Concatenation, and $M = 2$ for Track Shuffled Objects and StrategyQA. The effect of different M on SLM performance is shown in Figure 4.

[instruction] Please output strictly according to the format of Example. [example] Question: Alice, Bob, and Claire are playing a game. At the start of the game, they are each holding a ball: Alice has a orange ball, Bob has a purple ball, and Claire has a pink ball. As the game progresses, pairs of players trade balls. First, Alice and Claire swap balls. Then, Bob and Alice swap balls. Finally, Alice and Claire swap balls. At the end of the game, Alice has the Which choice is true? Answer choices: (A) purple ball. (B) orange ball. (C) pink ball.Why the answer is B. Explanation: 1. Alice-orange, Bob-purple, Claire-pink ball. 2. Alice-pink, Bob-purple, Claire-orange. 3. Alice-purple, Bob-pink, Claire-orange. 4. Alice-orange, Bob-pink, and Claire-purple. Question:{# question} Why the answer is {# Answer} Explanation:

Table 6: The prompt for concise rationale.

	Base	Base w. Weight	Base w. Refine
GPT2-base (124M)			
TSO	37.33	36.88	43.11
GPT2-medium (355M)			
TSO	41.77	42.22	36.88

Table 7: The accuracy of different methods on TSO. Base refers to Ho et al. (2023) without diverse rationale. Base w. Weight and Base w. Refine represent the two naive solutions to address the oversmoothing problem.

B Naive Method for Oversmoothing

There are two naive solutions to solve the oversmoothing problem, namely weighted and refined rationale. Specifically, the first solution involves increasing the loss weight for core reasoning tokens in the rationale, while the second solution focuses on designing prompts to guide the LLM in generating refined rationales with minimal non-reasoning content.

In this work, we evaluate the feasibility of these two solutions using the Track Shuffled Objects (TSO) dataset. For the weighted solution, we leverage tokens from key exchanging results in every step as the most core reasoning tokens in the rationale. Subsequently, the loss weight for these core tokens is doubled compared to the remain-

Methods	GPT2-medium (355M)							T5-base (220M)						
	SE	AD	MA	SVAMP	TSO	LLC	SQA	SE	AD	MA	Svamp	TSO	LLC	SQA
Base	11.84	15.96	18.88	10.00	67.11	24.00	59.24	6.57	10.92	17.22	10.66	71.11	64.66	54.87
Base w. AC	11.18	17.64	17.22	10.00	75.89	26.66	60.64	8.55	11.76	16.11	12.66	77.33	74.00	60.11
Base w. SBC	12.50	19.32	19.44	10.66	87.11	28.66	61.13	9.21	13.44	17.77	13.33	93.33	79.33	60.98
Base w. Skip	11.18	18.48	18.33	10.66	100.00	14.00	61.28	8.55	13.44	18.33	12.66	99.55	40.66	61.71
Based w.(SSC)	13.15	19.32	20.00	11.00	100.00	16.00	61.71	9.86	15.12	19.44	13.66	100.00	80.66	61.86
Methods	GPT2-large (774M)							T5-large(700M)						
Base	13.15	15.96	20.00	11.00	68.88	25.33	60.84	9.24	14.28	17.77	12.33	92.44	76.66	57.64
Base w. AC	12.50	16.80	21.33	12.22	85.71	27.33	61.57	8.55	15.96	13.88	13.33	95.08	81.33	61.71
Base w. SBC	16.44	17.64	23.33	14.00	94.66	28.66	62.44	10.52	16.80	19.44	14.00	100.00	82.66	63.75
Base w. Skip	15.13	16.80	22.22	14.00	100.00	15.33	62.88	9.21	15.96	19.44	13.33	100.00	57.33	63.90
Based w.(SSC)	18.42	19.32	23.88	15.00	100.00	29.33	63.46	11.18	18.48	21.11	15.33	100.00	59.33	64.04

Table 8: The performance of SLM under different models and different training strategies

ing tokens. For the refined rationale solution, we design prompts (shown in the Table 6) to guide the LLM *GPT-3.5-Turbo* in generating the most concise rationales.

The results of both solutions are presented in the Table 7. The results indicate that the weighted solution performs similarly to the baseline, suggesting its effectiveness is limited. Moreover, even if this solution exhibits some effectiveness, its applicability is limited, as not all tasks can identify core reasoning tokens through artificial rules, as in TSO. The refined rationale solution demonstrates effectiveness for smaller model sizes. However, for larger model sizes, the reduced information content compared to normal rationales leads to overfitting, resulting in performance inferior to the baseline.

