

A ILLUSTRATION OF TRAINING EFFICIENCY OF CURRENT METHODS

Training efficiency comparisons across current long-context methods are inherently challenging: performance gains typically exhibit diminishing returns with increased token budgets, and reported results often stem from divergent training setups — including data composition, optimization hyper-parameters, and hardware configurations. These factors render direct “gain-per-token” comparisons unreliable when conditions are unmatched. To fairly compare training efficiency across methods — despite differing hyper-parameters and convergence behaviors — we adopt a controlled proxy: average task gain per 1B tokens, measured under identical data, optimizer, batch size, learning rate, and hardware ($8 \times$ A100 GPUs). Specifically, we compare ProLong (Gao et al., 2024b) - one long-context SFT method, and LongCE (Fang et al., 2024b) - one token-level re-weighting training method, on the Llama3-8B-Base model. As shown in Table 3, we evaluate model performance on LongBench-E (12 real-world tasks) per 50 training steps (0.41B tokens per 50 steps), and find that LongCE achieves a 13-point gain per 1B tokens versus ProLong’s 0.3-point gain per 1B tokens.

Table 3: Performance comparison between ProLong (SFT) and LongCE across training steps, where each step contains the same training setting.

Method	Step 0	Step 50	Step 100	Step 150	Step 200
ProLong (SFT)	25.50	27.32	28.15	28.44	29.13
LongCE (same data)	25.50	28.30	29.72	31.01	32.91

B PRELIMINARY STUDY DETAILS

B.1 PRELIMINARY TASK CONSTRUCTION

Task Selection We select 3-hop and 4-hop tasks based on qa3 tasks in the BABILong Benchmark to build our datasets, as these tasks generally pose significant challenges for LLMs. However, it is worth noting that the original BABILong qa4 samples do not truly require 4-hop reasoning to produce correct outputs. For example, a sample from this subset with 0k context is shown in Figure 11. In this case, the task only requires attention to a single fact, “The bedroom is west of the bathroom” to answer the question, while the first sentence serves as an interference fact. Even in terms of keywords, the model only needs to focus on three keywords: “bathroom”, “west”, and “bedroom” from the second sentence. Thus, we design our 4-hop dataset based on the BABILong qa3 source data, with one sample shown in Figure 12. By carefully arranging the order of facts and reducing the conditions of questions in the long context, we ensure that the model is required to search for all four supporting facts in sequence to produce the correct output.

Table 4: Variable settings, where R. denotes random.

Hops	Samples	Permute	Lengths
2	100	5	8K
3/4	R.	R.	0k - 64k

Controlled Evaluation Data Synthesis We use the 4-hop task with non-zero context as an example here. As shown in Table 4, all variables used for building data include the facts sample, the facts permutation, and the context length. Firstly, we select source samples from the BABILong official file “qa3_three-supporting-facts” as our base data. Then, we modify the original BABILong qa3 supporting facts following the pattern shown in Figure 13. Afterward, we add interference to these four original facts while maintaining the relative order of the supporting facts. The process begins by selecting a noise context of the appropriate length and inserting the facts into it. Specifically, we divide the noise context into 10 equal-length chunks, leaving 10 candidate positions for the insertion of the 4 supporting facts (excluding the tail). Next, we randomly select five permutations from the full set of C_{10}^4 candidate position permutations. After injecting noise, we randomly insert interference facts, i.e., facts that are similar to the supporting facts but irrelevant, among all sentences. We

Table 5: Performance statistics of using different numbers of attention heads on our preliminary synthetic task. Notably, we find that selecting the top-30 heads yields results that are nearly identical to those obtained when using all attention heads.

Head Number	Supporting		Interference		Irrelevant		Low-frequency	
	Correct	Wrong	Correct	Wrong	Correct	Wrong	Correct	Wrong
Top-30	0.21	0.11	0.07	0.17	0.72	0.72	0.00	0.00
All	0.20	0.13	0.09	0.15	0.71	0.72	0.00	0.00

ensure that at least one interference fact is placed after the last supporting fact to test the model’s robustness. To ensure the correctness of the samples, we make sure that the objects appearing in the interference facts do not overlap with those in the supporting facts. Additionally, we ensure that the number of interference facts is between one and two times the number of supporting facts to avoid making the samples either too easy or too difficult. Finally, for all samples with the same context length, we use the same noise context to maintain consistency. In the end, we randomly insert a few emojis into the constructed context to test the sensitivity of the model to low-frequency tokens. For the 3-hop task, we directly use the original qa3 task format from BABILong as the base, and the subsequent processing follows a similar approach to the one described above for the 4-hop task.

