One-Pass to Reason: Token Duplication and Block-Sparse Mask for Efficient Fine-Tuning on Multi-Turn Reasoning

Ritesh Goru
DevRev
Austin, USA
ritesh.goru@devrev.ai

Shanay Mehta
DevRev
Bengaluru, India
shanay.mehta@devrev.ai

Prateek Jain
DevRev
Austin, USA
prateek.jain@devrev.ai

Abstract

Fine-tuning Large Language Models(LLMs) on multi-turn reasoning datasets requires N (number of turns) separate forward passes per conversation due to reasoning token visibility constraints, as reasoning tokens for a turn are discarded in subsequent turns. We propose duplicating response tokens along with a custom attention mask to enable single-pass processing of entire conversations. We prove our method produces identical losses to the N-pass approach while reducing time complexity from $O\left(N^3\right)$ to $O\left(N^2\right)$ and maintaining the same memory complexity for a transformer based model. Our approach achieves significant training speedup while preserving accuracy. Our implementation is available online[1].

1 Introduction

Recent progress in LLMs has sparked a shift from models that directly generate final responses to those that perform explicit intermediate reasoning before generating responses (referred to as reasoning models). Open-source reasoning models, such as DeepSeek-R1 [6], demonstrate high performance on several benchmarks. However, these existing reasoning models were trained primarily on single-turn reasoning data.

While numerous studies have investigated fine-tuning LLMs for multi-turn dialogues to improve coherence, context awareness, tool-calling [17, 13], these approaches assume non-reasoning dialogues.

Training LLMs for multi-turn reasoning conversations presents novel challenges in managing token visibility. Following industry-standard practices for multi-turn conversations [10, 1], reasoning models generate internal reasoning tokens, produce a response, and then discard the reasoning tokens from the context in subsequent turns. This creates two fundamental constraints that cannot be addressed with standard multi-turn optimization techniques: (1) **Visibility Constraints**: Reasoning tokens must be visible during generation but hidden from subsequent conversation turns, requiring conditional visibility that static attention masks cannot satisfy. (2) **Position ID Discrepancy**: Response tokens follow reasoning tokens during generation but directly follow human messages in a later context, creating positional misalignment.

While prior works have explored masking techniques and position ID assignments to control information flow and enable selective attention within sequences for various pre-training objectives or efficiency gains [16, 5, 12], none address the specific challenges of multi-turn reasoning conversations where reasoning tokens must be conditionally visible across turns.

[1] `https://github.com/devrev/One-Pass-to-Reason`

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MTI-LLM @ NeurIPS 2025.

This paper addresses these challenges with two primary contributions. (1) We present a theoretical framework featuring a block-sparse visibility mask and strategic position ID assignment scheme that enables processing an entire multi-turn reasoning conversation in a single forward pass while maintaining training correctness (Theorem 2.3). (2) Due to the absence of a publicly available multi-turn reasoning dataset (to the best of our knowledge), we create and release a novel dataset, MathChat$_{\text{sync}}$Reasoning, in which each assistant message is augmented with synthetically generated reasoning. (3) We provide comprehensive empirical validation for the proposed framework on Qwen3 models [18].

Notation. We use $\mathcal{D}$ to denote a multi-turn reasoning dataset where each conversation $c \in \mathcal{D}$ consists of N turns. Each turn T_i contains a sequence of alternating human and assistant messages: $T_i = (h_{i,j}, a_{i,j})_{j=1}^{M_i}$ where M_i is the number of message pairs in turn i. Thus, a complete conversation is $c = (T_i)_{i=1}^{N}$. Each assistant message $a_{i,j}$ comprises thinking tokens $t_{i,j}$ and response tokens $r_{i,j}$. We denote $T_{i,<j} = (h_{i,k}, a_{i,k})_{k=1}^{j-1}$ as turn history before jth set of messages and $\mathcal{H}_{<i} = (O_k)_{k=1}^{i-1}$ as the conversation history before turn i, where $O_k = (h_{k,j}, r_{k,j})_{j=1}^{M_k}$ represents the observable content of turn k (excluding thinking tokens). We denote $O_{i,<j} = (h_{i,k}, r_{i,k})_{k=1}^{j-1}$ as observable turn history before jth set of messages. For token sequence x, s_x and e_x represent starting and ending position IDs. The notation $x \to \mathcal{A}(\cdot)$ indicates sequences that x attends to, and $\mathcal{L}(\cdot)$ denotes language modeling loss (detailed in Appendix A.1).

