Fewer is More: Boosting Math Reasoning with Reinforced Context Pruning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have shown impressive capabilities, yet they still struggle with math reasoning. In this work, we propose **CoT-Influx**, a novel approach that pushes the boundary of few-shot Chain-of-Thoughts (CoT) learning to improve LLM mathematical reasoning. Motivated by the observation that adding more concise CoT examples in the prompt can improve LLM reasoning performance, CoT-Influx employs a coarse-to-fine pruner to maximize the input of effective and concise CoT examples. The pruner first selects as many crucial CoT examples as possible and then prunes unimportant tokens to fit the context window. As a result, by enabling more CoT examples with double the context window size in tokens, CoT-Influx significantly outperforms various prompting baselines across various LLMs (LLaMA2-7B, 13B, 70B) and 5 math datasets, achieving up to 4.55% absolute improvements. Remarkably, without any fine-tuning, LLaMA2-70B with CoT-Influx surpasses GPT-3.5 and a wide range of larger LLMs (PaLM, Minerva 540B, etc.) on GSM8K. CoT-Influx is a plug-and-play module for LLMs, adaptable in various scenarios. It's compatible with advanced reasoning prompting techniques, such as self-consistency, and supports different long-context LLMs, including Mistral-7B-v0.3-32K and Yi-6B-200K.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a range of tasks (Brown et al., 2020; OpenAI, 2023a). However, it remains a significant challenge to improve LLM performance on math reasoning tasks. While existing efforts focus on optimizing Chain-of-Thought (CoT) prompts (Wei et al., 2022; Wang et al., 2023d; Yao et al., 2023) and fine-tuning LLMs (Luo et al., 2023) under the zero-shot setting, the potential of few-shot learning in improving LLM reasoning has not been fully explored. Inspired by the human reasoning process, we propose the hypothesis: *if LLMs are exposed to more step-by-step problem-solving examples (i.e., CoTs) before answering questions, it could potentially improve LLMs reasoning capability to generate a correct solution*. This leads us to ask: *what's the boundary of LLM reasoning capability achievable through inputting more CoT examples?*

We face two major obstacles. First, the limited token length of LLMs' context window restricts the number of few-shot examples. Extending the context window is one solution, but it requires expensive fine-tuning and increases inference overhead (Peng et al., 2023a; Ding et al., 2024). While prompt compression (Li et al., 2023b; Jiang et al., 2023b) is another approach, it underperforms in math reasoning. Tokens like numerical and format ones, though identified as redundant, are crucial for few-shot prompting to solve math problems.

Second, it's challenging to select effective CoT examples. Section 3 reveals that random choices can even harm reasoning performance. Existing retrieval-based methods (Liu et al., 2021; Scarlatos and Lan, 2023) are not tailored for math reasoning, making them suboptimal. These retrieved examples are model-agnostic, while we found that different LLMs favor CoT examples of varying characteristics (e.g., diverse difficulty levels).

In this work, we propose **CoT-Influx**, which addresses all the above challenges and pushes the boundaries of utilizing few-shot learning to improve LLM math reasoning capability. CoT-Influx is motivated by the observation that *current LLM context window has not been fully utilized due to redundancy at both the example and token levels in natural language input*. As such, these redundant inputs can be pruned to free up space for more informative context. The central idea of CoT-Influx is to input long lengthy CoT examples, select the crucial examples for the target LLM, and then prune redundant tokens. As a result, by inputting much more helpful CoT examples, each composed solely of informative tokens and with a shorter length, we greatly improve LLM math reasoning ability. Moreover, as all these inputs remain within the context window, we do not increase any inference overhead. This stands in stark contrast to other methods (Hao et al., 2022; Chen et al., 2023a).

CoT-Influx treats the target LLM as a black box, and serves as a plug-and-play module for LLMs as shown in Fig. 3. The key module is a coarse-to-fine pruner consisting of two steps: (i) a shot pruner first selects the most helpful CoT examples from candidates, and (ii) a token pruner then removes unimportant tokens from selected CoT examples.

However, training the pruner presents two challenges: (1) since we identify discrete tokens before the LLM tokenizer, the loss gradient cannot be backpropagated through the tokenizer to update the pruner; (2) The high difficulty of many math problems, which consistently yield incorrect answers regardless of the quality of compressed few-shot examples, poses a challenge to the effective training of the pruner. To this end, we introduce a novel training approach with reinforcement learning to mitigate the gradient issue. We design a reward function to measure the LLM loss, few-shot math reasoning effectiveness, and token length constraints. Then, we propose a difficulty-aware dataloader filtering appropriate problems and introduce two techniques to stabilize the RL training.

Extensive experiments on various LLMs and diverse reasoning datasets demonstrate the effectiveness of CoT-Influx. CoT-Influx significantly boosts LLM reasoning capability, achieving up to 14.09% absolute improvements over SOTA baselines, and establishes a new prompting-based benchmark in math reasoning accuracy without any fine-tuning or additional inference costs. Remarkably, LLaMA2-70B with CoT-Influx outperforms a broad range of larger LLMs and surpasses GPT-3.5 by 2.5% on GSM8K. Moreover, we prove that CoT-Influx is also highly beneficial on long-context LLMs. For example, we achieve ∼2.5% higher GMS8K accuracy with 15× fewer input tokens on Yi-6B-200K.

2 Related Works

LLMs for Math Reasoning. Drawing from the Chain-of-Thought (CoT) (Wei et al., 2022), recent research has greatly improved the reasoning capabilities of LLMs by providing step-by-step reasoning paths. The main efforts are twofold: enhancing

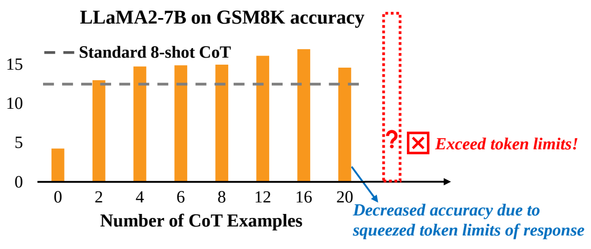


Figure 1: LLaMA2-7B reasoning accuracy under an increasing number of TopK retrieved CoT examples.

CoT prompts, such as Program-of-Thoughts (Chen et al., 2023b), Tree-of-Thoughts (Yao et al., 2023), and Everything-of-Thoughts (Ding et al., 2023), and innovating CoT-based training data for fine-tuning LLMs like WizardMath (Luo et al., 2023).

However, most works focus on the zero-shot setting with only task instruction or CoT prompts, leaving the potential of few-shot CoT largely untapped. We explore leveraging few-shot CoT learning to improve LLMs' math reasoning capabilities.

Prompt Compression. One related works involve compressing input prompts, including (1) token pruning (Kim et al., 2022; Li et al., 2023a); (2) soft prompt compression (Mu et al., 2023; Chevalier et al., 2023); and (3) information-entropy-based approaches (Li et al., 2023b; Jiang et al., 2023b).

However, they do not effectively solve our problem for two reasons. First, they prune tokens based on designed metrics, often failing to remove redundancy of the entire CoT examples. Second, some tokens such as numerical and format tokens, although redundant, are crucial for math reasoning.

Prompt Retrieval aims to optimize LLM performance by selecting effective few-shot examples. Heuristic methods, such as the TopK retrieval (Liu et al., 2021; Gao et al., 2021), BM25 (Robertson et al., 2009), VoteK (Hongjin et al., 2022), and entropy (Lu et al., 2022), select examples based on semantic similarity. Recently, supervised-based methods like EPR (Rubin et al., 2021), LLM-R (Wang et al., 2023b), and IDS (Qin et al., 2023) have been proposed, which train a retrieval model to learn better example selection. However, these methods do not consider token redundancy, which can limit the number of retrieved examples.

3 Pilot Study

This section presents our key observations of few-shot learning in improving LLMs math reasoning, upon which the CoT-Influx design is based. Note that experiments are done with our CoT dataset, MRD^3, as introduced in Sec. 4.1.

Observation 1: *LLMs can improve reasoning with more helpful CoT examples, but the context window length limits further accuracy gains*.

A standard practice for evaluating LLMs' math reasoning capability is the use of 8-shot manually-designed CoTs (Wei et al., 2022). We increase the number of CoT shots to see if reasoning accuracy improves. To avoid poor-quality examples, we use the TopK method (Liu et al., 2021) to select the k most relevant CoT examples for each question. Given LLaMA2's context window limit of 4096 tokens, we could only input up to 20 CoT examples[1]. As Fig. 1 shows, increasing CoT examples improves LLaMA2-7B's reasoning accuracy on the GSM8K dataset, significantly outperforming the standard 8-shot setting. However, the limited LLM context window hinders the full potential of few-shot CoT learning for improving math reasoning. For instance, even with 20 CoTs not hitting the token limit, accuracy drops as the large input context limits the LLM's response space.

Observation 2: *CoT example selection is crucial for math reasoning. Simply adding CoT examples randomly doesn't boost performance*.

The prior study suggests that more CoT examples can improve LLM reasoning performance. However, the quality of CoT examples is crucial to the final performance. As shown in Table 1, even with up to 16 CoT shots, random selection underperforms the standard 8-shot setting, which is manually curated for quality.

Table 1: The selection of CoT examples heavily impacts LLM math reasoning performance on GSM8K.

Model	Manual 8 Shots	Random 16 Shots
LLaMA2-7B	13.79	12.82±0.94
LLaMA2-13B	27.82	23.16±0.42

Observation 3: *A CoT example contains redundant tokens for math reasoning, which can be pruned to free up space for more informative content.*

Observation 2 indicates that few-shot CoT examples contain useless or even harmful examples that can be pruned. We further observe that a CoT example often has redundant tokens. For instance, the blue tokens in Fig. 2 can be removed without affecting LLM performance. However, identifying redundant tokens for math reasoning poses a challenge. Simply using existing prompt compression methods (Jiang et al., 2023b; Li et al., 2023b) leads to a significant performance decline. Fig. 2 shows a compressed example using LLMLingua (Jiang et al., 2023b). Some numerical and format tokens (colored in red), while identified as redundant, are crucial for LLM to comprehend the context for solving a math problem.

A compressed CoT example by LLMLingua:

Q: There are 15 trees in the ~~grove~~. ~~Grove~~ workers will plant trees in ~~the grove today~~. After they are done, ~~there will be~~ 21 trees~~.~~ ~~How many~~ trees did ~~the grove workers~~ plant ~~today?~~

~~A:~~ Let's think ~~step by~~ step. ~~There are 15~~ trees originally. Then there ~~were 21 trees~~ after some more ~~were planted~~. So there must ~~have been 21 - 15 = 6~~. The ~~answer is 6.~~

Figure 2: A compressed CoT example using the prompt compression tool of LLMLingua (Jiang et al., 2023b). The pruned tokens contain truly redundant tokens (colored in blue) and crucial tokens (colored in red).

4 CoT-Influx Methodology

Motivated by our observations, this section introduces CoT-Influx, which maximizes CoT examples within the original LLM context window by identifying the most important CoT examples and tokens from a long input context.