One BABILong qa4 sample with 0k context

Input :

The bedroom is west of the office.
The bathroom is west of the bedroom.

Question:

What is west of the office?

Supporting Facts:

The bedroom is west of the office.

Ground truth:

bedroom

Figure 11: A BABILong qa4 sample with 0k context

B.2 DESIGN OF IG SCORE

Prior work (Wu et al., 2024) has shown that not all attention heads behave uniformly, i.e., some are specialized for retrieval-like behaviors, while others are not. However, it is important to note that these findings are primarily derived from studies focused on copy-oriented tasks, such as NIAH. In contrast, our task involves reasoning and inference, which fundamentally differs from the objectives of retrieval heads. As a result, the mechanisms for attending to relevant context in our setting cannot be directly aligned with those used in retrieval-focused tasks. To further analyze the appropriate number of attention heads to select, we conduct experiments where we select the top-k attention heads ($k = 30$) that retrieve the most relevant information based on the attention scores (Table 5). We find that the performance using only a subset of attention heads was highly consistent with the results obtained by averaging over all attention heads. Therefore, for simplicity and ease of deployment, we adopt the latter approach, i.e., averaging IG scores across all attention heads.

C DERIVATION OF RELATION BETWEEN INFORMATION FLOW AND EMBEDDING GRADIENTS

In transformer-based models, the Information Flow in attention is essentially the product of the attention distribution and its corresponding gradient. Therefore, we can transform the derivation into **constructing the gradient relationship between the attention score distribution (A) and**