2 Single Pass Fine-tuning on Multi-Turn Reasoning

In this section, we highlight the challenges associated with fine-tuning language models on multi-turn reasoning datasets. We present an optimized approach to process an entire conversation in a single forward pass. In multi-turn reasoning data, response tokens $r_{i,j}$ must attend to reasoning tokens $t_{i,j}$ during the generation of assistant message $a_{i,j}$, as well as to thinking tokens from all previous assistant messages within the same turn $(t_{i,k})_{k=1}^{j-1}$. Similarly, human messages $h_{i,j+1}$ within the same turn following the generation of $a_{i,j}$ must attend to reasoning tokens $t_{i,j}$ and thinking tokens from all previous assistant messages within that turn $(t_{i,k})_{k=1}^{j-1}$. However, these reasoning tokens must remain invisible from the context during generation of subsequent turns $T_{l>i}$. Within the same turn $(l = i)$, assistant messages and human messages that follow assistant messages can access thinking tokens from all previous messages in that turn. Consequently, it is not possible to construct a single static attention mask that satisfies both conditions in a conversation within a single forward pass—a capability that is typically feasible with non-reasoning datasets.

2.1 N-Pass Approach

A straightforward solution is to perform a separate forward pass for every turn $(\mathcal{H}_{<i}, T_i)$ of a given conversation c. While functionally correct, this approach is computationally inefficient: a conversation with N assistant turns results in N separate training examples. Consequently, the effective size of the dataset increases from $|\mathcal{D}|$ to $|\mathcal{D}| \times N$, inflating training time proportionally. Fig. 1(a) shows causal attention mask at the time of generation of ith turn response tokens, and Fig. 1(b) shows causal attention mask for ith turn response tokens when they are part of context during $j > i$ turns.

2.2 1-Pass Approach

The primary challenge in applying a single forward pass during training due to discrepancy in the attention behavior of $r_{i,j}$ and $h_{i,j+1}$ can be illustrated as follows[2]:

$$r_{i,j} \to \begin{cases} \mathcal{A}(\mathcal{H}_{<i}, T_{i,<j}, h_{i,j}, t_{i,j}) & \text{generation} \\ \mathcal{A}(\mathcal{H}_{<i}, O_{i,<j}, h_{i,j}) & \text{context} \end{cases}$$

$$h_{i,j+1} \to \begin{cases} \mathcal{A}(\mathcal{H}_{<i}, T_{i,<j+1}) & \text{generation} \\ \mathcal{A}(\mathcal{H}_{<i}, O_{i,<j+1}) & \text{context} \end{cases}$$

We can resolve this issue through the following steps:

[2] For ease of understanding, we omit the detail that each token within a token sequence also attends to all its preceding tokens, which must be encoded in the attention mask.

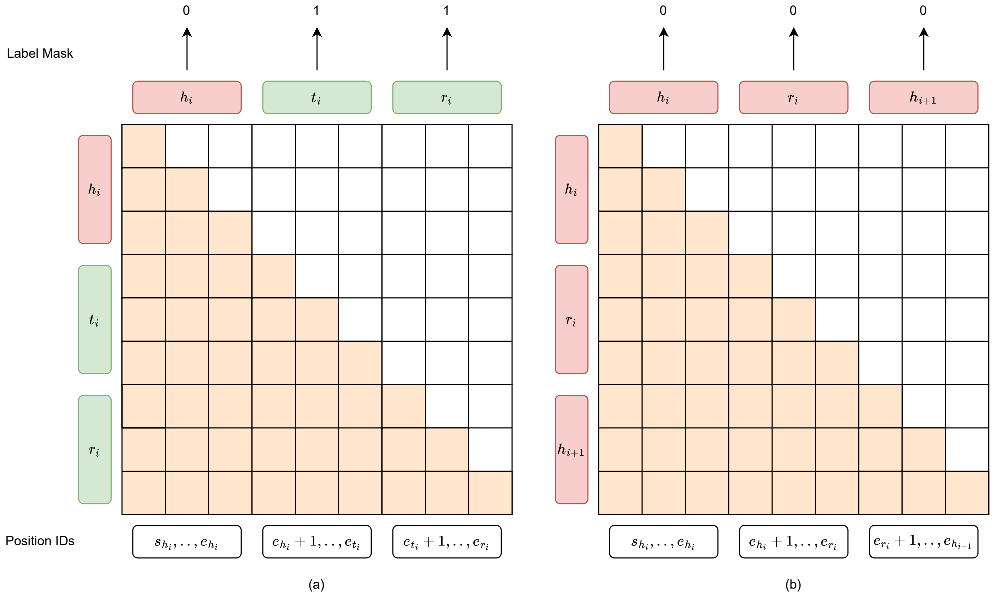


Figure 1: Causal Attention Masks for N-Pass Approach 🟧 represents non-zero attention. (a) Attention Mask for generation of response tokens. (b) Attention Mask when response tokens are in context.

Duplicating response tokens of each assistant message and subsequent human message. We duplicate the response tokens for each assistant message and subsequent human message so that one sequence ($r_{i,j}^{out}$) and ($h_{i,j+1}^{out}$) is used during generation and attends to preceding thinking tokens of turn i. In contrast, the other sequence (r_i^{in}) and ($h_{i,j+1}^{in}$) is used only as context and does not attend to thinking tokens.