C Sentence-wise and step-wise training

In addition to partitioning into a fixed number of chunks, we also segment the rationale by sentences or reasoning steps, enabling the SLM to learn only one sentence or reasoning step per training iteration. For both approaches, we evaluate two schemes: one incorporating prefixes like the CWT with AC and one without prefixes. The detailed results of these approaches on the TSO dataset are presented in the Table 9. As shown, the performance of all approaches exhibits a decline. The Table 10 also highlights the most frequent failure cases for these schemes. It can be observed that these schemes often generate repetitive reasoning steps until the maximum generation length is reached. This occurs because the number of chunks resulting from sentence- or step-based segmentation is typically variable, making it challenging for the SLM to determine the required number of reasoning steps for different problems after chunk-wise training.

D Extension of Ablation study

We further conduct extensive ablation experiments on SLMs with varying parameters. The results are presented in the Table 8. The results demonstrate that the proposed strategy performs effectively across various conditions.

E Case study

E.1 Core reasoning tokens

Figure 6 presents six cases across different types of benchmarks, demonstrating the improvement in the SLM's core reasoning logic following CWT training.

E.2 Comparsion between AC and SBC

The Figure 7 illustrates the differences in chunk division results between AC and SBC. As shown, the SBC division results in chunks with more coherent internal semantics.

E.3 The case for skip-thinking.

Figure 8 demonstrates that skip-thinking reduces the risk of SLM's hallucinations in rationale generation compared to full-thinking.

Question: Henry had 11 dollars. For his birthday he got 18 more dollars but spent 10 on a new game. How much money does he have now? **Answer:** 19
Rationale of base: Henry had 11 dollars. For his birthday, he got 18 more dollars. That means he had 29 dollars in total. But he spent 10 on a new game. That means he has 18 less dollars. That means he has **29 - 10 = 18** dollars left. Therefore, the answer is 18. **Answer of base:** 18
Rationale of SBC: First, Henry had 11 dollars. Then, he got 18 more dollars for his birthday, so he had 29 dollars in total. However, he spent 10 dollars on a new game, so he has 19 dollars left. 19 **Answer of SBA:** 19

(a) case 1.

Question: There are some baskets of peaches. Each basket has 4 red peaches and 3 green peaches. If there are a total of 7 peaches in all baskets How many baskets of peaches are there? **Answer:** 1.0
Rationale of base: There are a total of 7 peaches in all baskets. Each basket has 4 red peaches and 3 green peaches. If there are a total of 7 peaches in all baskets, then there are 7 groups of 4 red peaches and 3 green peaches. Therefore, there are **2** baskets of peaches in all baskets. Therefore, the answer is 2.0. **Answer of base:** 2.0
Rationale of SBC: There are 7 baskets. Each basket has 4 red peaches and 3 green peaches. Therefore, there are 1 basket of peaches. 1.0 **Answer of SBA:** 1.0

(b) case 2.

Question: Mary picked 122 oranges and Jason picked 105 oranges from the orange tree . How many oranges were picked in total ? **Answer:** 227
Rationale of base: Mary picked 122 oranges. Jason picked 105 oranges. How many oranges were picked in total? There were a total of **252** oranges picked. Therefore, the answer is 252 **Answer of base:** 252
Rationale of SBC: First, we need to find out how many oranges were picked in total. To do this, we add up the number of oranges that each person picked. So, we have 122 + 105 = 227 oranges. 227 **Answer of SBA:** 227

(c) case 3.

Question: Take the last letters of each words in "Garry Arielle Elba Rodney" and concatenate them. **Answer:** yeay
Rationale of base: The last letter of "Garry" is "y". The last letter of "Arielle" is "e". The last letter of "Elba" is "a". The last letter of "Rodney" is "y". So the final answer is **"yeaa".** Therefore, the answer is yeaa. **Answer of base:** yeaa
Rationale of SBC: The last letter of "Garry" is "y". The last letter of "Arielle" is "e". The last letter of "Elba" is "a". The last letter of "Rodney" is "y". So the final answer is "yeay". yeay **Answer of SBA:** yeay

(d) case 4.

Question: Alice, Bob, and Claire are dancers at a square dance. At the start of a song, they each have a partner: Alice is dancing with Rodrigo, Bob is dancing with Jamie, and Claire is dancing with Izzi. Throughout the song, the dancers often trade partners. First, Bob and Alice switch partners. Then, Claire and Alice switch partners. Finally, Bob and Claire switch partners. At the end of the dance, Claire is dancing with Which choice is true? Answer choices: (A) Rodrigo, (B) Jamie, (C) Izzi. **Answer:** A
Rationale of base: First, Bob and Alice switch partners. So, at the end of this step, Alice is dancing with Jamie and Bob is dancing with Rodrigo. Then, Claire and Alice switch partners. So, at the end of this step, Alice is dancing with Izzi and Claire is dancing with Jamie. Finally, Bob and Claire switch partners. So, at the end of this step, Bob is dancing with **Izzi** and Claire is dancing with Rodrigo. Therefore, the answer is **C**. **Answer of base:** C
Rationale of SBC: At the start of the song, Alice is dancing with Rodrigo, Bob is dancing with Jamie, and Claire is dancing with Izzi. First, Bob and Alice switch partners. This means that now Alice is dancing with Jamie and Bob is dancing with Rodrigo. Then, Claire and Alice switch partners. This means that now Alice is dancing with Izzi and Claire is dancing with Jamie. Finally, Bob and Claire switch partners. This means that now Claire is dancing with Rodrigo and Bob is dancing with Jamie. A **Answer of SBA:** A