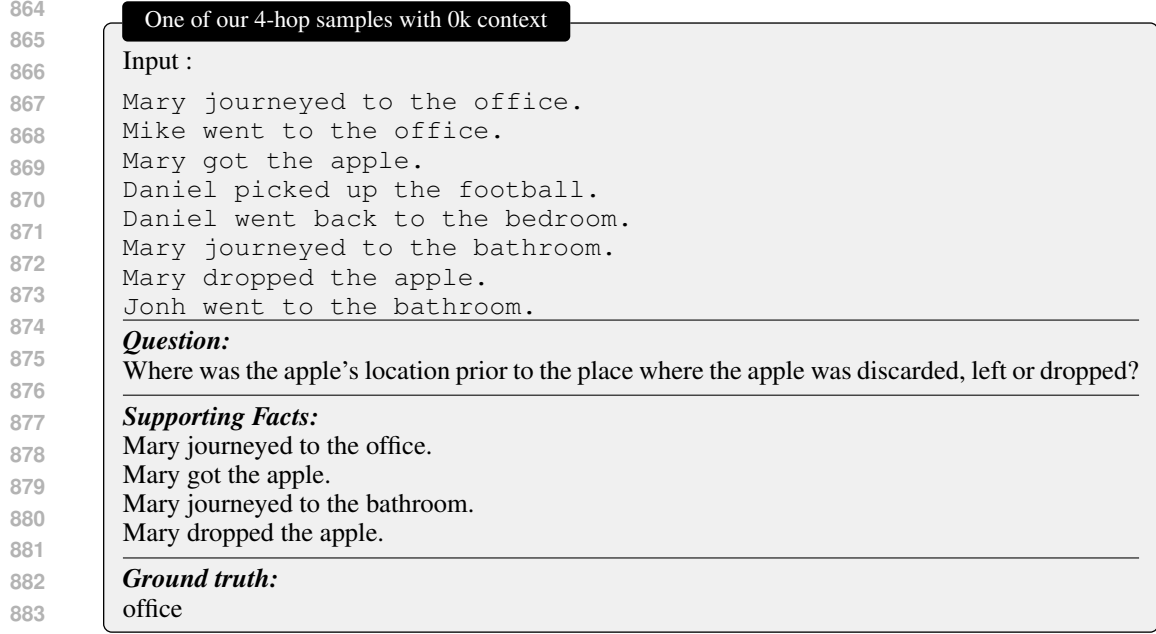


Figure 12: One of our 4-hop samples with 0k context

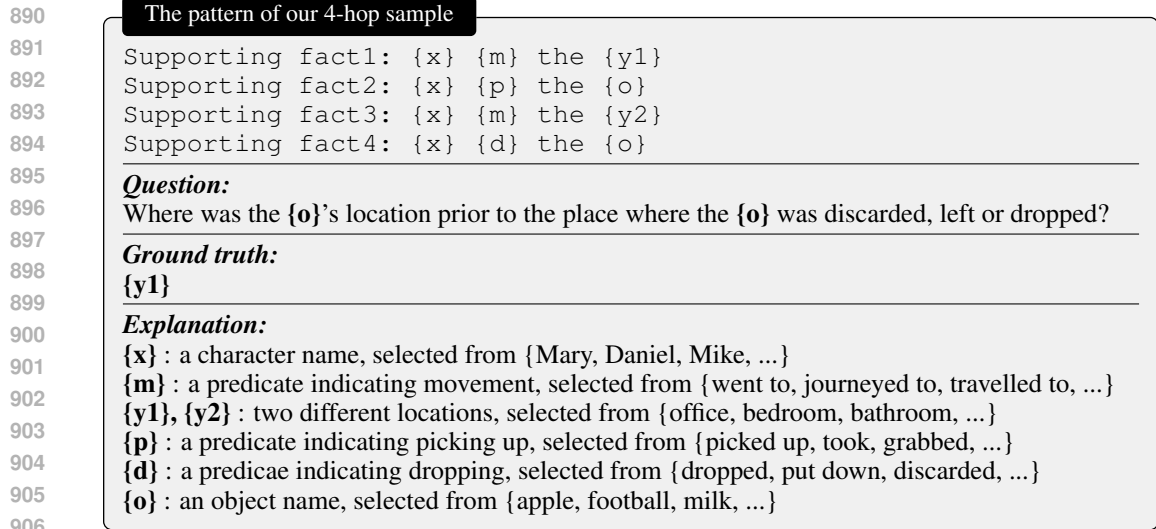


Figure 13: The pattern of our 4-hop sample

the embedding ($E(X)$). This can be established via the chain rule and implemented through the specific computation steps of the attention mechanism. Notably, in the following derivation, for simplicity, we omit the activation layers in the model. Additionally, considering that transformer-based models are composed of multiple identical network blocks stacked together, one can easily extend the conclusions from a single layer to multiple layers. Therefore, we focus on proving the case with **one embedding layer and one attention module**.

Given the basic definition of the attention mechanism, we have:

$$\begin{cases} Q = E(X)W_Q, & A = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right), \\ K = E(X)W_K, & O = A \cdot V, \\ V = E(X)W_V, \end{cases}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$ are the model parameters, O is the attention output, $E(X) \in \mathbb{R}^{n \times d}$ is the input embedding matrix, n and d are sequence length and model dimension, respectively.

Let the loss function be L . By the chain rule, the gradient of the loss with respect to $E(X)$ is:

$$\begin{aligned} \frac{\partial L}{\partial E(X)} &= \frac{\partial L}{\partial O} \frac{\partial O}{\partial E(X)} = \frac{\partial L}{\partial A} \frac{\partial A}{\partial E(X)} \\ &\quad + \frac{\partial L}{\partial V} \frac{\partial V}{\partial E(X)}. \end{aligned} \quad (5)$$

Since we have $\frac{\partial V}{\partial E(X)} = W_V^T$ and $\frac{\partial O}{\partial V} = A$, the gradient relationship between A and $E(X)$ is:

$$\frac{\partial L}{\partial E(X)} \propto \frac{\partial L}{\partial A} \frac{\partial A}{\partial E(X)} \quad (6)$$

To eliminate the influence of the Softmax($\cdot$) function, we can further decompose equation 6 into:

$$\begin{cases} S = \frac{QK^T}{\sqrt{d}}, \\ \frac{\partial L}{\partial E(X)} \approx \frac{\partial L}{\partial A} \cdot \left(\frac{\partial A}{\partial S} \cdot \frac{\partial S}{\partial E(X)} \right), \end{cases} \quad (7)$$

where $\frac{\partial A}{\partial S}$ is the Jacobian of Softmax($\cdot$) function, with elements $A_{ij}(\delta_{ik} - A_{ik})^2$.