Custom Attention Mask. Duplication of response tokens makes it possible to have a single attention mask that satisfies visibility constraints. We define a custom masking strategy for each type of token sequence, ensuring that each token only attends to the appropriate subsequence:

$$\begin{aligned}
t_{i,j} &\rightarrow \mathcal{A}(\mathcal{H}_{<i}^{in}, T_{i,<j}^{out}, h_{i,j}^{out}) \\
r_{i,j}^{out} &\rightarrow \mathcal{A}(\mathcal{H}_{<i}^{in}, T_{i,<j}^{out}, h_{i,j}^{out}, t_{i,j}) \\
h_{i,j+1}^{out} &\rightarrow \mathcal{A}(\mathcal{H}_{<i}^{in}, T_{i,<j+1}^{out}) \\
r_{i,j}^{in} &\rightarrow \mathcal{A}(\mathcal{H}_{<i}^{in}, O_{i,<j}^{in}, h_{i,j}^{in}) \\
h_{i,j+1}^{in} &\rightarrow \mathcal{A}(\mathcal{H}_{<i}^{in}, O_{i,<j+1}^{in})
\end{aligned}$$

Assigning Consistent Position IDs. After duplication of response tokens, we need to assign consistent position IDs to tokens to maintain the correct relative positions—as if multiple forward passes were performed for each turn in the conversation. If they are assigned sequentially, or the duplicated assistant response tokens share the same position IDs, it will lead to incorrect relative positions. We need a strategic way of assigning position IDs. The following assignment of the first position ID for each token sequence ensures the relative positions are correct and equivalent to N-Pass approach[3]:

$$
\begin{aligned}
s_{t_{i,j}} &= e_{h^{out}_{i,j}} + 1 \\
s_{r^{out}_{i,j}} &= e_{t_{i,j}} + 1 \\
s_{h^{out}_{i,j+1}} &= e_{r^{out}_{i,j}} + 1 \\
s_{r^{in}_{i,j}} &= e_{h^{in}_{i,j}} + 1 \\
s_{h^{in}_{i,j+1}} &= e_{r^{in}_{i,j}} + 1
\end{aligned}
$$

Label Mask. Duplication of the response tokens also raises the question of which tokens should be included in the loss calculation. The following label mask outlines the inclusion criteria for each token type:

$$
\begin{aligned}
h^{in}_{i,j} &\leftarrow 0 \\
h^{out}_{i,j} &\leftarrow 0 \\
t_i &\leftarrow 1 \\
r^{in}_i &\leftarrow 0 \\
r^{out}_i &\leftarrow 1
\end{aligned}
$$

Fig. 2 shows custom attention mask for ith turn in the 1-Pass Approach. It combines masks for generation and context from the N-Pass Approach into a single mask with position IDs and a label mask consistent with N-Pass Approach.

We have demonstrated the ability to do 1-Pass approach for a specific type of conversation structure, but it can be done for any prefix compatible compatible defined below:

Definition 2.1 (Causally Consistent Conversation)**.** A conversation c is *causally consistent* if, for every assisatnt token a generation step, given the subsequence of preceding tokens $S_a = (s_1, s_2, \ldots, s_k)$ that a attends to, then the attention set of each token $s_i \in S_a$ must be exactly $\{s_j \in S_a : j < i\}$ at this generation step of t.

Theorem 2.2. *Given a causally consistent conversation, one can always construct an attention mask that enables single forward pass of all tokens through the language model while preserving all causal attention constraints.*

Proof. Let c be a causally consistent conversation. We construct a token sequence T (allowing duplicates when needed) and an attention mask M such that a single forward pass over T with mask M respects all causal constraints.

Initialization. Set $T \leftarrow \emptyset$ and let M be an empty square matrix.

Iterative construction. Process assistant tokens a in their generation order in c. For each such a, let $S_a = (s_1, \ldots, s_k)$ be the ordered list of tokens a must attend to at its generation step.

- For every $s_i \in S_a$, check if an instance of s_i already exists in T whose attention pattern equals exactly $\{s_j \in S_a : j < i\}$ at this generation step. If yes, reuse that instance; otherwise create a new instance of s_i and append it to T.
- For each newly created instance, extend M with a new row and column, and set its attention to exactly the tokens corresponding to $\{s_j \in S_a : j < i\}$.
- Append a to T, extend M by one row and column, and set the attention row of a to the (possibly duplicated) instances of tokens in S_a.

Properties. By causally consistent definition, every token appearing in S_a itself only attends to earlier elements of S_a. Hence each instance we place into T can be given a fixed attention pattern matching that step, and M can encode those visibilities. Because a's row in M points exactly to the instances of S_a, a observes the same context as in the original generation step. Since we process

[3] Position IDs are assigned sequentially based on the order of tokens within each sequence.

tokens in their causal order and only add backward-looking (causal) edges, a single forward pass over T with M is valid.