(e) case 5.

Question: Is 3D printing able to make adenovirus? **Answer:** No
Rationale of base: **Can 3D print anything? Yes.** Can adenovirus be made? Yes. Therefore, 3D printing can make adenovirus. Therefore, the answer is Yes. **Answer of base:** Yes
Rationale of SBC: 3D printing can create things. Adenoviruses are small, virus-like organisms that can only replicate inside the cells of a living organism. So, it is unlikely that 3D printing could create adenovirus. No **Answer of SBA:** No

(f) case 6.

Figure 6: The case for core reasoning tokens.

	Base	Base w. sent	Base w. sent prefix	Base w. step	Base w. step prefix
TSO	37.33	7.11	27.11	14.22	30.22

Table 9: The accuracy of training the SLM using rationale partitioning methods with varying granularities.

Question: Alice, Bob, and Claire are dancers at a square dance. At the start of a song, they each have a partner: Alice is dancing with Ophelia, Bob is dancing with Rodrigo, and Claire is dancing with Patrick. Throughout the song, the dancers often trade partners. First, Bob and Alice switch partners. Then, Claire and Bob switch partners. Finally, Claire and Alice switch partners. At the end of the dance, Bob is dancing with Which choice is true? Answer choices: (A) Rodrigo. (B) Ophelia. (C) Patrick.
Rationale: Sure, let's break it down step by step. At the start of a song, Alice is dancing with Ophelia, Bob is dancing with Rodrigo, and Claire is dancing with Patrick. After the first partner switch, Bob is now dancing with Ophelia and Alice is dancing with Rodrigo. After the secend partner switch, ..., After the third partner switch, ..., After the third partner switch, ..., After the third partner switch, ...

Table 10: The base case for sentence-wise and step-wise training.

Question: Rachel had to complete 4 pages of reading homework, 7 pages of math homework and 19 more pages of biology homework. How many more pages of math homework than reading homework did she have?
Rationale of LLM: First, we need to find out how many pages of homework she had in total. \n4 + 7 + 19 = \n30\nNext, we need to find out how many pages of reading homework she had. \n4 \nThen, we need to find out how many pages of math homework she had. \n7 \nLast, we need to find out how many more pages of math homework she had than reading homework. \n7 - 4 = \n3
AC:
< First, we need to find out how many pages of homework she had in total. 4 + 7 + 19 = > < 30 Next, we need to find out how many pages of reading homework she had. > < 4 Then, we need to find out how many pages of math homework she had. > < 7 Last, we need to find out how many more pages of math homework she had than reading homework. 7 - 4 = 3>
SBC:
< First, we need to find out how many pages of homework she had in total. 4 + 7 + 19 = 30 > < Next, we need to find out how many pages of reading homework she had. 4 > < Then, we need to find out how many pages of math homework she had. 7> < Last, we need to find out how many more pages of math homework she had than reading homework. 7 - 4 = 3 >

(a) case 1.

Question: Alyssa picked 17 plums and Jason picked 10 plums . Melanie picked 35 pears . How many plums were picked in all ?
Rationale of LLM: Alyssa picked 17 plums. \nJason picked 10 plums. \n17 + 10 = 27 plums. \nMelanie picked 35 pears. \n27 + 35 = 62 \nThere were 62 fruits picked in all.
AC:<[Alyssa picked 17 plums. > <Jason picked 10 plums. > <17 + 10 = 27 plums. > < Melanie picked 35 pears. 27 + 35 = 62. There were 62 fruits picked in all.]
SBC:
< Alyssa picked 17 plums. Jason picked 10 plums. 17 + 10 = 27 plums. > < Melanie picked 35 pears. 27 + 35 = 62. > < There were 62 fruits picked in all. > < >

(b) case 2.