4.1 CoT Dataset Collection

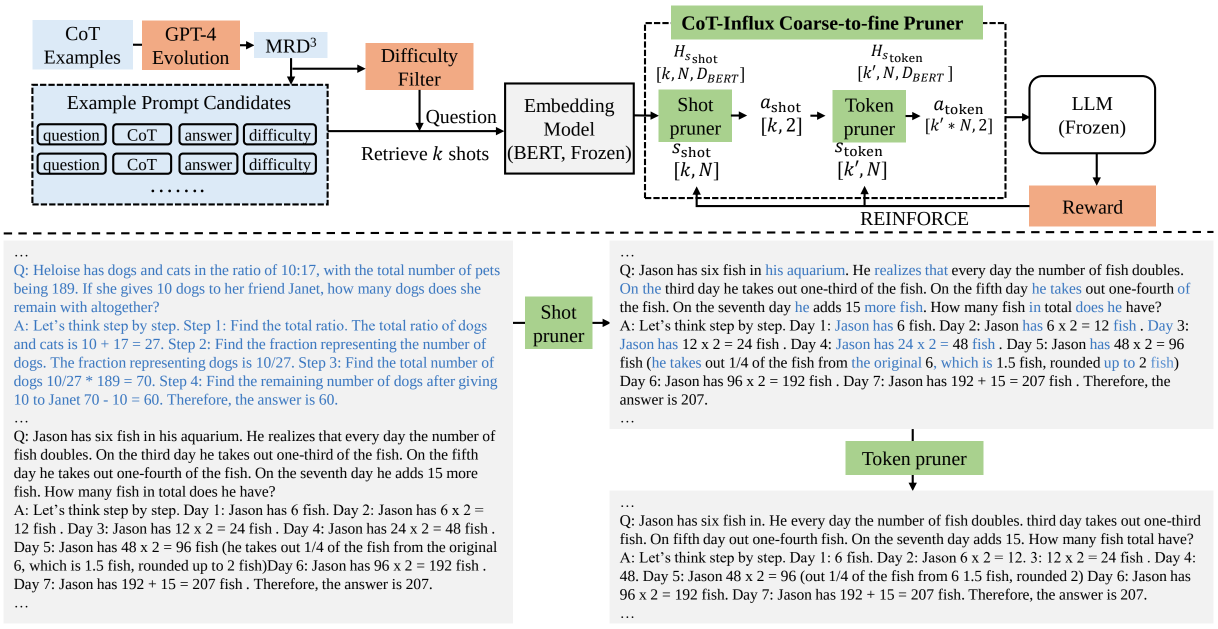
We start by collecting a high-quality math reasoning dataset comprising diverse CoT examples with varying steps and difficulties. We merge the training set of GSM8K (Cobbe et al., 2021), MAWPS, MAWPS-single (Koncel-Kedziorski et al., 2016), and 1000 random examples from AQuA (Ling et al., 2017) to create an initial dataset of 9.7K question-answer pairs. Then, we prompt GPT-4 to generate formatted CoT reasoning steps. We apply 5 mutation schemes, three to increase reasoning difficulty and two to simple questions. The final dataset is referred to as *M*ath *R*easoning *D*ataset with *D*iverse *D*ifficulty (*MRD*^{3}). The details of dataset evolution are listed in Section. B in the Appendix.

4.2 Problem Formulation

Let $\mathcal{D}$ denote the CoT dataset (i.e., the MRD^3), and $\hat{\mathcal{D}} = \{x_i^{\text{cot}}\}_{i=1}^{k}$ be a subset of k CoT examples, each composed of a question, reasoning steps, and an answer. The total number of tokens in these k CoT examples far exceeds the LLM context window length limit of T. CoT-Influx is designed to perform a two-step pruning process:

$$\hat{\mathcal{D}} = \{x_i^{\text{cot}}\}_{i=1}^{k} \xrightarrow{\text{Shot Pruner}} \{x_j^{\text{cot}}\}_{j=1}^{k'} \xrightarrow{\text{Token Pruner}} \{\hat{x}_j^{\text{cot}}\}_{j=1}^{k'} \quad (1)$$

[1]The input token length is less than the context window token limit, as the answer generation also shares this limit.


CoT Examples
GPT-4 Evolution
MRD³
Difficulty Filter
Example Prompt Candidates
question
CoT
answer
difficulty
question
CoT
answer
difficulty
....
Question
Retrieve k shots
Embedding Model (BERT, Frozen)
CoT-Influx Coarse-to-fine Pruner
$H_{s_{\text{shot}}}$ $[k, N, D_{BERT}]$
$H_{s_{\text{token}}}$ $[k', N, D_{BERT}]$
Shot pruner
a_{shot} $[k, 2]$
Token pruner
a_{token} $[k' * N, 2]$
s_{shot} $[k, N]$
s_{token} $[k', N]$
LLM (Frozen)
Reward
REINFORCE
Shot pruner
Token pruner

…
Q: Heloise has dogs and cats in the ratio of 10:17, with the total number of pets being 189. If she gives 10 dogs to her friend Janet, how many dogs does she remain with altogether?
A: Let's think step by step. Step 1: Find the total ratio. The total ratio of dogs and cats is 10 + 17 = 27. Step 2: Find the fraction representing the number of dogs. The fraction representing dogs is 10/27. Step 3: Find the total number of dogs 10/27 * 189 = 70. Step 4: Find the remaining number of dogs after giving 10 to Janet 70 - 10 = 60. Therefore, the answer is 60.
…
Q: Jason has six fish in his aquarium. He realizes that every day the number of fish doubles. On the third day he takes out one-third of the fish. On the fifth day he takes out one-fourth of the fish. On the seventh day he adds 15 more fish. How many fish in total does he have?
A: Let's think step by step. Day 1: Jason has 6 fish. Day 2: Jason has 6 x 2 = 12 fish . Day 3: Jason has 12 x 2 = 24 fish . Day 4: Jason has 24 x 2 = 48 fish . Day 5: Jason has 48 x 2 = 96 fish (he takes out 1/4 of the fish from the original 6, which is 1.5 fish, rounded up to 2 fish)Day 6: Jason has 96 x 2 = 192 fish . Day 7: Jason has 192 + 15 = 207 fish . Therefore, the answer is 207.
…

…
Q: Jason has six fish in his aquarium. He realizes that every day the number of fish doubles. On the third day he takes out one-third of the fish. On the fifth day he takes out one-fourth of the fish. On the seventh day he adds 15 more fish. How many fish in total does he have?
A: Let's think step by step. Day 1: Jason has 6 fish. Day 2: Jason has 6 x 2 = 12 fish . Day 3: Jason has 12 x 2 = 24 fish . Day 4: Jason has 24 x 2 = 48 fish . Day 5: Jason has 48 x 2 = 96 fish (he takes out 1/4 of the fish from the original 6, which is 1.5 fish, rounded up to 2 fish) Day 6: Jason has 96 x 2 = 192 fish . Day 7: Jason has 192 + 15 = 207 fish . Therefore, the answer is 207.
…

…
Q: Jason has six fish in. He every day the number of fish doubles. third day takes out one-third fish. On fifth day out one-fourth fish. On the seventh day adds 15. How many fish total have?
A: Let's think step by step. Day 1: 6 fish. Day 2: Jason 6 x 2 = 12. 3: 12 x 2 = 24 fish . Day 4: 48. Day 5: Jason 48 x 2 = 96 (out 1/4 of the fish from 6 1.5 fish, rounded 2) Day 6: Jason has 96 x 2 = 192 fish. Day 7: Jason has 192 + 15 = 207 fish. Therefore, the answer is 207.
…

Figure 3: *Above*: The overview procedure of CoT-Influx; *Below*: an example illustrating the use of CoT-Influx to first prune entire CoT examples and then prune tokens. The blue-highlighted text indicated the pruned context.

Initially, unhelpful CoT examples are pruned from $\hat{\mathcal{D}}$, resulting in a reduced set of k' examples. Then, for each retained CoT example x^{cot}, redundant tokens are pruned, yielding a shorter example, $\hat{x}^{\text{cot}}$.

Let x^{question} denote the question that LLM is tasked to solve. For final input x^{input}, we concatenate all tokens from $\{\hat{x}^{\text{cot}}_j\}_{j=1}^{k'}$ and place them before x^{question}. Our goal is to optimize the input x^{input} so that LLM can correctly answer the question under x^{input}. Meanwhile, the token count of x^{input}, t, must be less than the LLM context window limit T. Formally, we optimize the following:

$$\min_{\hat{\mathcal{D}}\subseteq\mathcal{D}} L_{\text{LLM}}\left(x^{\text{input}}\right), \max_{\hat{\mathcal{D}}\subseteq\mathcal{D}} R_{\text{Acc}}\left(y_{\text{LLM}}\left(x^{\text{input}}\right), y^{\text{answer}}\right),$$
$$s.t. \quad t\left(x^{\text{input}}\right) = \sum_{1}^{k'} |\hat{x}^{\text{cot}}| + |x^{\text{question}}| \leq T \tag{2}$$

where L_{LLM} is LLM loss, and R_{Acc} evaluates if LLM's answer $y_{\text{LLM}}(x^{\text{input}})$ matches the correct answer y^{answer}, this will be elaborated in Sec. 4.4.

Overview. Fig. 3 illustrates our approach. The core component is a lightweight, plug-and-play module (Sec. 4.3), which consists of a small text embedding extractor and a coarse-to-fine pruner.

To train the pruner, we face the challenge of gradient backpropagation when pruning discrete tokens outside the LLM. The LLM loss gradient cannot be backpropagated through the tokenizer. To address this, we design a multi-objective reward function and use reinforcement learning for effective training (Sec. 4.4). The overall training process is outlined in Algorithm 1.

4.3 Coarse-to-fine Pruner Design

Text embedding extractor. As CoT-Influx serves as an external module, we need to extract text embedding as prediction features. However, it's non-trivial to extract features for long inputs beyond the LLM context window. To address this, we use a small encoder model, BERT-Large (Devlin et al., 2018), to extract sentence-level (i.e., a CoT example) embedding instead of extracting token embedding for the entire long context. For a batch of k CoT examples, each example is padded to N=512 tokens. BERT then inferences these examples to obtain the final layer of text embedding, denoted as $H_{s_{\text{shot}}} \in \mathbb{R}^{k\times N\times D_{\text{BERT}}}$, where D_{BERT} is BERT's hidden dimension size.

State. As shown in Fig. 3, we define state $s_{\text{shot}} \in \mathbb{R}^{k\times N}$ for the first shot pruner, representing the input batch of k CoT examples $\in \hat{\mathcal{D}}$. For the second token pruner, we define state $s_{\text{token}} \in \mathbb{R}^{k'\times N}$, which represents all remaining tokens after the shot pruner. k' is the number of retained examples.

Action. Let a_{shot} and a_{token} denote the actions predicted by the shot and token pruner, respectively. The action space is defined as {0, 1}, where 1 signifies retention and 0 indicates pruning. a_{shot} determines whether each CoT example should be pruned, while a_{token} predicts the pruning of each token in the retained CoT examples.