Therefore there exists a sequence T and mask M enabling a single forward pass that preserves all causal constraints. □

Theorem 2.3. *Consider a language model with output distributions determined solely by attention patterns, positional encodings, and input representation. For any causally consistent conversation c as input to the model, the sum of the N-Pass language modeling losses is equivalent to the 1-Pass loss:*

$$\mathcal{L}^{\textit{1-Pass}}(c) = \mathcal{L}^{\textit{N-Pass}}(c)$$

Proof. We establish the equivalence by demonstrating that both approaches yield identical probability distributions over sequences, which directly implies equal language modeling losses. The proof proceeds in three parts: we show that (1) positional encodings can be made equivalent, (2) attention patterns are identical, and (3) the resulting loss functions are mathematically equivalent.

Part I: Position Encoding Equivalence. In the N-Pass approach, for each turn, tokens receive sequential position IDs based on their order in the current context.

In the 1-Pass approach with token duplication (from Theorem 2.2):

- When we create a new instance of a token due to different attention requirements, we can assign it contiguous position ID following its preceding tokens because of causally consistent nature of the conversation.
- For the tokens which are not duplicated they by default follow the attention requirements and can be assigned sequential position ID due to same reason.
- For each assistant token a with attention sequence S_a at the time of generation, the absolute and relative positions of tokens in S_a are preserved.

Since the model's output depends only on positions within the attention window, and these positions are identical in both approaches for each token's generation, the positional encoding contribution to the model output is equivalent.

Part II: Attention Pattern Preservation. From the construction in Theorem 2.2:

- For each assistant token a in conversation c, let S_a be the sequence of tokens it attends to at the time of generation.
- In the N-Pass approach, token a attends exactly to the tokens in S_a with their causal attention patterns.
- In the 1-Pass approach, our mask construction ensures:
  - We duplicate any token $s_i \in S_a$ if its required attention pattern differs from existing instances.
  - Each duplicated instance maintains the exact attention pattern required by causal consistency.
  - Token a attends to the appropriate instances that match the attention patterns in S_a.

Therefore, the attention patterns observed by each assistant token a are identical in both approaches.

Part III: Loss Function Equivalence. The language modeling loss for the N-Pass approach is:

$$\mathcal{L}^{\textit{N-Pass}}(c) = - \sum_{a \in \text{assistant tokens}} \log P_\theta \left(a \mid S_a \right) \tag{1}$$

where $P_\theta(a \mid S_a)$ is the probability of token a given its context S_a with the attention patterns and positional encodings as specified in the N-Pass approach. For the 1-Pass approach:

- A label mask ensures only assistant tokens contribute to the loss.

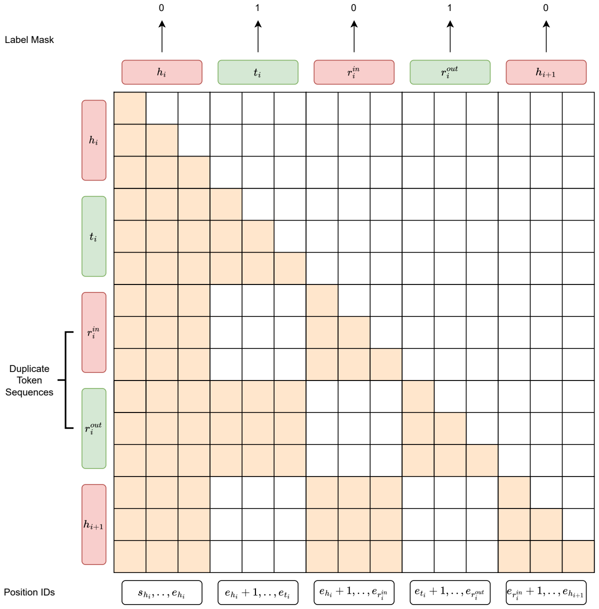

Label Mask
0
1
0
1
0
h_i
t_i
r_i^{in}
r_i^{out}
h_{i+1}
h_i
t_i
r_i^{in}
Duplicate Token Sequences
r_i^{out}
h_{i+1}
Position IDs
$s_{h_i}, .., e_{h_i}$
$e_{h_i} + 1, .., e_{t_i}$
$e_{h_i} + 1, .., e_{r_i^{in}}$
$e_{t_i} + 1, .., e_{r_i^{out}}$
$e_{r_i^{in}} + 1, .., e_{h_{i+1}}$

Figure 2: Custom Attention Mask for 1-Pass Approach. 🟧 represents non-zero attention.