Question: Take the last letters of each words in "Martin Sage Tanisha Rick" and concatenate them.
Rationale of LLM: If we split the string into a list of words, we get:\n ['Martin', 'Sage', 'Tanisha', 'Rick']\nFrom there, we can take the last letter of each word using indexing:\n['n', 'e', 'a', 'k']\nLastly, we can concatenate these letters together into a string:\n'neak'
AC:
< If we split the string into a list of words, we get:> <['Martin', 'Sage', 'Tanisha', 'Rick'> < From there, we can take the last letter of each word using indexing:> <['n', 'e', 'a', 'k'] Lastly, we can concatenate these letters together into a string: 'neak'>
SBC:
< If we split the string into a list of words, we get: ['Martin', 'Sage', 'Tanisha', 'Rick']> < From there, we can take the last letter of each word using indexing: ['n', 'e', 'a', 'k']> < Lastly, we can concatenate these letters together into a string: ['n', 'e', 'a', 'k'] > <>

(c) case 3.

Question: Alice, Bob, and Claire are holding a white elephant gift exchange. At the start of the event, they are each holding a present of a different color: Alice has a yellow present, Bob has a brown present, and Claire has a blue present. As the event progresses, pairs of people swap gifts. First, Bob and Alice swap their gifts. Then, Claire and Alice swap their gifts. Finally, Bob and Alice swap their gifts. At the end of the event, Claire has the Which choice is true? Answer choices: (A) yellow present, (B) brown present, (C) blue present.
Rationale of LLM: First, Bob and Alice swap gifts. This means that: \n-Alice now has the brown present \n-Bob now has the yellow present \nNext, Claire and Alice swap gifts. This means that: \n-Alice now has the blue present \n-Claire now has the brown present \nFinally, Bob and Alice swap gifts again. This means that: \n-Alice now has the yellow present \n-Bob now has the blue present \nTherefore, Claire has the brown present at the end of the event.
AC:
<First, Bob and Alice swap gifts. This means that:> <-Alice now has the brown present> <-Bob now has the yellow present> <Next, Claire and Alice swap gifts. This means that: -Alice now has the blue present -Claire now has the brown present Finally, Bob and Alice swap gifts again. This means that: -Alice now has the yellow present -Bob now has the blue present Therefore, Claire has the brown present at the end of the event.>
SBC:
< First, Bob and Alice swap gifts. This means that: -Alice now has the brown present -Bob now has the yellow present Next, Claire and Alice swap gifts.> < This means that: -Alice now has the blue present> < -Claire now has the brown present Finally, Bob and Alice swap gifts again. > < This means that: -Alice now has the yellow present -Bob now has the blue present Therefore, Claire has the brown present at the end of the event. >

(d) case 4.

Figure 7: The case for SBC. <·> represents a chunk.

Question: Jenny ran 0.6 of a mile and walked 0.4 of a mile. How much farther did Jenny run than walk? **Answer:** 0.2
Rationale of SBC: We know that Jenny ran 0.6 of a mile and walked 0.4 of a mile. This means that she ran 4/10th of a mile **and walked 3/10th of a mile.** 3. 0
Answer of SBC: 3.0
Rationale of Skip: We know that she walked 0.4 of a mile, so we can multiply that by 1 mile to get 0.4 miles. This means that Jenny ran 0.2 miles farther than she walked. 0.2
Answer of Skip: 0.2

(a) case 1.

Question: Jack received 6 emails in the morning and 8 emails in the afternoon. How many more emails did Jack receive in the afternoon than in the morning? **Answer:** 2.0
Rationale of SBC: In the morning, Jack received 6 emails. In the afternoon, Jack received 8 emails. **In total, Jack received 14 emails in the afternoon.** 14.0
Answer of SBC: 14.0
Rationale of Skip: In the afternoon, Jack received 8 more emails. 2.0
Answer of Skip: 2.0

(b) case 2.

Question: Is Sirius part of a constellation of an animal? **Answer:** Yes
Rationale of SBC: **There are many different types of animals, so it is impossible to say which is the first or the last. However, there are many different types of companions, so it is also impossible to say which is the first companion. There are many different types of pets, so it is also impossible to say which is the first pet. No** **Answer of SBC:** No
Rationale of Skip: So, based on what we know, we can conclude that Sirius is part of a constellation of an animal. Yes **Answer of Skip:** Yes

(c) case 3.

Question: Take the last letters of each words in "Noelia Cassidy Ashok Francisco" and concatenate them. **Answer:** ayko
Rationale of SBC: The last letter of "Noelia" is "a". The last letter of "Cassidy" is "y". The last letter of "Ashok" is "k". The last letter of "Francisco" is "o". So the final answer is "ayko". ayko
Answer of SBC: ayko
Rationale of Skip: The last letter of "Ashok" is "k". So the final answer is "ayok". ayok **Answer of Skip:** ayok

(d) case 4.

Figure 8: The case for Skip-thinking.