Two-stage policy network. The pruner module is a two-stage policy network, each stage is a two-layer feed-forward network (MLP) with GELU activa-

Algorithm 1 Pruner Training and Inference

```
Input: target LLM, dataset $\mathcal{D}$, number of CoTs $k$, token limit $T$, manual few-shot cot $x^{\text{few-shot}}$, repeat $t_{\text{repeat}}$
 1: ▶ Phase 1: MRD$^3$ preperation
 2: Perform evolution and difficulty evaluation to get $\mathcal{D}$;
 3: Use the difficulty filter and split $\mathcal{D}$ into questions set $\mathcal{D}_{\text{question}}$ and CoT candidates set $\mathcal{D}_{\text{cot}}$
 4: ▶ Phase 2: Training the two-stage pruner (1 epoch)
 5: for $(x^{\text{question}}, y^{\text{answer}})$ in $\mathcal{D}_{\text{question}}$ do
 6:     Retrieve Top-$k$ examples $\hat{\mathcal{D}} = \{x^{\text{cot}}\}_{i=1}^{k}$ from $\mathcal{D}_{\text{cot}}$
 7:     $H_{s_{\text{shot}}}$ = BERT$\left(\{x^{\text{cot}}\}_{i=1}^{k}\right)$
 8:     for $j$=1 to $t_{\text{repeat}}$ do
 9:         Get $\pi\ (a_{\text{shot}}|s_{\text{shot}};\theta)$ with Eq. 3, sample $a_{\text{shot}}$
10:         $\{x^{\text{cot}}\}_{i=1}^{k} \rightarrow \{x^{\text{cot}}\}_{i=1}^{k'}$
11:         $H_{s_{\text{token}}}$ = BERT$\left(\{x^{\text{cot}}\}_{i=1}^{k'}\right)$
12:         Get $\pi\ (a_{\text{token}}|s_{\text{token}};\theta)$ with Eq. 4, sample $a_{\text{token}}$
13:         $\{x^{\text{cot}}\}_{i=1}^{k'} \rightarrow \{\hat{x}^{\text{cot}}\}_{i=1}^{k'}$
14:         $x^{\text{input}} = \left(\{\hat{x}^{\text{cot}}\}_{i=1}^{k'}, x^{\text{few-shot}}, x^{\text{question}}\right)$
15:         Output LLM($x^{\text{input}}$); Compute $R$ with Eq. 5
16:     end for
17:     Compute policy gradient using Eq. 6, update $\theta$
18: end for
19: ▶ Phase 3: LLM reasoning with pruner and MRD$^3$
20: Retrieve Top-$k$ shots $\{x_q^{\text{cot}}\}^k \in \mathcal{D}$ for target question $q$
21: Do pruning: $\{x_q^{\text{cot}}\}^k \xrightarrow{\theta} \{\hat{x}_q^{\text{cot}}\}^{k'}$, reconstruct $\{\hat{x}_q^{\text{cot}}\}^{k'}$
22: $x_q^{\text{input}} = \left(\{\hat{x}_q^{\text{cot}}\}^{k'}, x^{\text{few-shot}}, x_q^{\text{question}}\right)$
23: Get LLM reasoning output LLM($x_q^{\text{input}}$)
```

tion. This module outputs a continuous categorical distribution π, used for sampling discrete actions (i.e., $\{0, 1\}$). Let θ denote the MLP's trainable parameters and $\sigma(\cdot)$ the sigmoid function. Based on the current states $\{s_{\text{shot}}, s_{\text{shot}}\}$ and the hidden states $\{H_{s_{\text{shot}}}, H_{s_{\text{token}}}\}$, the policy network sequentially make two action predictions as follows:

$$\pi(a_{\text{shot}}|s_{\text{shot}};\theta) = \sigma\left(\text{MLP}\left(H_{s_{\text{shot}}}\right)\right) \tag{3}$$

$$\pi(a_{\text{token}}|s_{\text{token}};\theta) = \sigma\left(\text{MLP}\left(H_{s_{\text{token}}}\right)\right), \tag{4}$$

where a_{shot} and a_{token} are the predicted actions, sequentially predicting whether to prune each of the k CoT examples and each token within the retained examples, respectively. We predict the discrete action by sampling from the categorical distribution π with a softmax function.

4.4 End-to-end RL Optimization

Multi-objective Reward. Our objective in Eq. 2 is to train the pruner module to identify the most crucial CoT examples and useful tokens for math problem-solving while keeping the final tokens within the original LLM context window. To achieve this, we design a multi-objective reward.

Let x^{input} be the final input to LLM, which includes the retained CoT tokens from the policy network and the target question. t represents the token count of x^{input}, and T is the token count limit. The reward R is defined as follows:

$$R\left(x^{\text{input}}\right) = \left(\frac{1}{1+L_{\text{LLM}}} + R_{\text{Acc}}\right) \times \left[\frac{t}{T}\right]^{w} \tag{5}$$

where the first term evaluates the effectiveness of inputted CoT tokens, and the second term ensures they are within the LLM context window. L_{LLM} is LLM's prediction loss under x^{input}, R_{Acc} evaluates the reasoning accuracy (to be discussed later). w is a hyperparameter that penalizes the token count t for being too short (i.e., $w > 0$) or exceeding (i.e.,$w < 0$) the token limit T.

In addition to L_{LLM}, we introduce R_{Acc} to evaluate math reasoning accuracy. This is because L_{LLM}, the average loss of generated tokens, doesn't fully reflect LLM's ability to generate correct answers. Specifically, R_{Acc} is set to 1 for a correct answer and 0 for an incorrect one. Notably, we found that if the format or crucial tokens are pruned, LLM struggles to interpret the input, leading to irrelevant answers for math problem-solving. In such cases, we penalize R_{Acc} with a value of -0.1.

Optimization with REINFORCE. We employ reinforcement learning to maximize the reward and train the two-stage policy network. According to REINFORCE (Williams, 1992), the network parameters are updated by the gradients:

$$R \cdot \nabla_{\theta}\log\pi(a_{\text{shot}}|s_{\text{shot}})\pi(a_{\text{token}}|s_{\text{token}}) \tag{6}$$

Notably, as shown in Fig. 3, only the parameters of the policy network require training. The embedding extractor and LLM are frozen. Thus, the overall training overhead is lightweight.

Difficulty-aware data filter. Existing LLMs, particularly smaller ones, underperform in math reasoning. Suppose the question is too challenging for LLMs. In that case, the answer will always be incorrect, regardless of the quality of compressed few-shot CoT examples, making it challenging to train our pruner module effectively. To address it, we use a difficulty filter to sample a math question set $\mathcal{D}_{\text{question}}$ from $\mathcal{D}$, which includes only easy questions with a difficulty score below a threshold d_{thr}. During training, each question in $\mathcal{D}_{\text{question}}$ samples a batch of k CoT examples from $\mathcal{D}_{\text{cot}}$, where $\mathcal{D}_{\text{cot}}$ is the CoT candidate set sampled from $\mathcal{D}$.

Stabilize the training. Another challenge is that pruning CoT and tokens during training introduces instability, making it difficult for effective training.

First, despite the optimization of question set $\mathcal{D}_{\text{question}}$ through the filter, LLM performance for a randomly sampled question under different few-shot prompts can still be unpredictable. This unpredictability, where a question might yield correct results under low-quality pruned prompts and a complex question might fail under carefully pruned prompts, can affect the pruner's training effectiveness. To address this, we continuously repeat a sampled question multiple times, t_{repeat}, each time with a different pruned few-shot prompt from the pruner. Moreover, we use exponential moving average (EMA) to smooth reward R_{Acc} in Eq. 5.

Second, during the early training, our pruner module makes random decisions, leading to arbitrary removal of CoT examples and tokens. These randomly pruned few-shot prompts can cause instability in RL training. Empirically, we append the manually-designed 8-shot CoTs (Wei et al., 2022) to the pruned prompts. This ensures a good lower bound and stabilizes the training.

5 Evaluation and Analysis

Models, datasets and metric. We evaluate CoT-Influx on LLaMA2-7B, LLaMA2-13B, LLaMA2-70B and two long-context LLMs. The math datasets for evaluation include GSM8K (Cobbe et al., 2021), AddSub (Hosseini et al., 2014), Multi-arith (Roy and Roth, 2015), Svamp (Patel et al., 2021), and Singleeq (Koncel-Kedziorski et al., 2015). We also include MMLU-STEM (Hendrycks et al., 2020) for general reasoning evaluation. We report the Exact Match (EM) accuracy of the predicted answers as the metric. We also include the evaluation of a challenging math dataset MATH (Hendrycks et al., 2021) in Appendix A.

Baselines We set three baselines for comparison:

- *CoT and few-shot CoT prompting*: We compare with widely-used prompts for LLM reasoning, including zero-shot, zero-shot-CoT (Kojima et al., 2022), and the standard manual few-shot-CoT (Wei et al., 2022) with 8 shots.
- *Prompt retrieval*: we also compare with retrieval baselines, specifically using random, TopK (Liu et al., 2021), and BM25 (Robertson et al., 2009) methods. We select as many CoT examples as possible using each method, without exceeding the LLM context window. Random retrieval is to reflect the average quality of our CoT dataset.
- *Prompt compression*: To evaluate the effectiveness of identifying crucial tokens, we compare the resulting compressed prompts from the same batch of CoT shots with state-of-the-art prompt compression baselines: Selective Context (Li et al., 2023b), LLMLingua (Jiang et al., 2023b), and compression through GPT-4.

5.1 Main Results

Effectiveness of enabling more CoT shots. We first evaluate how far the boundary of few-shot learning can be pushed using CoT-Influx. For comparison, we set up two baselines: *(i)* Few-shot CoT, using 8 manual-designed CoT shots. *(ii)* TopK, which retrieves 20 CoT shots from our dataset. For CoT-Influx, we test LLaMA2 7B and 13B on GSM8K, adjusting the number of CoT shots from 16 to 64 examples, which corresponds to 0.7× to 2.8× the token count of LLaMA2 context window.

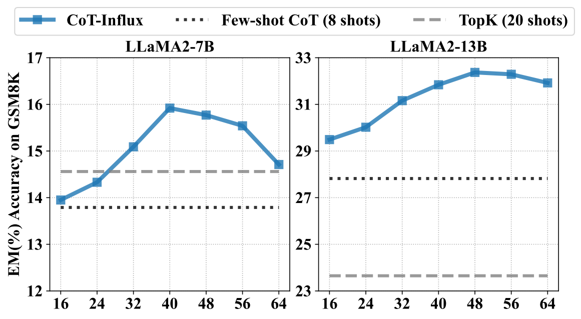


Figure 4: EM(%) accuracy on GSM8K with inputting different number of CoT examples for CoT-Influx.

As shown in Fig. 4, we make two observations: **(1)** More CoT shots, facilitated by CoT-Influx, indeed boosts LLM math reasoning performance, particularly for larger LLMs. On LLaMA2-13B, by inputting 48 CoTs, we significantly outperform the standard few-shot CoT and TopK by 4.55% and 8.72%, respectively. **(2)** There is an optimal number of CoT shots for CoT-Influx. Its peak performance on LLaMA2 7B and 13B are at 40 and 48 shots, respectively. We attribute this to two potential reasons. First, an extremely large number of shots complicates CoT-Influx's optimization. Second, there may be an upper limit to improving LLM reasoning capability through few-shot learning.