- From Parts I and II, each assistant token a sees identical:
  - Attention patterns to tokens in S_a.
  - Positional encodings.
  - Input representations (through appropriate token instance selection)

Since the model's output distribution depends solely on these three factors (by hypothesis), we have: $P_\theta^{\text{1-Pass}}(a \mid S_a) = P_\theta^{\text{N-Pass}}(a \mid S_a)$

Therefore, the 1-Pass loss is:

$$\begin{aligned}\mathcal{L}^{\textit{1-Pass}}(c) &= -\sum_{a \in \text{assistant tokens}} \log P_\theta(a \mid S_a) \\ &= \mathcal{L}^{\textit{N-Pass}}(c)\end{aligned} \tag{2}$$

This completes the proof of loss equivalence. □

2.3 Complexity Analysis

We compare the computational complexity of our 1-Pass method against N-Pass approach for transformer-based models with hidden dimension d [15]. Table 1 summarizes the time and memory complexities for a conversation c, where ℓ denotes its characteristic turn length.

	N-Pass	1-Pass
$T(c)$	$O\big(N^3\ell^2 d\big)$	$O\big(N^2\ell^2 d\big)$
$M(c)$	$O\big(N^2\ell^2\big)$	$O\big(N^2\ell^2\big)$

Table 1: Time and Memory Complexity for N-Pass and 1-Pass Approach.

The 1-Pass approach yields an asymptotic time complexity improvement of one order in N, offering significant speedups at scale. While it introduces a higher constant memory overhead due to token replication, both methods share the same asymptotic memory complexity. Full derivations are provided in Appendix B.

2.4 Efficient Mask Generation

While our custom attention mask (illustrated in Figure 2) enables single-pass training, generating it involves computing complex visibility patterns across token types and conversation turns. At scale, this computation could become non-trivial, particularly for longer conversations or larger batch sizes. To ensure this remains efficient, we develop an optimized mask generation algorithm that performs all operations on GPU using vectorized tensor operations. Additionally, we simplify the boolean logic for visibility constraints using Karnaugh map reduction, minimizing the number of logical operations required. We provide the complete algorithm in Appendix C.1 for practitioners seeking to implement our method efficiently.

3 Experiments

We evaluate our single-pass fine-tuning on Qwen3 models (4B, 8B, 32B) with QLoRA [3]. All experiments were run on a 8×H100 instance (CUDA 12.8, PyTorch 2.7.0), with our method implemented in LLaMA-Factory [19] and benchmarked against multi-pass baselines. See Appendix C.2 for experimental setup.

Qwen3's Persistent Reasoning. Qwen3 is a reasoning model. When used in Non-Thinking mode, it still appends an empty reasoning block (<think></think>) to each response. As a result, a reasoning segment—albeit empty—remains in every output. This implies that any fine-tuning scenario aiming for contamination-free training (including those that use purely non-reasoning data) must employ our 1-Pass approach with token duplication to properly isolate and manage these empty reasoning tokens. This ensures that no unintended reasoning fragments leak into subsequent turns.

3.1 Dataset Creation

To enable supervised training with explicit step-by-step reasoning, we construct and release MathChat$_{\text{sync}}$Reasoning[4] along with its generation script. The dataset is obtained by augmenting the original MathChat$_{\text{sync}}$ corpus [9] with a synthetically-generated rationale for every assistant turn. The procedure comprises three stages.