For each element $S_{ij} = \frac{Q_i K_j^T}{\sqrt{d}} \in S$, the gradient with respect to $E(X)$ can be written as:

$$\begin{aligned} \frac{\partial S_{ij}}{\partial E(X)} &= \frac{\partial \left(\frac{(E(X)_i W_Q)(E(X)_j W_K)^T}{\sqrt{d}} \right)}{\partial E(X)} \\ &= \frac{1}{\sqrt{d}} (W_Q^T \cdot K_j \cdot \delta_{ik} + W_K^T \cdot Q_i \cdot \delta_{jk}). \end{aligned} \quad (8)$$

Based on equation 7 and equation 8, we can summary that:

$$\begin{aligned} \frac{\partial L}{\partial E(X)_i} &\propto \underbrace{\frac{\partial L}{\partial A_{ij}}}_{\text{Sensitivity of } L \text{ to } A} \\ &\quad \times \underbrace{A_{ij}(1 - A_{ij})}_{\text{Derivation from Softmax}} \\ &\quad \times \underbrace{\frac{\partial S_{ij}}{\partial E(X)}}_{\text{Linear Transformation}}. \end{aligned} \quad (9)$$

Based on equation 9, we can derive that when A_{ij} increases, indicating higher attention between token i and token j , the sensitivity of L to A ($\frac{\partial L}{\partial A_{ij}}$) also increases. This results in larger derivatives on the embeddings. Additionally, if A_{ij} becomes excessively large, approaching 1, the value of $A_{ij}(1 - A_{ij})$ might tend toward 0. However, this is often not an issue in long-context scenarios, as the attention scores are unlikely to approach values near 0.5 due to the long context. Even if they exceed 0.5 (possibly for some special tokens), the increase in the first term ($\frac{\partial L}{\partial A_{ij}}$) helps mitigate this effect.

² δ_{ik} is the Kronecker delta function. If i equals to k , $\delta_{ik} = 1$, else $\delta_{ik} = 0$. We can also rewrite this equation into $A_{ij}(1 - A_{ij})$.

Table 6: Configuration of context window scaling training setting.

Context Window Scaling Training Setting	
Backbone	Llama-3-8B-base
Training Objective	Language modeling
RoPE base	20,000,000
Context window size	8K $\rightarrow$ 64K
Data seq-length	64,000
Deepspeed	Zero2
Global batch size	64
Epoch	2
Training Steps	160
Ring-attention size	4
Learning-rate	1e-5
LR-scheduler	cosine_with_min_lr
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.95$)
GPUs	A100 (80GB) $\times$ 8
Training time	$\approx$ 8h / epoch
Training data	PG19 (Rae et al., 2019)
Total consumed tokens	0.65B

Table 7: Configuration of language modeling training setting.

Language Modeling Post-training Setting	
Backbone	Llama-3.1-8B-base
Training Objective	Language modeling
RoPE base	500,000
Context window size	128K
Data seq-length	64,000
Deepspeed	Zero2
Epoch	2
Global batch size	32
Training Steps	320
Ring-attention size	4
Learning-rate	1e-5
LR-scheduler	cosine_with_min_lr
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.95$)
GPUs	A100 (80GB) $\times$ 8
Training time	$\approx$ 8.5h / epoch
Training data	PG19 (Rae et al., 2019)
Total consumed tokens	0.65B

Table 8: Configuration of long-context SFT training setting.

Long-context Alignment Training Setting	
Backbone	Llama-3.1-8B-Instruct
Training Objective	Supervised fine-tuning
RoPE base	500,000
Context window size	128K
Data seq-length	4,000 $\sim$ 128,000
Deepspeed	Zero2
Global batch size	32
Epoch	2
Training Steps	250
Ring-attention size	4
Learning-rate	1e-5
LR-scheduler	cosine_with_min_lr
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.95$)
GPUs	A100 (80GB) $\times$ 8
Training time	$\approx$ 6.5h / epoch
Training data	LongMIT (Chen et al., 2024), LongAlpaca (Chen et al., 2023c)
Total consumed tokens	0.53B

Table 9: Testing configuration of RULER

Evaluation Configuration of RULER	
Question Answering	qa_1, qa_2
Single NIAH	niah_single_1, niah_single_2, niah_single_3
Multi-keys NIAH	niah_multikey_1, niah_multikey_2, niah_multikey_3
Multi-values NIAH	niah_multiquery
Multi-queries NIAH	niah_multivalue
Others	common words extraction (CWE), frequent words extraction (FWE), variable tracking (VT)
Length	32K, 64K
Num samples/task	50

D IMPLEMENTATION DETAILS

D.1 TRAINING DETAILS

For all experiments, we utilize the open-source training framework OpenRLHF³ (Hu et al., 2024), Ring-flash-attention⁴ (Liu et al., 2023) and DeepSpeed (Rajbhandari et al., 2020). For LongCE training (Fang et al., 2024b), we set the sliding context window size as 8192 and employ the recommended hyper-parameters in the official code⁵.