Comparison with state-of-the-art baselines. Table 2 and Table 3 present the comparison results of CoT-Influx with state-of-the-art baselines across LLaMA2 family and 5 mathematical datasets, highlighting the following observations: **(1)** by utilizing more few-shot CoTs that are twice the LLM context window, CoT-Influx significantly outperforms all baselines, with up to 4.55% absolute improvements. **(2)** Despite using fewer input tokens, CoT-Influx consistently outperforms retrieval baselines by 1.36% to 14.09% absolute improvements. This is because our compressed tokens indicate more

Table 2: Comparison of EM (%) accuracy on GSM8K with state-of-the-art baselines. Note that the 20 CoT shots of retrieval baselines are the max number, given that the context window limit of LLaMA2 is 4096 tokens.

Method	#Input CoT shots	#Average tokens	LLaMA2-7B	LLaMA2-13B	LLaMA2-70B
Zero-shot	0	-	4.25	5.84	11.45
Zero-shot-CoT (Kojima et al., 2022)	0	-	1.74	12.28	21.91
Few-shot-CoT (Wei et al., 2022)	8	655	13.79	27.82	55.42
Random retrieval	20	3379.8	12.51	22.21	53.07
TopK retrieval (Liu et al., 2021)	20	3535.4	14.56	23.65	54.59
BM25 retrieval (Zhenyu et al., 2023)	20	3816.1	13.42	25.17	54.21
TopK+GPT4 Compression	40	1376.0	7.08	11.01	25.17
TopK+Selective Context (Li et al., 2023b)	40	2262.4	0.45	0.76	2.50
TopK+LLMLingua (Jiang et al., 2023b)	40	2048.0	5.38	8.34	22.74
CoT-Influx	48	2037.0	**15.92 (↑1.36)**	**32.37 (↑4.55)**	**59.59 (↑4.17)**

Table 3: Comparison of EM (%) accuracy on Addsub, Multiarith, Svamp, and Singleeq math reasoning dataset

Model	Method	AddSub	Multiarith	Svamp	Singleeq	Avg.
	Zero-shot	58.73	5.50	32.2	62.79	39.81
	Few-shot-CoT	56.96	43.67	38.1	66.54	51.32
LLaMA2-7B	TopK retrieval	46.08	34.50	38.1	46.46	41.29
	TopK+LLMLingua	12.91	10.50	19.5	19.49	15.60
	CoT-Influx	62.28	47.00	40.2	72.05	**55.38**
	Zero-shot	70.13	6.50	43.8	71.07	47.88
	Few-shot-CoT	65.82	72.83	42.7	77.36	64.68
LLaMA2-13B	TopK retrieval	60.76	57.00	50.2	68.50	59.12
	TopK+LLMLingua	22.28	22.33	27.5	25.20	24.33
	CoT-Influx	69.62	73.87	50.5	83.07	**69.26**

Table 4: Comparison of EM (%) accuracy on GSM8K with larger LLMs under the few-shot-CoT setting.

Model	Parameters	EM (%)
Finetuned GPT-3 (Wei et al., 2022)	175B	34.0
Chinchilla (Hoffmann et al., 2022)	70B	43.7
Text-davinci-002 (Kojima et al., 2022)	175B	51.5
PaLM (Chowdhery et al., 2022)	540B	56.5
GPT-3.5 (OpenAI, 2023a)	-	57.1
Minerva (Lewkowycz et al., 2022)	540B	58.8
LLaMA2-70B+CoT-Influx	70B	**59.6**

informational CoT examples without redundancy. In contrast, they select entire examples, which may contain redundant tokens. **(3)** CoT-Influx significantly outperforms prompt compression baselines in preserving the most crucial tokens for math reasoning, while methods like Selective Context and LLMLingua suffer accuracy declines due to difficulties in maintaining few-shot prompt structure. GPT-4 tends to prune essential reasoning steps, which negatively impacts CoT effectiveness.

We further demonstrate the effectiveness of CoT-Influx by comparing LLaMA2-70B with larger-size LLMs on GSM8K. As shown in Table 4, CoT-Influx significantly boosts LLM reasoning capabilities. Remarkably, without any fine-tuning, LLaMA2-70B with CoT-Influx outperforms much larger LLMs such as GPT-3.5.

Compatible with existing reasoning prompts. As a method to improve LLM reasoning capability, CoT-Influx is complementary with other advanced

Table 5: CoT-Influx is compatible with advanced prompt techniques like self-consistency (i.e., maj@20) and self-verification (i.e., verify@20).

Method	LLaMA2-13B	LLaMA2-70B
CoT-Influx	32.37	59.59
CoT-Influx+maj@20	**33.43 (↑1.06)**	**60.73 (↑1.14)**
CoT-Influx+verify@20	**34.04 (↑1.67)**	**61.79 (↑2.20)**

reasoning-based prompts. To prove this, we apply self-consistency (Wang et al., 2023d) and self-verification (Weng et al., 2023) to the prompts generated by CoT-influx. For evaluation efficiency, we sampled 20 times. As Table 5 shows, applying self-consistency and self-verification further improves LLaMA2's performance on GSM8k.

Generalization to other reasoning tasks. To validate the generalizability of CoT-Influx on general reasoning tasks beyond math, we further verify our method on an additional benchmark: MMLU-STEM (Hendrycks et al., 2020). As shown in Table 6, despite CoT-Influx being trained on MRD[3] with only math data, it still improves commonsense reasoning performance over various baselines. We believe that by integrating CoT datasets for more tasks, our CoT-Influx can achieve superior reasoning performance. We leave this as future work.

Table 6: Comparison of accuracy on MMLU-STEM benchmark with state-of-the-art baselines.

Method	#Input shots	LLaMA2-7B	LLaMA2-13B
Few-shot	5	36.4	44.1
TopK retrieval	5+20	35.7	43.9
TopK+LLMLingua	5+40	34.2	43.3
CoT-Influx	5+40	**37.0** (↑0.6)	**44.3** (↑0.2)

5.2 CoT-Influx on Long Context LLMs

Recently, an increasing number of long-context LLMs (context length≥32K) have emerged to address tasks involving extensive input contexts. These models naturally facilitate handling as many few-shot examples as possible. However, researchers have pointed out that scaling the few-shot examples does not consistently improve the in-context learning performance (Zhao et al., 2024; Li et al., 2024), and most long-context tasks can be solved by short-context input (Qian et al., 2024).

We verify these observations and further demonstrate that CoT-Influx is highly beneficial to long-context LLMs by selecting high-quality, concise CoT examples. Moreover, CoT-Influx can serve as a **prompt compressor** for long-context LLMs, saving inference costs by pruning redundant input tokens. Fig. 5 shows the results on Mistral-7B-v0.3-32K (Jiang et al., 2023a) and Yi-6B-200K (AI et al., 2024). While more CoT examples may not consistently improve accuracy, CoT-Influx significantly outperforms current few-shot and prompt retrieval baselines. With an average 72.9% and 86.4% input token reduction in the prompt, we can achieve 2.7% and 2.5% absolute improvement on GSM8K, respectively.

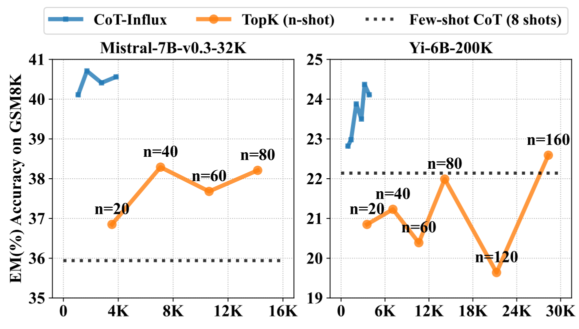


Figure 5: Prompt compression effect on long-context LLMs. The x-axis indicates the number of input tokens.

5.3 Ablation Study and Analysis

Ablation study on coarse-to-fine pruner. Our pruner operates at both shot and token levels to fully exploit redundancy within CoT examples. To verify the effectiveness, we conducted experiments with only a shot or token pruner in the same setting. As shown in Table 7, removing any pruning stage decreases performance. Notably, removing token-only pruning causes a larger accuracy drop than shot-only pruning, indicating that shot-level redundancy is easier for the pruner to learn.

Table 7: Comparison of different pruning strategies.

Pruning Strategy	LLaMA2-7B	LLaMA2-13B	LLaMA2-70B
CoT-Influx (shot+token)	**15.92**	**32.37**	**59.59**
Prune shot only	15.69 (↓0.23)	31.08 (↓1.29)	57.77 (↓1.82)
Prune token only	12.05 (↓3.87)	25.32 (↓7.05)	49.36 (↓10.23)

Token pruning ratios. We now investigate token pruning ratios by our pruner. Fig. 6 shows the remaining token length for LLaMA2-70B after our pruner. In total, we achieve a 4.28× pruning ratio, with the shot pruner contributing a 3.87× ratio. The results suggest that our pruner favors pruning more coarse-grained shots over fine-grained tokens.

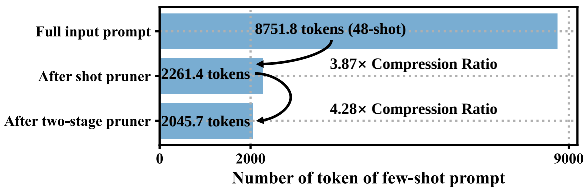


Figure 6: Token length after each stage of our pruner.

Inference cost. CoT-Influx is a lightweight plug-and-play module, including a 336MB BERT-Large model and a tiny 4MB coarse-to-fine pruner. We measure its additional inference cost. Table 8 shows the total inference latency and GPU memory required to run LLaMA2-7B with different methods on GSM8K, measured on a single NVIDIA A100 GPU. The results reveal that CoT-Influx introduces a negligible 1.4GB additional memory and a 1.7% increase in latency. This is more effective than prompt compression baselines, such as Selective Context and LLMLingua, which require significantly higher latency and more GPU memory, potentially hindering efficient deployment.

Table 8: The total inference costs on GSM8K.

Method	#Input-shot	#Token	Time	GPU Memory
LLaMA2-7B	12	2108.6	2.99h	19.7GB
Selective Context	40	2262.4	4.38h	23.5GB
LLMLingua	40	2048.0	3.65h	33.0GB
CoT-Influx	40	2037.0	3.04h	21.1GB

Implications. Our analysis of retained CoT examples and tokens yields the following insights: **(1)** More capable LLMs favor harder CoT examples, while smaller LLMs opt for simpler ones. **(2)** Numerical and format tokens are essential for math reasoning. Function words like *with*, *the*, *then*, and irrelevant background context can be pruned without affecting reasoning capability.

6 Conclusion

We present CoT-Influx, a plug-and-play module that improves LLM math reasoning by pruning unnecessary few-shot examples at shot and token levels for a more effective input context. To train the module, we use reinforcement learning to optimize a math reasoning-specific reward. Extensive experiments on various datasets and LLMs compared with state-of-the-art baselines demonstrate the effectiveness of our method. This paper highlights the vast potential of few-shot CoT prompting in augmenting LLMs' math reasoning abilities.