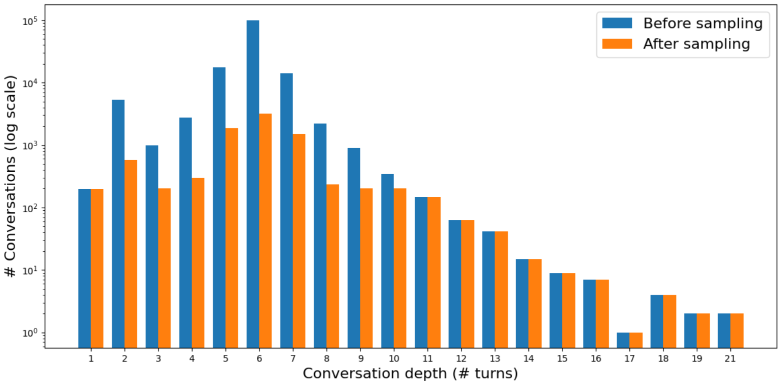
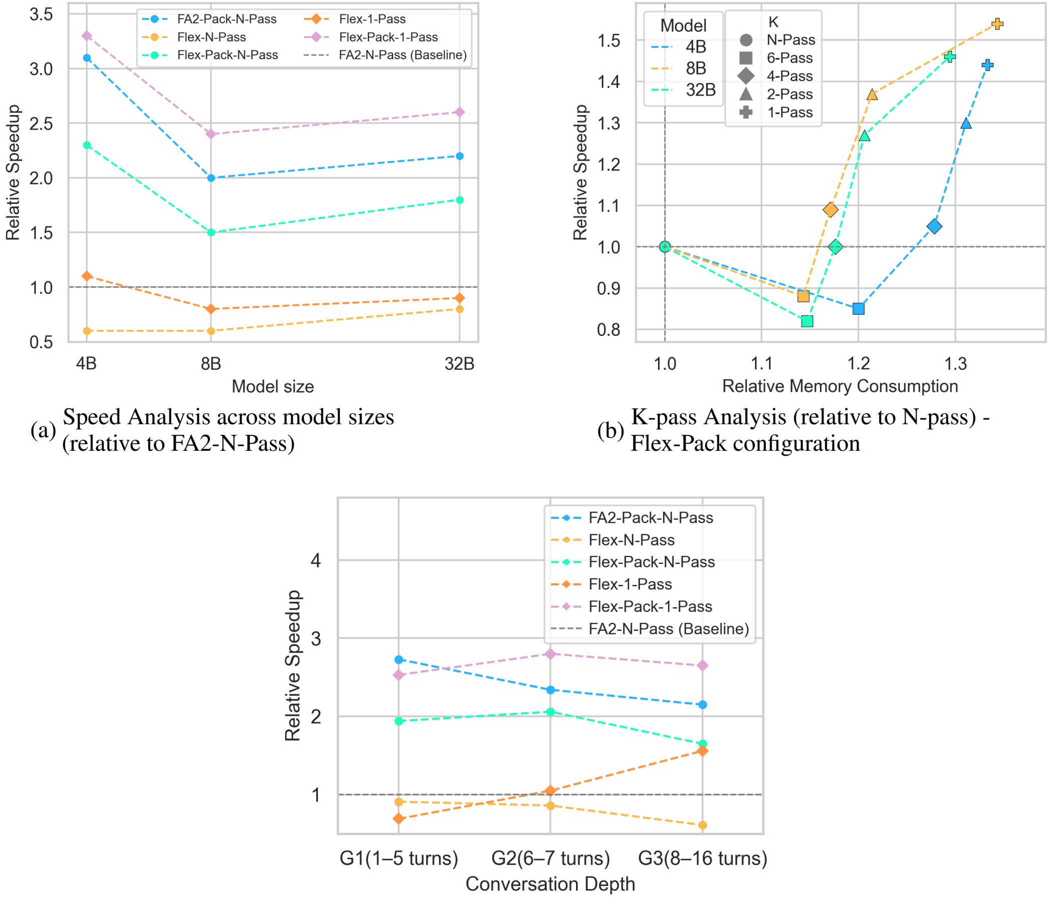


Figure 3: Dataset depth distribution: before vs. after sampling.

1. Source corpus. MathChat$_{\text{sync}}$ is a synthetic, dialogue-based mathematics tutoring dataset containing 144,978 conversations with alternating human and assistant messages but no reasoning traces.

[4] `https://huggingface.co/datasets/devrev-research/MathChatSync-reasoning`



(a) Speed Analysis across model sizes (relative to FA2-N-Pass)

(b) K-pass Analysis (relative to N-pass) - Flex-Pack configuration

(c) Conversation Depth Analysis on Qwen3-8B (relative to FA2-N-Pass)

Figure 4: Training-time experiments

2. Depth-balanced sampling. Conversation depth in MathChat$_{\text{sync}}$ is highly skewed toward six-turn dialogues (69 % of all conversations; see Figure 3). To mitigate this bias, we first down-sample depth-6 dialogues from 100,443 to 30,000 instances. From the resulting pool we draw a stratified sample of 8,000 conversations.

- For each depth d, we calculate the proportion of the pool that depth represents.
- We allocate to that depth the corresponding proportion of the 8,000-conversation budget, rounding up to the nearest whole conversation.
- If the resulting number is below 200, we raise it to (i) 200 or (ii) the total number of conversations available at that depth, whichever is smaller. This guarantees broad coverage across conversation depths.

The final split contains 8,797 assistant turns. Figure 3 compares the depth distribution before and after sampling.

3. Reasoning augmentation. For every assistant turn we generate an intermediate reasoning string using gpt-4.1-mini. The model is provided with (i) the dialogue history up to the current human utterance and (ii) the assistant's reply, and is instructed to output only the hidden rationale that could have produced that reply. These rationales are concatenated to the original conversations to form MathChat$_{\text{sync}}$Reasoning. All experiments in this paper use this dataset.

3.2 Experimental Setup

We use FlashAttention2 (FA2) [2] and FlexAttention [4] backends. Our 1-Pass method requires a custom attention mask, thus using FlexAttention, as FA2 lacks support for passing custom attention mask; FA2's speed motivates reporting baselines on both for fair comparison. We compare our **1-Pass method** (with response token duplication) against a standard **N-Pass baseline** (requiring N forward passes). Both are evaluated with and without sequence packing[5] [7]. When packing is enabled, we use llama-factory's `neat_packing` implementation: FA2 baselines rely on position IDs to separate packed sequences [8], while our 1-pass method combines the contamination-free packing mask with our custom attention mask via logical AND.