Context Window Scaling To scale the context window size of the Llama-3-8B-base model from 8K to 64K (8 $\times$), we adjust the RoPE base from 500,000 to 20,000,000 and directly train the model. We provide training configurations in Table 6.

Data Post-processing Details For the context window scaling experiments, we employ the PG-19 (Rae et al., 2019) dataset. For long-context SFT and CDT experiments, we construct our data

³<https://github.com/OpenRLHF/OpenRLHF.git>

⁴<https://github.com/zhuzilin/ring-flash-attention.git>

⁵<https://github.com/PKU-ML/LongPPL.git>

Table 10: Testing configuration of BABILong.

Metric	QA1	QA2	QA3	QA7	QA8
Num	100	100	100	100	100
Supporting Fact	1	2	3	1~10	1~8
Interference Fact	1~9	1~66	1~317	1~42	1~42

Table 11: Evaluation results on HELMET (Yen et al., 2025).

Model	Recall	RAG	ICL	Re-rank	QA	Summ.	Cite	Avg.
Claude-3.5-Sonnet	94.7	38.1	61.0	7.2	12.6	36.6	18.7	38.4
Mistral-Nemo-12B	14.6	40.0	84.0	0.0	22.5	18.5	0.5	25.7
ProLong-512K-Instruct	98.8	63.2	86.5	22.5	43.9	29.2	1.4	49.4
Meta-Llama-3.1-8B	95.2	59.5	83.9	14.0	43.2	27.0	2.9	46.5
+ CDT	97.2	61.8	86.6	18.5	46.7	27.9	9.4	49.7

from publicly available long-context QA datasets, including LongMiT (Chen et al., 2024) and LongAlpaca (Chen et al., 2023c). The LongMiT dataset primarily consists of multi-hop QA tasks that require reasoning over 2 to 6 evidence passages. To adapt it for our setting, we apply two pre-processing steps: (i) Length distribution control — we constrain the sampled instances to fall within 16K–128K tokens. This range balances the need for sufficiently long contexts with training efficiency, given our compute resources ($8 \times$ A800 GPUs). Excessively long sequences were avoided as they considerably slow down training. (ii) Evidence balancing — we uniformly sample across different numbers of supporting passages to obtain a more balanced distribution for multi-hop reasoning. To complement this, we include data from LongAlpaca, which predominantly features single-evidence QA with lengths around 16K tokens (under our model’s tokenizer). This addition enriches the training distribution by covering shorter single-evidence scenarios, which are underrepresented in LongMiT. In total, our final training set comprises 7,000 samples from LongMiT and 1,000 samples from LongAlpaca, which are shuffled together before training.

Language Modeling Post-training and Long-context SFT The language modeling post-training and long-context SFT are directly applied to the Llama3.1-8B-base and Llama3.1-8B-Instruct, respectively, which already have 128K context window size. We provide the training configurations in Table 7 and Table 8 respectively.

D.2 EVALUATION DETAILS

We conduct long-context evaluation mainly based on the long-context evaluation framework LOOM-Scope⁶ (Tang et al., 2025).

HELMET HELMET (Yen et al., 2025) is a comprehensive long-context evaluation benchmark containing 7 different subtasks, including recall, RAG, in-context learning (ICL), re-rank, QA, summarization, and citation. The context length of test samples ranges from 0 to 128K tokens. For inputs exceeding 128K, we truncate from the end to fit within the model’s maximum context window. We show the experimental results on HELMET in Table 11, where our CDT model achieves the best performance.

LongBench-E LongBench-E is a variant of LongBench (Bai et al., 2024b) designed specifically for long-context real-world tasks. We chose LongBench-E because it shares the same test dataset distribution as LongBench while covering a wider range of context lengths. For the Llama3-8B-base model, we truncate the input to 8K tokens, whereas for other models, we truncate the input to 32K tokens.

⁶<https://github.com/LCM-Lab/LOOM-Scope>

Table 12: Evaluation results of two more LLMs on real-world long-context tasks and long-form reasoning tasks.

Models	LongBench-E					BABILong		
	Type	S-Doc QA	M-Doc QA	Summ	Few-shot	Code	Avg.	Avg.
Qwen2.5-7B-Instruct	-	44.54	46.29	28.15	56.03	16.52	38.30	43.32
+ CDT	SFT	44.93	47.29	28.65	57.33	19.18	39.48	47.56
Qwen3-8B	-	44.12	48.10	29.30	44.12	29.18	38.85	48.06
+ CDT	SFT	45.33	49.13	31.89	46.24	32.98	41.11	52.88
Mistral-V0.3-Instruct	-	44.89	40.76	20.52	67.11	47.04	44.06	22.36
+ CDT	SFT	45.01	41.79	26.08	67.75	57.27	47.58	53.84

Language Modeling For the language modeling task, we calculate both LongPPL and PPL metrics on the GovReport dataset (Huang et al., 2021), which consists of long sequences from government reports. We sample 50 documents from GovReport, each with a context length of up to 32K tokens.