Limitations

As in-context learning with LLM heavily relies on the selected examples in the prompt, the performance of CoT-Influx can be influenced by the quality of CoT generation. Despite this, CoT-Influx still demonstrates strong performance on our GPT4-evolved dataset MRD^3. We currently use BERT to obtain the feature embedding of a CoT example, which cannot handle long-sequence examples exceeding 512 tokens. We will take these limitations into account and mitigate them in future work.

References

01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. 2024. Yi: Open foundation models by 01.ai.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023b. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

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

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Everything of thoughts: Defying the law of penrose triangle for thought generation. *arXiv preprint arXiv:2311.04254*.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. 2023. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance. *arXiv preprint arXiv:2305.17306*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2021*, pages 3816–3830. Association for Computational Linguistics (ACL).

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*.

Yaru Hao, Yutao Sun, Li Dong, Zhixiong Han, Yuxian Gu, and Furu Wei. 2022. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

SU Hongjin, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. 2022. Selective annotation makes language models better few-shot learners. In *The Eleventh International Conference on Learning Representations*.

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.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 784–794. Association for Computing Machinery.

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

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.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Junyan Li, Li Lyna Zhang, Jiahang Xu, Yujing Wang, Shaoguang Yan, Yunqing Xia, Yuqing Yang, Ting Cao, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2023a. Constraint-aware and ranking-distilled token pruning for efficient transformer inference. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 1280–1290.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023b. Compressing context to enhance inference efficiency of large language models.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*.

OpenAI. 2023a. Gpt-4 technical report.

OpenAI. 2023b. Welcome to the openai platform.

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.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023a. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023b. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Yujia Zhou, Xu Chen, and Zhicheng Dou. 2024. Are long-llms a necessity for long-context tasks? *arXiv preprint arXiv:2405.15318*.

Chengwei Qin, Aston Zhang, Anirudh Dagar, and Wenming Ye. 2023. In-context learning with iterative demonstration selection. *arXiv preprint arXiv:2310.09881*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

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.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Alexander Scarlatos and Andrew Lan. 2023. Reticl: Sequential retrieval of in-context examples with reinforcement learning. *arXiv preprint arXiv:2305.14502*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

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*.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2023. Focused transformer: Contrastive training for context scaling.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160*.

Liang Wang, Nan Yang, and Furu Wei. 2023b. Learning to retrieve in-context examples for large language models. *arXiv preprint arXiv:2307.07164*.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023c. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023d. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, Song Han, and Maosong Sun. 2024. Infllm: Unveiling the intrinsic capacity of llms for understanding extremely long sequences with training-free memory. *arXiv preprint arXiv:2402.04617*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Kang Min Yoo, Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang-goo Lee, and Taeuk Kim. 2022. Ground-truth labels matter: A deeper look into input-label demonstrations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2437.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2024. Is in-context learning sufficient for instruction following in llms? *arXiv preprint arXiv:2405.19874*.

Wu Zhenyu, Wang Yaoxiang, Ye Jiacheng, Feng Jiangtao, Xu Jingjing, Qiao Yu, and Wu Zhiyong. 2023. Openicl: An open-source framework for in-context learning. *arXiv preprint arXiv:2303.02913*.

Appendix

This appendix includes additional analysis, the evolution of MRD^3, pruner training details, additional related works, and prompt settings. These contents are organized in separate sections as follows:

- Sec. A provides additional analysis and case studies, including the comparison of CoT-Influx with context window extension methods, the performance of CoT-Influx on instruction-finetuned LLMs (LLaMA2-13B-Chat and GPT-3.5-Turbo), ablation study on the reward design, and sensitivity analysis on hyperparameters of the pruner. The effectiveness of CoT-Influx on a more difficult dataset MATH is verified. Additional case studies on the GSM8K with different prompting methods are given to prove the effectiveness of our method.
- Sec. B introduces the prompt we used for the evolution of the examples in our MRD^3. Both the original input and the evolution results are given as examples. We also analyze the difficulty and reasoning step distribution of different evolution methods and derive a new observation regarding difficulty preference for different LLMs.
- Sec. C includes the algorithm for training data preparation as a supplement to Algorithm 1. The hyperparameter settings, the training dynamic of the pruner, and the detailed introduction of the evaluation dataset are also included.
- Sec. D elaborates previous LLM context window extension and LLM in-context learning methods and analyzes the advantage of our proposed CoT-Influx compared with various previous methods.
- Sec. E demonstrates the prompt we used in this work for difficulty and reasoning step evaluation, and GPT-4 based compression on input few-shot prompts.

A Additional Analysis and Case Study

A.1 Comparison with context window extension methods

While our work tackles the challenge of the limited context window by pruning the redundant input few-shot prompts, another solution is to extend the context window of LLMs. We compare the math reasoning performance of LLaMA2-7B with CoT-Influx and LLaMA2-7B with 32K token context window extended with Positional Interpolation (PI) (Chen et al., 2023a). The results are listed in Table 9.

Table 9: Comparsion of EM(%) on GSM8K of LLaMA2-7B with CoT-Influx and LLaMA2-7B-32K with PI.

Number of input shots	12	16	20	24	28	32	40
Average number of tokens	2108.6	2820.6	3535.4	4217.2	4929.1	5641.2	7070.8
LLaMA2-7B	13.87	15.08	14.02	-	-	-	-
LLaMA2-7B+CoT-Influx	-	-	-	14.33	15.09	15.92	15.77
LLaMA2-7B-32K	11.37	12.81	11.37	11.83	11.83	11.52	11.30

When the input prompt does not exceed the window token limit (the number of input shots is not more than 20), we compare the performance of LLaMA2-7B-32K with LLaMA2-7B. When the input prompt exceeds the context window length, we apply our CoT-Influx to prune the prompts to make sure that they can be directly input to LLaMA2-7B without PI. The results show that the context window extension weakens reasoning ability when using the same input prompt. The limit of the context window can be unlocked with our CoT-Influx. Moreover, our observation that LLMs can improve reasoning with more helpful CoT examples does not hold true for LLMs with extended context windows, which is also discussed in recent research (Li et al., 2024).

Another drawback of the existing context window extension method lies in the efficiency of fine-tuning models with the extended context. Compared to the previously proposed methods such as PI (Li et al., 2024), YaRN (Peng et al., 2023a), and LongRoPE (Ding et al., 2024), as can be seen from the Table 10

Table 10: Comparsion of Efficiency of CoT-Influx and common context extension methods.

Method	PI (Li et al., 2024)	YaRN (Peng et al., 2023a)	LongRoPE (Ding et al., 2024)	**CoT-Influx**
Devices for Training	128 A100 GPUs	16 A100 GPUs	16 A100 GPUs	**1 A100 GPU**
Training Time	unknown	unknown	2 weeks	**3 hours**

which compares the number of GPUs and the training time, our CoT-Influx is significantly more efficient in terms of the training cost.

Recently, researchers proposed a training-free LLM context window extension method InfLLM (Xiao et al., 2024), which improves the efficiency of attention computation by storing distant contexts into additional memory units and using dynamic multi-step memory lookup. While this work solves the problem of expensive finetuning costs to some extent, significant inference overhead still exists for inputting the long context to the LLM and additional operations such as the memory lookup. In addition to the computation overhead, the results in our paper also suggest that removing informative tokens in prompts can also improve reasoning performance. However, context extension methods keep the original prompt as the input and also retain these "harmful tokens". We believe this is another important advantage of our method compared to all context extension methods, including InfLLM (Xiao et al., 2024).

A.2 CoT-Influx on instruction-finetuned LLMs

In Sec. 5.1, we verify the effectiveness of CoT-Influx on LLaMA2-7B, 13B, and 70B. LLaMA2-chat (Touvron et al., 2023) and GPT-3.5-Turbo (OpenAI, 2023b) are also the widely adopted LLMs that are acquired after supervised instruction finetuning (SIFT) and Reinforcement Learning from Human Feedback (RLHF), respectively. The different finetuning strategies and the various finetuning data result in unique properties of the LLMs. For example, LLaMA2-Chat-13B performs significantly better than LLaMA2-13B on math reasoning tasks with zero-shot-cot prompts. To show that our CoT-Influx can also help improve the reasoning ability of these finetuned LLMs, we conduct experiments of LLaMA2-13B-Chat and GPT-3.5-Turbo (`gpt35-turbo-0613`) on GSM8K dataset. As shown from the results listed in Table 11, our CoT-Influx also surpasses a wide range of prompting baselines with more input shots and fewer tokens. Specifically on LLaMA2-13B-Chat, CoT-Influx achieve an absolute improvement of 9.78% compared to the TopK retrieval baseline with only 57.6% average tokens.

Table 11: The EM (%) accuracy on GSM8K with CoT-Influx and other baselines. Note that the context window limit of LLaMA2-13B-Chat and GPT-3.5-Turbo are all 4096 tokens.

Method	#Input CoT shots	#Average tokens	LLaMA2-13B-Chat	GPT-3.5-Turbo
Few-shot-CoT (Fu et al., 2023)	8	655	27.82	72.55
TopK retrieval (Liu et al., 2021)	20	3535.4	31.16	70.74
TopK+LLMLingua (Jiang et al., 2023b)	40	2048.0	10.69	49.96
CoT-Influx	48	2037.0	**40.94**	**73.31**

A.3 Ablation study on reward design

The reward of our CoT-Influx pruner is made up of three parts: math reasoning accuracy reward R_{Acc}, LLM loss reward $R_{\text{Loss}} = \frac{1}{1+L_{\text{LLM}}}$, and context window token limit reward $R_{\text{Limit}} = \left[\frac{t}{T}\right]^{w}$. Each part of the full reward function is important for the effective learning of the pruner. We perform ablation studies on these components and the results are listed in Table 12. As can be seen from the results, whenever a reward component is removed, the CoT-Influx pruner gives sub-optimal prompt selection and compression results, which cause a decrease compared to the full reward baseline. Among these three reward function parts, the token limit reward R_{Limit} is the most important because training without this term will cause the pruner **not** to prune any shot or token and directly output the truncated prompt of the redundant input.

Table 12: The EM (%) accuracy on GSM8K of LLaMA2-7B and LLaMA2-13B with different reward function.

Reward Function	LLaMA-2-7B	LLaMA-2-13B
Full Reward	15.92	32.37
w/o R_{Acc}	15.24	31.46
w/o R_{Loss}	14.78	31.16
w/o R_{Limit}	14.25	29.72

A.4 Sensitivity analysis on hyperparameters and training settings

We perform sensitivity analysis on the hyperparameters to investigate the robustness of our CoT-Influx pruner training. The most important setting in the training of our CoT-Influx pruner is the token target T, token penalty co-efficient w, and the reward penalty value in R_{Acc}. Table 13 and Table 13 present the results of CoT-Influx using different sets of hyperparameters T, w, and reward penalty in R_{Acc}. The results demonstrate that the training of our CoT-Influx pruner is highly robust as long as the token target T is not overly aggressive (token target T should not be too small). We also empirically set the value of the reward penalty in R_{Acc} as -0.1 based on the experiments.