3.3 Results:

Training Speedup. Figure 4a shows training speedups. Our 1-Pass method with packing (Flex-Pack-1-Pass) is $1.05\times$, $1.21\times$, and $1.22\times$ faster than FA2-N-Pass baseline with packing (FA2-Pack-N-Pass) on 4B, 8B, and 32B models, respectively. Despite FlexAttention's inherent slowness versus FA2, our method's single-pass efficiency compensates. Compared to N-Pass FlexAttention with packing (Flex-Pack-N-Pass), our Flex-Pack-1-Pass yields $1.44\times, 1.54\times,$ and $1.46\times$ speedups for 4B, 8B, and 32B models, respectively. Without packing, our 1-pass method (Flex-1-Pass) lags FA2-N-Pass baseline for 8B and 32B models. We hypothesize that this is because response-token duplication widens the length disparity between conversations, making the method more sensitive to the absence of packing than the N-Pass baseline. Across all experiments, the 1-Pass variants consume roughly 33% more GPU memory than their N-Pass counterparts.

K-Pass Trade-offs. The 1-Pass and N-Pass approaches represent two extremes: processing the entire conversation in a single pass or in as many passes as there are turns. We therefore also investigate intermediate settings, processing each conversation in K passes. Concretely, we split every dialogue into K contiguous chunks and apply our single-pass mask only to the current chunk, duplicating response tokens and computing loss exclusively for that portion (see Appendix C.3.1 for full details). Figure 4b reveals a speed-memory trade-off for K$\in$1,2,4,6,N. Our 1-Pass method maximizes speed with $\sim$33% more memory (vs. N-Pass). K=2 offers a balance (1.30×–1.37× speedups, $\sim$20% extra memory). Gains diminish for $K > 4$ because, beyond $K = 4$, the extra time incurred by the longer sequences created through token duplication outweighs the savings from processing a few turns together.

Conversation Scalability. The dataset contains conversations with depths from 1 to 16 turns. To analyse the effect of depth, we partition it into three groups: G1 (1–5 turns), G2 (6–7 turns), and G3 (8–16 turns)[6]. Figure 4c shows our Flex-Pack-1-Pass speedups (vs. FA2-Pack-N-Pass) grow with conversation depth ($0.93\times, 1.19\times, 1.23\times$ for G1, G2, G3 respectively). A similar trend appears when comparing our method without packing (Flex-1-Pass) to the FA2-N-Pass baseline: speedups of $0.69\times$, $1.05\times$, and $1.56\times$ for G1, G2, and G3, respectively. This supports the theoretical complexity reduction from $O\big(N^3\big)$ to $O\big(N^2\big)$, as efficiency gains become more pronounced with depth.

These results confirm single-pass training yields significant computational savings, aligning with theoretical advantages, making multi-turn reasoning fine-tuning practical at scale. Please refer Appendix C.3 for comprehensive results of the experiments conducted.

4 Conclusion

We presented an optimized 1-Pass training method for multi-turn reasoning that reduces time complexity from $O(N^3)$ to $O(N^2)$ via strategic token duplication and custom attention mask. Our theoretical analysis confirms loss equivalence with the N-Pass method, enabling efficient training for longer conversations. As multi-turn reasoning becomes central to complex AI tasks, our method offers a scalable and broadly applicable solution. Future work includes exploring adaptive strategies to balance memory-efficiency trade-offs. Additionally, we aim to benchmark performance on latest back-ends such as FlashAttention3 [14] and port our masking logic to these faster implementations.

[5] We set the cutoff length to the maximum number of tokens in any datapoint in the dataset for all our experiments.

[6] This uneven distribution originates from the underlying MathChat$_{\text{sync}}$ dataset, which is heavily skewed toward 5–7 turn conversations, a bias that propagates to our reasoning corpus.

References

[1] Anthropic. Anthropic extended thinking. `https://docs.anthropic.com/en/docs/build-with-claude/extended-thinking`, 2025. Accessed: 2025-04-17.

[2] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[3] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023.

[4] Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2024.

[5] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, Dublin, Ireland, May 2022.

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

[7] Mario Michael Krell, Matej Kosec, Sergio P. Perez, and Andrew Fitzgibbon. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint arXiv:2107.02027*, 2022.

[8] Achintya Kundu, Rhui Dih Lee, Laura Wynter, Raghu Kiran Ganti, and Mayank Mishra. Enhancing training efficiency using packing with flash attention. *arXiv preprint arXiv:2407.09105*, 2024.

[9] Zhenwen Liang, Dian Yu, Wenhao Yu, Wenlin Yao, Zhihan Zhang, Xiangliang Zhang, and Dong Yu. Mathchat: Benchmarking mathematical reasoning and instruction following in multi-turn interactions. *arXiv preprint arXiv:2405.19444*, 2024.

[10] OpenAI. Openai reasoning. `https://platform.openai.com/docs/guides/reasoning`, 2024. Accessed: 2025-04-17.

[11] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, June 2018. URL `https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf`. Accessed: 10-11-2023.