RULER RULER (Hsieh et al., 2024) is a comprehensive synthetic dataset that includes 6 different testing categories to evaluate a model’s long-context understanding capabilities. We utilize all test categories, with each category containing 50 test samples covering lengths of 32K and 64K. We post the testing configuration of RULER in Table 9.

Long-form Reasoning We evaluate the long-form reasoning capability of models on selected tasks from BABILong (Kuratov et al., 2024). Specifically, we select tasks that involve multiple supporting facts, as well as QA1, as the testing dataset. The BABILong testing configurations are shown in Table 10.

D.3 BASELINE ILLUSTRATION

We evaluate our method on three foundation models, i.e., LLaMA-3-8B-Base, LLaMA-3.1-8B-Base, and LLaMA-3.1-8B-Instruct—to ensure fair and consistent comparisons across all baselines. The baselines include: YaRN, which extends the context window using an improved NTK-based positional scaling method; CE (Cross Entropy), a standard language modeling objective without any context-aware weighting; LongCE, which builds upon the LongPPL method by identifying key tokens via perplexity during training and assigning them higher loss weights; SFT, an instruction tuning setup where input tokens are excluded from the loss calculation; and LOGO, a DPO-based training approach designed to mitigate misalignment in long-context tasks. Additionally, we compare against several strong open-source long-context models: ProLong-512K-Instruct and NExtLong-512K-Instruct, which apply long-context scaling techniques on top of LLaMA-3-8B-Instruct and LLaMA-3.1-8B-Instruct, respectively; and LLaMA-3.1-8B-SEALONG, a DPO-trained model specifically optimized for long-context alignment.

Table 13: Model performance on language modeling tasks.

Models	LongPPL	PPL
Llama-3-8B-Base	> 100	> 100
+ YaRN	3.55	5.60
+ CE	3.90	6.46
+ LongCE	3.55	5.60
+ CDT (ours)	3.04	5.40
Llama-3.1-8B-Base	3.22	4.79
+ CE	3.28	4.86
+ LongCE	3.24	5.28
+ CDT (ours)	2.10	5.19
Llama-3.1-8B-Instruct	4.05	5.52
+ SFT	3.31	5.51
+ LOGO	4.11	5.54
+ CDT (ours)	2.36	5.64

E MORE EVALUATION RESULTS

E.1 ANALYSIS OF RESULTS ON REAL-WORLD LONG-CONTEXT TASKS

The strong performance of CDT on code-related tasks, as shown in Table 1, is particularly notable. Code Completion requires models to accurately interpret local context and pre-

Table 14: Statistical significance calculation on LongBench-E data with t-Test.

Models	P-Value
Llama3-8B-Base V.S. Llama3-8B-Base-CDT	3.68e-15
Llama3.1-8B-Base V.S. Llama3.1-8B-Base-CDT	1.53e-2
Llama3.1-8B-Instruct V.S. Llama3.1-8B-Instruct-CDT	2.39e-3

dict missing segments accordingly. CDT is especially well-suited for such tasks, as it enhances the model’s ability to focus on local context information during generation, which likely contributes to the observed performance improvements. Table 15 offers a more intuitive illustration through a specific Code Completion example. In LongBench-E, this task is evaluated using the Edit Similarity (Edit Sim) metric, which is highly sensitive to the number of tokens generated—especially under the official 64-token generation limit. In the provided example, LLaMA-3.1-8B-Instruct produces entirely incorrect outputs, while GPT-4o generates overly lengthy responses that negatively affect the Edit Sim score. In contrast, the CDT-enhanced model generates a concise and accurate response, resulting in a significantly higher Edit Sim score. Furthermore, CDT leads to substantial improvements for both LLaMA-3.1-8B-Instruct and LLaMA-3-8B-Base on the Code task. These improvements can be attributed to two main factors. First, the training set includes code completion instances (e.g., 263 examples from LongMIT), which enable the model to learn relevant instruction-following patterns. Second, the baseline model’s lower performance in this domain makes the gains from CDT more apparent. By contrast, LLaMA-3.1-8B-Base already demonstrates strong performance on code-related tasks—likely due to the composition of its pretraining data—resulting in smaller relative gains when CDT is applied.