Table 13: Sensitivity analysis on token target T and token penalty co-efficient w

Token target T	LLaMA-2-13B
2048	32.37
1024	29.57
3072	32.37

Token penalty co-efficient w	LLaMA-2-13B
(-1,1)	32.37
(-0.5,1)	31.69
(-1,0.5)	32.22

Table 14: Sensitivity analysis on reward penalty value in R_{Acc}

Reward Penalty in R_{Acc}	0	-0.05	-0.1	-0.2	-0.5
LLaMA2-13B@GSM8K	31.69	32.37	32.37	32.07	31.92

We have also verified the effect of applying different LLMs for the training of the CoT-Influx pruner. The results are listed in Table 15. Based on our observations, the choices of LLM during training will not significantly influence the pruning capability of CoT-Influx, as the performance of CoT-Influx+LLaMA2-7B on GSM8K with different training LLM is close.

A.5 CoT-Influx for more difficult math reasoning tasks

As described in the abstract, introduction, and analysis, the proposed dataset MRD^3 and reward function design of CoT-Influx are tailored for grade-school-level math reasoning tasks. To explore the generalizability of CoT-Influx to higher-difficulty math reasoning problems, we further verify our method on one additional benchmark MATH (Hendrycks et al., 2021) consisting of 12,500 challenging competition mathematics problems, covering algebra, calculus, statistics, geometry, linear algebra, and number theory. We directly apply our CoT-Influx pruner trained on MRD^3 to optimize the prompt for MATH evaluation. The results are shown in Table. 16. We notice that the improvement of CoT-Influx is less significant compared to GSM8K, mainly because reasoning on MATH is more difficult, and the average difficulty CoT candidates in MRD^3 shown in Figure 7 is closer to GSM8K instead of MATH.

A.6 Case Study on different prompt compression methods

To show how different prompt compression methods prune input few-shot prompts in different manners, we give an example of an 8-shot prompt selected using the TopK retriever. The original full few-shot prompts are listed in the following box:

Table 15: Comparison of different choices on the LLMs used for training the CoT-Influx pruner.

LLM used for training	LLaMA2-7B	LLaMA2-13B	LLaMA2-70B
CoT-Influx+LLaMA2-7B	15.77	15.92	15.85

Table 16: Comparison of accuracy on MATH dataset with state-of-the-art baselines.

Method	#Input CoT shots	LLaMA2-7B	LLaMA2-13B	LLaMA2-70B
Zero-shot	0	2.8	3.9	13.5
Few-shot-CoT (Wei et al., 2022)	8	7.7	11.5	24.1
TopK retrieval (Liu et al., 2021)	20	9.1	13.5	24.5
TopK+LLMLingua (Jiang et al., 2023b)	40	6.6	10.3	19.9
CoT-Influx	48	**9.3** (↑0.2)	**14.0** (↑0.5)	**25.2** (↑0.7)

Original full few-shot prompt for math reasoning (1331 tokens):

Q: Dave won 11 tickets at the arcade and spent 5 on a beanie. Afterward, he won 10 more tickets. Calculate his final ticket count by first finding the remaining tickets after his purchase and then adding the newly won tickets.
A: Let's think step by step. Dave had 11 tickets, spent 5, leaving him with 6. Then he won 10 more, resulting in: 6 + 10 = 16 tickets. The answer is 16.

Q: At the carnival, tickets for rides cost 0.75 dollars each, or you can buy a 15-dollar armband for unlimited rides for one night. To determine the number of rides where the armband's cost equals that of individual tickets, set up and solve an equation involving x, the number of rides.
A: Let's think step by step. Let x be the number of rides. Equate the cost of x rides using individual tickets, 0.75x dollars, to the 15-dollar armband cost: 0.75x = 15. Solve for x: x = 15/0.75, which gives x = 20. The answer is 20.

Q: Mitch, Jam, and Jay went out for a movie. Mitch paid $7 per ticket for 3 friends, Jam purchased 2 popcorn boxes at $1.5 each, and Jay got 3 milk teas for $3 each. To equitably split the expenses, how much should each of them contribute?
A: Let's think step by step. The total cost of 3 tickets at $7 each, 2 popcorn boxes at $1.5 each, and 3 milk teas at $3 each is $21 + $3 + $9 = $33. Dividing the overall expenses among 3 friends results in a contribution of $33/3 = $11 per person. The answer is $11.

Q: Connor is taking his date to the movies, with tickets costing $10.00 each. They opt for the large popcorn & 2 drink combo meal at $11.00, and each choose a box of candy at $2.50 per box. Determine the combined expenses for the movie tickets, combo meal, and candy to find the total amount Connor will spend on his date.
A: Let's think step by step. Calculate the cost of two movie tickets (2 x $10.00 = $20.00), the combo meal ($11.00), and two boxes of candy (2 x $2.50 = $5.00), then sum them up ($20.00 + $11.00 + $5.00 = $36.00). The answer is $36.00.

Q: Scott has 4 tickets. Ernest starts with 9 tickets and later discovers a stash of 72 more. Calculate the final number of tickets Ernest possesses.
A: Let's think step by step. Ernest initially holds 9 tickets and acquires 72 additional ones, leading to a total of 9 + 72 = 81 tickets. The answer is 81.

Q: Joseph and his friends watched two movies at his place. The first movie lasts 1 hour and 30 minutes, and the second is 30 minutes longer. They took 10 minutes for popcorn and double that for fries. Determine, in hours, the cumulative time spent cooking and watching movies by breaking down each component of time spent.
A: Let's think step by step. First, find the second movie's length (1 hour and 30 minutes + 30 minutes = 2 hours). Then, sum both movies' lengths (1 hour and 30 minutes + 2 hours = 3 hours and 30 minutes). Next, calculate cooking time (10 minutes for popcorn + 20 minutes for fries = 30 minutes). Lastly, add movie and cooking times (3 hours and 30 minutes + 30 minutes = 4 hours). The answer is 4 hours.

Q: The movie theater sold a number of tickets to the horror and romance movies. The horror movie ticket sales were 18 more than three times the romance movie ticket sales. If there were 25 romance movie tickets sold, how many tickets were sold for the horror movie, considering the given relationship?
A: Let's think step by step. Let "h" represent the horror movie tickets sold. Given that h = 3(25) + 18, we can simplify the equation: h = 75 + 18, resulting in h = 93. The answer is 93.

Q: On Saturday, Sara purchased 2 movie theater tickets at $10.62 each, rented a movie for $1.59, and bought another movie for $13.95. Determine Sara's total expenditure on movies by performing a step-by-step calculation.
A: Let's think step by step. Firstly, calculate the movie tickets' cost by multiplying the ticket price ($10.62) by the quantity (2), resulting in $21.24. Secondly, combine the rental ($1.59) and purchase ($13.95) costs, equaling $15.54. Lastly, sum the ticket cost and rental/purchase cost: $21.24 + $15.54 = $36.78. The answer is $36.78.

Most of the examples above have similar backgrounds and targets (tickets, movies, expenses, etc.), but the difficulty and number of reasoning steps are various. In addition, the solutions to most questions are highly redundant. When performing math reasoning, it is important to select the most suitable and concise examples considering the characteristics of the current input question. In our evaluation across different methods shown in Sec. 5.1, we have empirically observed the previous methods fail to retain the structural integrity of the prompt. We show the pruned prompt with different methods and similar token lengths in the following box. We can see that Selective Context and LLMLingua frequently discard the important part including 'Q:', 'A:', '\n', "Let's think step by step", and "The answer is" in these examples. Although GPT-4 can retain a majority of these tokens, the reasoning steps are systematically removed because GPT-4 cannot be instructed to utilize the redundancy in both example-level and token-level. Our proposed CoT-Influx, however, selects the most representative examples and only removes the redundant function words.

Pruned few-shot prompt of different methods:

Selective Context:
Q Dave won 11 tickets Afterward won: step Dave 11 tickets spent leaving Then won 10 resulting: 16 Q At tickets rides rides where set solve x: step Let x rides Equate x rides individual tickets dollars = x 20 Q Mitch Jam went paid per 3 friends Jam purchased equitably how: step 3 tickets + 3 friends results $ Q Connor tickets They opt the large popcorn & 2 drink combo meal choose candy combo meal candy Connor: step combo boxes sum $ Q Scott 4 tickets starts 9 tickets discovers 72 Ernest possesses: step initially holds 9 tickets 72 additional ones leading 81 Q Joseph watched lasts They popcorn double hours cooking breaking: step First find + Then sum both movies' lengths + Next, calculate cooking time popcorn + Lastly add movie cooking times + 4 hours Q sold 25 romance movie tickets considering the given relationship: step Let "h the horror movie tickets Given = 18 simplify 75 93 Q Sara purchased rented movies performing: step Firstly calculate resulting Secondly combine rental Lastly sum $

LLMLingua:
: Dave won11ets the and5 a be. After he. his final count by first theets after the: Lets think. Daveets5,, in. : the,ets 5, or a-ollarides for one. To theidesband cost equals of, equation involving r. A: think. Let.ides using individualets, the1ollar cost5 which. :, Jam and Jay a7 ticket3 Jam2orn5 Jay3 milk. To equ the.ets boxes53 milk each1. the overallenses3 friends a. The : Connor is his,.. They theorn & drinkbo and0. theandy think. ofets0 theboal and two then :. Ernest and later a7. think. Ernest initially and, 9: friends at movie the minutes They and for. the spent by think, the, calculate The a the and ticket, think.:, bought.by-step calculation. A: Let's think step by step. Firstly, calculate the movie tickets' cost by multiplying the ticket price ($10.62) by the quantity (2), resulting in $21.24. Secondly, combine the rental ($1.59) and purchase ($13.95) costs, equaling $15.54. Lastly, sum the ticket cost and rental/purchase cost: $21.24 + $15.54 = $36.78. The answer is $36.78.

GPT-4 Compression:
Q: Dave won 11, spent 5 and won 10 more. Determine final count.
A: The answer is 16.
Q: Tickets cost 0.75 per ride, armband cost 15. Determine rides that armband's cost equals tickets.
A: The answer is 20.
Q: $7 per ticket for 3, 2 popcorn boxes at $1.5, 3 milk teas for $3. Determine each contribute.
A: The answer is $11.
Q: Tickets cost $10.00 each, meal cost $11.00, a box of candy at $2.50. Determine the expenses.
A: The answer is $36.00.
Q: Scott has 4. Ernest starts with 9 and discovers 72 more. Determine the final number.
A: The answer is 81.
Q: The first 1.5 hour, the second is 30 minutes longer. 10 minutes for popcorn. Determine the time.
A: The answer is 4 hours.
Q: Horror movie were 18 more than 3 times romance. 25 romance movie sold, Determine number of horror movie.
A: The answer is 93.
Q: Sara purchased 2 at $10.62 each, a movie for $1.59, and another $13.95. Determine total expenditure.
A: The answer is $36.78.

CoT-Influx:
Q: Mitch, Jam, and went out a. Mitch paid $7 per ticket for 3, Jam purchased 2 boxes at $1.5 each, and got 3 for $3 each. To equitably split, how much should each them contribute?
A: Let's think step by step. The total cost 3 tickets $7 each, 2 popcorn boxes $1.5 each, and 3 milk $3 each is $21 + $3 + $9 = $33. Dividing the overall expenses among 3 results of $33/3 = $11 per. The answer is $11.

Q: The theater sold number tickets to horror and romance movies. The horror movie ticket sales were 18 more than three times romance ticket. If there 25 romance sold, how many tickets were sold horror movie, considering?
A: Let's think step by step. Let "h" represent horror tickets sold. Given h = 3(25) + 18, we can simplify equation: h = 75 + 18, resulting h = . The answer is 93.

Q: On, Sara purchased 2 theater tickets $10.62 each, rented movie $1.59, and bought movie $13.95. Determine Sara's total expenditure movies performing a calculation.
A: Let's think step by step. , calculate tickets' cost price ($10.62) by quantity (2), resulting $21.24. Secondly, combine rental ($1.59) purchase ($13.95), equaling. Lastly, sum ticket rental/purchase: $21.24 + $15.54. The answer is $36.78.

B Evolution of MRD^3

B.1 Prompt template for evolution

The prompt we used for the evolution of the examples in our dataset is listed as follows:

Prompt for different evolution strategies

I want you to act as a Prompt Rewriter. Your objective is to rewrite a given prompt into a more complex version to make those famous AI systems (e.g., LLaMA, ChatGPT, and GPT4) a bit harder to handle.
The prompt is made up of a math reasoning question and the corresponding answer.
The rewritten prompt must be reasonable and must be understood and responded by humans.
Your rewriting cannot omit or change the input and results in #Given Prompt#. Also, please retain the format of 'Question: ' and 'Answer: ' in your response.
You SHOULD complicate the given prompt using the following method:
{Evolution template}
You should try your best not to make the #Rewritten Prompt# become verbose, #Rewritten Prompt# can only add 10 to 20 words into #Given Prompt#.
The #Rewritten Prompt# should also follow the format that the rewritten question appears after 'Question: ' and the rewritten answer appears after 'Answer: '.
The rewritten answer should end up with 'The answer is [results]'.
#Given Prompt#:
Question: **{Given question}**
Answer: **{Given answer}**
#Rewritten Prompt#:

Evolution template for evolution strategy add_constraints:
Please add one more constraint/requirement to the question of #Given Prompt#

Evolution template for evolution strategy deepening:
Please increase the depth and breadth of the question and answer of #Given Prompt#

Evolution template for evolution strategy increase_reasoning:
If #Given Prompt# can be solved with just a few simple thinking processes, please rewrite it to explicitly request multiple-step reasoning.

Evolution template for evolution strategy revise_difficulty:
Please revise the high-difficulty questions to lower difficulty.

Evolution template for evolution strategy produce_easier:
Please produce a new and easier question with another different topic.

Most parts of the prompt of different evolution strategies are similar. Based on our quantitative analysis of the difficulty and reasoning step distribution, GPT-4 can effectively follow our instructions to modify the constraints or difficulty level of input questions.

B.2 Difficulty and Reasoning Steps Distribution of MRD^3

Based on the GPT-4-based estimation, we are able to quantitatively look into the distribution of difficulty and reasoning step distribution in MRD^3 without evolution and MRD^3 with various evolution schemes. The results are shown in Figure 7. The original distribution of both difficulty level and reasoning steps of questions centralized between 2 and 4. More questions with higher difficulty using add_constraints, deepening, and increase_reasoning. As we discuss in the reward design of our RL pruner, easy questions are important for the stabilization of RL and can help effectively identify the quality of pruned prompts. Easier questions are generated with revise_difficulty and produce_easier evolution scheme.

B.3 Additional observation on difficulty distribution

As shown in Figure 7, the difficulty diversity of examples in MRD^3 is improved after prompt evolution. We then research the difficulty distribution of the input examples for in-context learning. The observation is shown as follows in addition to the 3 main observations proposed in Sec. 3:

Observation 4: *LLMs with different capabilities prefer CoT examples of varying difficulties.*

In our further exploration of the optimal selection of CoT examples for improved mathematical reasoning, we observe that LLMs of different capabilities exhibit preferences for CoT examples of varying difficulty levels. As Table 17 shows, we categorize each CoT example in the MRD^3-Evol dataset by difficulty level. We then select the top 16 CoT examples from different groups as few-shot examples for LLaMA2 models. Results show LLaMA2-7b prefers CoT examples with a difficulty level of 3-4, while

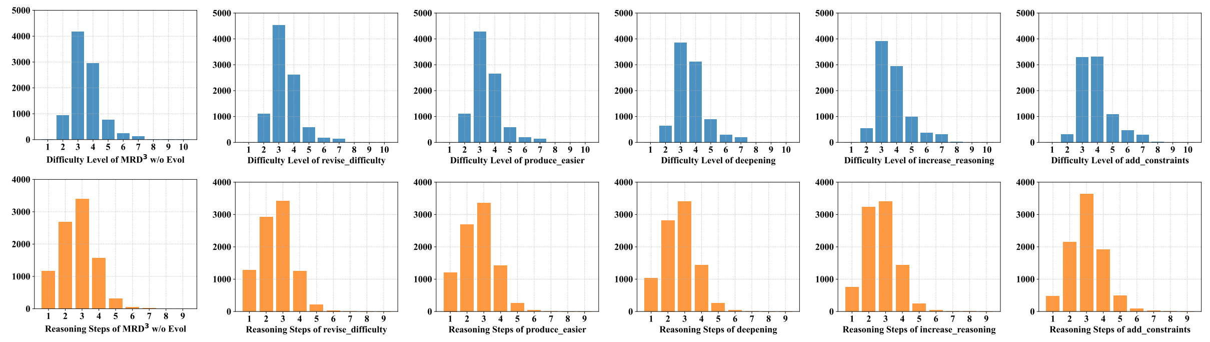


Figure 7: The difficulty distribution (first row) and the number of reasoning steps distribution (second row).

LLaMA2-13b, more capable, prefers those with a difficulty level of 4 or above. This aligns with intuition: for instance, when assisting a middle school student with math problems, it is more beneficial to provide examples of moderate difficulty that they can comprehend, whereas for a high school student, examples with a higher level of difficulty are more useful.

Table 17: Smaller, less capable LLMs favor simpler CoT examples, while larger ones prefer more complex ones.

Model	Difficulty (≤ 3)	Difficulty (3-4)	Difficulty (≥ 4)
LLaMA2-7B	14.49	**15.39**	14.86
LLaMA2-13B	23.81	25.32	**25.47**

In our evaluation of CoT-Influx with various LLMs, we found that the shot selection results are consistent with our observation. The average difficulty score and number of reasoning steps for the examples selected by LLaMA2-70B pruner are 3.57 and 3.04, which are higher than the results of LLaMA2-13B are 3.51 and 2.98. The empirical results further support our assumption that LLMs with larger sizes prefer harder examples than smaller-scale LLMs.

B.4 The effectiveness of MRD^3 dataset

We verify the effectiveness of the MRD^3 dataset, which is evolved by GPT-4 for diverse reasoning steps and difficulties. We compare with two baselines: (1) MRD^3 without evolution, excluding GPT-4 evolved examples, and (2) the human-labeled GSM8K training set, which excludes GPT-4's reformatted generation. We apply our pruner on these datasets under the same setting. As shown in Table 18, both GPT-4 generated and evolved CoT examples are vital for improving the reasoning performance.

Table 18: Comparison of EM(%) on GSM8K using CoT-Influx pruner across different CoT datasets.

CoT dataset	LLaMA2-7B	LLaMA2-13B	LLaMA2-70B
MRD^3	**15.92**	**32.37**	**59.59**
MRD^3 w/o evolution	14.94	30.55	57.70
GSM8K training set	14.18	29.64	56.71

B.5 Evolution example

We give an example of a math question and then show the evolved results of the questions and answers. The evolved results follow our instructions given in Sec. B.1. As can be seen from the evolution results, the answer to input questions can be modified (*e.g.* ground truth answer change from 16 to 12 in add_constraints). The whole background of the questions can also be replaced (*e.g.* computation target of question change from current tickets at the arcade to final points of a game in produce_easier). These modifications and variations improve the diversity of our prompt candidate sets, which are the building block for the training and reasoning with CoT-Influx.

Example of prompt evolution

Input example for evolution:
Q: At the arcade Dave won 11 tickets. If he spent 5 tickets on a beanie and later won 10 more tickets, how many would he have?
A: Dave initially had 11 tickets, spent 5 on a beanie, and later won 10 more, resulting in a total of: 11 - 5 + 10 = 16 tickets. The answer is 16.

Evolution result of add_constraints:
Q: At the arcade, Dave won 11 tickets. If he spent 5 tickets on a beanie, won 10 more tickets, and gave 4 tickets to his friend, how many would he have?
A: Dave initially had 11 tickets, spent 5 on a beanie, won 10 more, and gave 4 to his friend, resulting in a total of: 11 - 5 + 10 - 4 = 12 tickets. The answer is 12.
Evolution result of deepening:
Q: At the arcade, Dave triumphed in a game and earned 11 tickets. He redeemed 5 of those tickets for a beanie and subsequently won an additional 10 tickets. What is his final ticket count?
A: Dave started with 11 tickets, used 5 to acquire a beanie, and later added 10 more to his collection, resulting in: 11 - 5 + 10 = 16 tickets. The answer is 16.
Evolution result of increase_reasoning:
Q: Dave won 11 tickets at the arcade and spent 5 on a beanie. Afterward, he won 10 more tickets. Calculate his final ticket count by first finding the remaining tickets after his purchase and then adding the newly won tickets.
A: Dave had 11 tickets, spent 5, leaving him with 6. Then he won 10 more, resulting in: 6 + 10 = 16 tickets. The answer is 16.
Evolution result of revise_difficulty:
Q: Dave got 11 tickets at the arcade. He used 5 tickets to get a beanie and then won 10 more tickets. How many tickets does he have now?
A: Dave started with 11 tickets, used 5 for a beanie, and won 10 more, giving him: 11 - 5 + 10 = 16 tickets. The answer is 16.
Evolution result of produce_easier:
Q: In a game, Sarah scored 11 points. She lost 5 points in a penalty and then scored 10 more points. What is her final score?
A: Sarah initially scored 11 points, lost 5 due to a penalty, and scored 10 more, resulting in a total of: 11 - 5 + 10 = 16 points. The answer is 16.

C Pruner Training and Evaluation Details

C.1 Detailed algorithm for training data preparation

As a supplement to phase 1 in Algorithm 1, we show the algorithm for training data preparation in Algorithm 2. Both the difficulty level and the number of reasoning steps are involved in the GPT-4-based evaluation. However, we omit the reasoning step in this algorithm as we only use difficulty level in the training set split.