E.2 GENERALIZING CDT TO MORE MODELS

We apply our CDT method to more LLMs, including Qwen2.5-7B-Instruct (Yang et al., 2024) and Mistral-V0.3-Instruct (Jiang et al., 2023). We evaluate the model performance on real-world long-context tasks, long synthetic tasks, and long-form reasoning tasks. We report the model performance in Table 12, where we can observe that our CDT can significantly improve the model performance on different models. For instance, the Mistral-V0.3-Instruct model obtains more than 30 points on the long-form reasoning task.

E.3 EVALUATION RESULTS ON LANGUAGE MODELING TASKS

Apart from evaluating with LongPPL on the language modeling task, we also calculate the PPL scores, which are shown in Table 13.

E.4 EXPERIMENT STATISTICAL SIGNIFICANCE

We collect the prediction results of the original model and the CDT model on the LongBench-E benchmark, and conduct a paired-samples t-test to assess the statistical significance of the mean difference before and after the improvement, shown in Table 14. The results show that our method significantly outperforms the baseline model at the 5% significance level, indicating that our method achieves statistically significant improvements.

F ANALYSIS OF ATTENTION MAP BEFORE AND AFTER CDT

In this section, we present a visualization of the model’s attention patterns before and after applying the CDT training strategy. Given the long input length (12,000 tokens) used in our evaluation, we evenly partition the input sequence into 46 chunks and calculate the total attention score for each chunk individually. For each chunk, a higher total attention score indicates that the model places greater focus on this chunk. We visualize the attention maps of the 24th layer of the model, as this layer provides the clearest representation of CDT’s impact. As shown in Figure 14, we can observe that, before applying CDT, the model’s attention is predominantly concentrated on the question

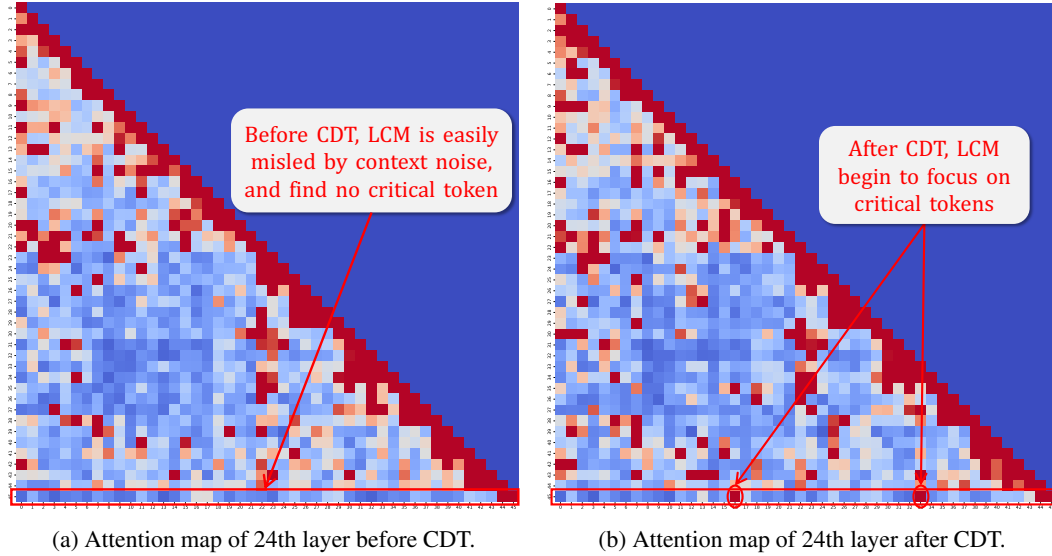


Figure 14: Comparison between Attention Maps Before and After CDT. In each figure, a deeper (red) color indicates larger model attention to the corresponding context chunk. The final row of each map represents how the question attends to the entire input sequence, including both the context and the question itself. For clearer visualization, we recommend zoom in on this figure.

itself (the rightmost portion of the final row in Figure 14a), while key information within the context is largely overwhelmed by noise. In contrast, after CDT training, the model not only attends to the question but also shows significantly increased attention to relevant contextual information, as highlighted by the red circles in the final row of Figure 14b. **It is noteworthy that the attention map shows no significant changes before and after CDT training, indicating that CDT training does not compromise the original characteristics of the LCM. Instead, it enhances the ability of LCM to capture critical information.**