Algorithm 2 Training dataset preparation

```
Input: CoT dataset $\{x_i^{\text{cot}}\}_{i=1}^{L}$, difficulty threshold $d_{\text{thr}}$,
Output: MRD$^3$ $\mathcal{D} = \{x_j^{\text{cot}}, d_j\}_{j=1}^{L^{\text{MRD}^3}}$, questions set $\mathcal{D}_{\text{question}}$, prompt set $\mathcal{D}_{\text{cot}}$
 1: ▶ Phase 1: MRD$^3$-Evol Preparation
 2: MRD$^3$ dataset $\mathcal{D} = \{\}$
 3: for $i = 1$ to $L$ do
 4:     Perform GPT-4 based prompt evolution on $x_i^{\text{cot}}$ to get $\{x_{i,e}^{\text{cot-E}}\}_e$
 5:     Evaluate difficulty on $\{x_{i,e}^{\text{cot-E}}\}_e$ to get score $\{d_{i,e}\}_e$ using GPT-4
 6:     Append examples $\{x_{i,e}^{\text{cot-E}}, d_{i,e}\}_e$ to $\mathcal{D}$
 7: end for
 8: Prompt set $\mathcal{D}_{\text{cot}} = \{\}$, question set $\mathcal{D}_{\text{question}} = \{\}$
 9: for $j = 1$ to $L^{\text{MRD}^3}$ do
10:     if $d_j \leq d_{\text{thr}}$ then
11:         Append example $(x_j^{\text{cot}}, d_j)$ to $\mathcal{D}_{\text{question}}$
12:     else
13:         Append example $(x_j^{\text{cot}}, d_j)$ to $\mathcal{D}_{\text{cot}}$
14:     end if
15: end for
16: Return full dataset with evolution $\mathcal{D}$, questions set $\mathcal{D}_{\text{question}}$, prompt candidate set $\mathcal{D}_{\text{cot}}$
```

C.2 Detailed settings and hyperparameters

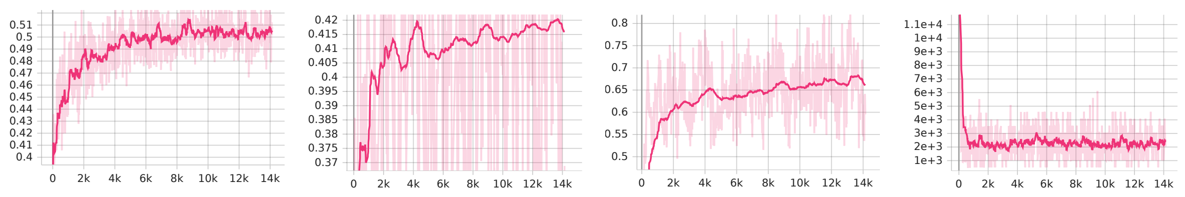
The detailed hyper-parameters setting of different LLMs' pruners are listed in Table 19. The majority of these hyperparameters are shared across different LLMs. The evolution subset as the prompt candidates for evaluation is determined by searching the performance of math reasoning on 100 random examples.

Table 19: Detailed hyper-parameters for pruner training scheme of different LLMs.

Model	LLaMA2-7B	LLaMA2-13B	LLaMA2-70B
Epoch	3	3	3
Batch Size	1	1	1
Pruner LLM Base	LLaMA2-13B	LLaMA2-13B	LLaMA2-70B
Input Shot	40	48	48
Input Shot (TopK)	32	32	32
Input Shot (Few-shot)	8	16	16
Optimizer	AdamW	AdamW	AdamW
Weight Decay	$1e^{-2}$	$1e^{-2}$	$1e^{-2}$
Learning Rate	$1e^{-5}$	$1e^{-5}$	$1e^{-5}$
Embedding Extractor	BERT-Large (cased)	BERT-Large (cased)	BERT-Large (cased)
Embedding Size	1024	1024	1024
Tokenizer Padding	512	512	512
Difficulty Threshold d_{thr}	2	2	2
Token Target T	2048	2048	2048
Token Penalty Coefficient w	(-1,1)	(-1,1)	(-1,1)
Selection Repeat t_{repeat}	10	10	5
Evol Subset	add_constraints	increase_reasoning	increase_reasoning
temperature	0.8	0.8	0.8
top_p	0.95	0.95	0.95
top_k	40	40	40
num_beams	1	1	1
max_new_tokens	1	1	1

C.3 Training dynamics

We visualize the RL training dynamics of the LLaMA2-13B pruner in Figure 8 including the LLM loss reward $\frac{1}{1+L_{\text{LLM}}}$, prediction reward R_{Acc}, moving average of the final pruner reward R, and remaining token count t. We can see from the results that the reward increases steadily with the RL training steps. The number of remaining tokens decreases rapidly in the early steps and then oscillates around the token target. Since our prediction reward R_{Acc} are discrete values of $\{-0.1, 0, 1\}$, the oscillation phenomenon is more obvious compared with another reward term. This highlights the effectiveness of question repetition and using the Exponential Moving Average (EMA) of the final reward to suppress this oscillation.



(a) LLM loss reward $\frac{1}{1+L_{\text{LLM}}}$ (b) Prediction reward R_{Acc} (c) EMA pruner reward R (d) Remaining token t

Figure 8: RL training dynamics of the LLaMA2-13B pruner.

C.4 Detailed introduction of dataset for evaluation

We introduce the details of the datasets we used for evaluation as follows:

- **GSM8K** (Cobbe et al., 2021) is a math reasoning dataset consisting of high-quality linguistically diverse grade school math word problems created by human problem writers. There are 7473 training examples and 1319 validation examples in the dataset.

- **SVAMP** (Patel et al., 2021) representing **S**imple **V**ariations on **A**rithmetic **M**ath word **P**roblems that conduct question sensitivity variation, reasoning ability variation, and structural variation on existing math datasets. There is a total of 1000 examples, and all of them are used for evaluation in our settings.

- **MultiArith** (Roy and Roth, 2015) is a collection of multi-step arithmetic problems with 600 examples that are used for evaluation in our settings.

- **AddSub** (Hosseini et al., 2014) is a dataset consisting of addition and subtraction problems with 395 examples, all used for evaluation in our settings.

- **SingleEq** (Koncel-Kedziorski et al., 2015) consists of grade-school algebra word problems that map to single equations with varying length. Every equation may involve multiple math operations, including multiplication, division, subtraction, and addition over non-negative rational numbers and only one variable. There are 508 problems, 1117 sentences, and 15292 words in the dataset.

C.5 Rule-based prompt reconstruction

To make sure the input prompt for inference remains structurally intact, we apply a rule-based prompt reconstruction on the input. For example, "*\n [question]*" will be reconstructed to "*\nQ: [question]*" and "*A: Let's step by step*" will be repaired to "*A: Let's think step by step*". While our pruner has been trained to learn the importance of structure integrity and consistency, there are still a few cases when important tokens are pruned, leading to incorrect reasoning results. The rule-based reconstruction can effectively alleviate the influence of a sub-optimal pruning strategy. The important tokens that should be reconstructed include 'Q:', 'A:', '\n', "Let's think step by step", and "The answer is".

D Additional Related Works

LLM In-Context Learning In-context learning (ICL) is one of the emerging abilities of LLMs that conduct various downstream tasks with provided few-shot demonstrations. To fully understand and optimize the ICL paradigm, previous research mainly focuses on the underlying mechanism of ICL or the proper application of ICL. Pioneering research (Von Oswald et al., 2023; Dai et al., 2023) empirically find the similarity between gradient-descent (GD) and ICL, which interprets the trained LLMs as meta-optimizers that can learn the examples in the context in the forward pass. More recently, Wang et al. (2023a) propose a hypothesis that label words in examples serve as anchors in ICL, and the anchors can help aggregate and distribute the task-relevant information flow. To better utilize ICL, previous research also researched the input format (Yoo et al., 2022) and order of examples (Min et al., 2022). Our work falls in the second category, which shows that compressed examples are an optimal choice for the input of ICL.

LLM Context Window Extension Recently, there has been rising interest in extending the context window of existing pre-trained LLMs. Common approaches include augmenting external memory modules (Tworkowski et al., 2023; Wang et al., 2023c), which add extra modules to memorize long past contexts but requires complex training, manipulating attention mechanisms (Han et al., 2023; Xiao et al., 2023) or the positional encoding (Chen et al., 2023a; Peng et al., 2023b). However, these require LLM modifications. Our method, applicable to black-box LLMs and extendable context windows, is orthogonal to this direction.

Comparison of CoT-Influx with Previous Methods We summarize the advantage of our CoT-Influx compared with previous prompting strategies in Table 20. *Gradient-free* indicates the methods do not need to be backward through LLMs. *Unlimited-token* represents the original input prompt for these methods, which are not limited by the context window length of LLMs. *Difficulty-aware* refers to whether the method takes the difficulty of problems into consideration of their prompt design. *Dynamic #Shots* means we do not need to set a target shot number, and the pruned input shot numbers are different across various questions. Our CoT-Influx demonstrates a significant advantage over all previous methods.

Table 20: Comparison of the advantages of different prompting strategies.

Methods	Frozen LLMs	Automated	Gradient-free	Unlimited-token	Transferable	Interpretable	Difficulty-aware	Dynamic #Shots
Fine-Tuning	✗	✓	✗	✗	✗	✗	✗	✗
Manual Prompt	✓	✗	✓	✗	✓	✓	✗	✗
Soft Prompt Tuning	✓	✓	✗	✗	✗	✗	✗	✗
Prompt Retrieval	✓	✓	✓	✗	✓	✓	✗	✗
AutoPrompt (Shin et al., 2020)	✓	✓	✗	✗	✓	✓	✗	✗
RLPrompt (Deng et al., 2022)	✓	✓	✓	✗	✓	✓	✗	✗
Context Extension	✓	✓	✓	✓	✓	✓	✗	✗
LLMLingua (Jiang et al., 2023b)	✓	✓	✓	✓	✓	✓	✗	✗
CoT-Influx(Ours)	✓	✓	✓	✓	✓	✓	✓	✓

E Prompt Settings

In this section, we show the prompt we used in this work for reproducibility. The prompt for evaluating the difficulty and reasoning steps of each example are:

Prompt for difficulty and reasoning steps estimation:

We would like you to evaluate and rate the difficulty and complexity of the following question. You should first give an overall score on a scale of 1 to 10, where a higher score indicates higher difficulty and complexity. You should then evaluate the answer and give how many reasoning steps are in the answer. You must just give the score and the number of reasoning steps without any other reasons. The reply format should be 'Score': [score], 'Steps: [#steps]'
Question: **{Given question}**
Answer: **{Given answer}**
Evaluation:

The prompt for GPT-4 Compression on prompts is shown as follows. Note that we encode the restriction of token limits in both the prompt and API by setting the *max_new_token*. However, the prompt compression results still fail to follow the restrictions for most cases. This disadvantage of uncontrollable token length is also discussed in previous work (Jiang et al., 2023b).

Prompt for GPT-4-based compression:

Please compress the following examplars for few-shot in-context learning on math reasoning. The complete exemplars could be removed if they are redundant, and the tokens within each exemplar can also be pruned. 'The answer is' in each examplar should be retained and please keep less than **{Given token}** tokens in total:
{Given examplars}