G LIMITATION AND FUTURE WORK

Due to the expectation maximization (EM) nature of CDT, it includes an additional context noise detection process, which introduces extra computational costs during the training phase. Although we have demonstrated in Section 6.3 that these additional costs are negligible compared to the performance gains, theoretically, *the noise detection cost will increase as the model size grows since it involves a complete forward-backward propagation process*. We leave this for future work, aiming to explore a simpler method for identifying the context noise or to develop more efficient model architectures. For example, designing specific network modules to handle noise, as proposed in Ye et al. (2024), could be a promising direction. Additionally, we observe that *the improvement brought by our method on complex reasoning tasks is not as significant as that on other tasks*, and we are yet to understand the relationship between this and the training data or the training objective function. In the future, we aim to further investigate the impact of context noise on the model’s long-form reasoning abilities, as well as the relationship between the CDT strategy and the enhancement of the model’s reasoning capabilities.

H USE OF LLMs

During the writing of this paper, we leveraged large language models (LLMs) to refine the clarity and fluency of our writing, particularly in the Abstract and Introduction sections. Specifically, we used the Qwen web interface⁷ to access the Qwen series of models (e.g., Qwen-Max), inputting early drafts of these sections and requesting stylistic improvements while preserving technical accuracy

⁷<https://chat.qwen.ai>

and original intent. The model’s suggestions helped enhance sentence structure, academic tone, and overall readability. All final content was carefully reviewed, validated, and edited by the authors to ensure fidelity to our research and adherence to scholarly standards.

I ERROR ANALYSIS

In this section, we analyze the error pattern of partial model predictions on real-world long-context tasks. As shown in Table 15, we use colored text to highlight the correct and incorrect parts of the model’s predictions.

Table 15: Generation samples of Llama3.1-8B-Instruct, Llama3.1-8B-Instruct-CD (ours), and GPT4o on real-world long-context tasks.

Code Completion Task		
Question:	Completing the code based on the context:	Llama3.1-8B-Instruct:
[...]\n self.interface =		for (int i
self.interface.replace(' ',		= 0; i < s.length(); i++) {if
").lower() \n self.param_type		(!Character.isDigit(s.charAt(i))
= None \n self.if_type = None		&& s.charAt(i) != ',') }
\n # state \n self.results		Llama3.1-8B-CD (ours):
= dict() \n [Completion]		if
self.result = dict() \n		self.interface == 'all':\n
def check_params(self): \n		self.param_type = INTERFACE_ALL\n
"""Check all input params"""		GPT4o:
\n if not self.interface: \n		if self.interface.lower()
self.module.fail_json(msg='Error:		== 'all':\n self.param_type
Interface name cannot be		= INTERFACE_ALL\n elif
empty.') [...]		'/' in self.interface or
		self.interface.isdigit():\n
		self.param_type =
		INTERFACE_FULL_NAME\n
		else:\n self.if_type =
		get_interface_type(self.interface)\n
		Golden Answer: if self.interface ==
		'all':\n
Long-context Multi-hop Reasoning Task		
Question:	What did M. King	Llama3.1-8B-Instruct:
Hubbert's employer announce		Shell announced
it was in the process of doing		the sale of some of its assets,
in April 2010? [...]		including its liquefied petroleum
In April		gas (LPG) business, to meet the
2010 Shell announced that the		cost of a planned \$28
corporation is in process of		Llama3.1-8B-CD (ours):
trying to find a potential		trying to find
buyer for all of its operations		a potential buyer for all of
in Finland and is doing similar		its operations in Finland and
market research concerning		is doing similar market research
Swedish operations. [...]		concerning Swedish operations.
		GPT4o:
		MV Miralda's owner
		announced it was in the process
		of trying to find a potential
		buyer for all of its operations
		in Finland and in doing similar
		market research concerning.
		Golden Answer:
		trying to find a
		potential buyer for all of its
		operations in Finland
Single-QA Long-context Task		
Question:	What metrics are used	Llama3.1-8B-Instruct:
in evaluation? [...]		Macro-averaged F-score.
Figure		Llama3.1-8B-CD (ours):
FIGREF22 shows the results		macro-averaged F-score,
of these experiments wherein		precision, recall, F1 score.
we are able to maintain the		GPT4o:
precision on the subset of the		Macro-averaged F-score.
training set used for testing		Golden Answer:
while reducing its recall and		precision, recall , F1 score
thus generalising the model.		
The fact that the model is		
generalizing on a dissimilar		
dataset is confirmed by the		
increase in the development		
set. F1 score. [...]		