[12] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[13] Traian Rebedea, Makesh Sreedhar, Shaona Ghosh, Jiaqi Zeng, and Christopher Parisien. CantTalkAboutThis: Aligning language models to stay on topic in dialogues. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12232–12252, Miami, Florida, USA, November 2024.

[14] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems*, 2024.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[16] Franklin Wang and Sumanth Hegde. Accelerating direct preference optimization with prefix sharing. *arXiv preprint arXiv:2410.20305*, 2024.

[17] Zezhong Wang, Xingshan Zeng, Weiwen Liu, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. ToolFlow: Boosting LLM tool-calling through natural and coherent dialogue synthesis. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4246–4263, Albuquerque, New Mexico, April 2025.

[18] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[19] Y. Zheng et al. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024.

A Background

A.1 Language Modeling Loss

For a token sequence ($\mathcal{H}_{<i}$, h_i, a_i), the language modeling loss [11] for assistant message a_i can be expressed as:

$$\mathcal{L}(\mathcal{H}_{<i}, h_i, a_i) = -log(P_\Theta(a_i|(\mathcal{H}_{<i}, h_i)) \tag{3}$$

where language model is parameterized by Θ.

B Complexity Analysis

B.1 Input Length

B.1.1 N-Pass Approach

In the N-Pass approach, each turn i is processed in a separate forward pass. The input to the model at turn i is:

$$\mathcal{H}_{<i}, h_i, t_i, r_i$$

because human and assistant response tokens from previous turns remain in the conversation history, while earlier reasoning tokens are discarded.

Let $L_{\textit{N-Pass}}$ denote the maximum input length possible for the N-Pass approach for a conversation c. It can be defined by:

$$L_{\textit{N-Pass}} = \sum_{i=1}^{N}(|h_i| + |r_i|) + max_{i=1}^{N}|t_i|, \tag{4}$$

which is sum of all the human messages and response tokens for entire conversation and maximum length of thinking tokens across turns. To simplify further, assume:

$$|h_i|, |t_i|, |r_i| \in O(\ell).$$

where ℓ denote the characteristic turn component length, defined as $\ell = P_{95}(|h_i|, |t_i|, |r_i| : i \in [1, N], c \in \mathcal{D})$, where P_{95} is the 95th percentile operator. Then:

$$L_{\textit{N-Pass}} \in O\big((2N + 1)\ell\big) = O(N\ell). \tag{5}$$

B.1.2 1-Pass Approach

Our 1-Pass approach processes the entire conversation c in a single forward pass. The input length L_{1-Pass} can be calculated as:

$$L_{\textit{1-Pass}} = \sum_{i=1}^{N} \big(|h_i| + |t_i| + 2|r_i|\big) \in O\big(4N\ell\big) = O\big(N\ell\big). \tag{6}$$

B.2 Time Complexity Analysis

For a transformer with hidden dimension d and context length n, each layer requires $O(n^2 d)$ operations when $n \gg d$ [15].

N-Pass Approach: Under the N-Pass approach, each of the N turns requires a forward pass, each operating on $O(L_{\textit{N-Pass}}) = O(N\ell)$ tokens. Thus, for conversation c:

$$T_{\textit{N-Pass}}(c) \in O\big(N \times (N\ell)^2 d\big) = O\big(N^3 \ell^2 d\big). \tag{7}$$

1-Pass Approach: In the 1-Pass approach, all the conversation tokens are given as input at once, thus operating on $L_{\textit{1-Pass}}$ tokens yielding a cost of:

$$T_{\textit{1-Pass}}(c) \in O\big((4N\ell)^2 d\big) = O\big(N^2 \ell^2 d\big). \tag{8}$$

This represents a factor of N improvement in asymptotic complexity, with substantial gains for large N.

B.3 Memory Complexity Analysis

A transformer layer with input context length n has memory complexity $O(n^2)$ assuming $n \gg d.$

N-Pass Approach: Peak Memory requirement for N-Pass approach is at $L_{\textit{N-Pass}}$ input. Thus for conversation c:

$$\mathrm{M}_{\textit{N-Pass}}(c) \in O\big((2N+1)^2 \ell^2\big) = O\big(N^2 \ell^2\big). \tag{9}$$

1-Pass Approach: Memory requirement for 1-Pass approach can be given by:

$$\mathrm{M}_{\textit{1-Pass}}(c) \in O\big((4N)^2 \ell^2\big) = O\big(N^2 \ell^2\big). \tag{10}$$

Though 1-Pass incurs a higher constant factor due to response token replication, both approaches exhibit identical asymptotic memory complexity.

C Experiments

C.1 Efficient mask generation

We present an efficient algorithm for generating the custom attention mask required by our 1-Pass training method. The algorithm leverages vectorized GPU operations to compute visibility patterns without explicit loops.

Implementation Notes:

- All operations are performed on GPU using PyTorch's vectorized tensor operations
- Role IDs: 0 = padding, 1 = human, 2 = thinking, 3 = response (first copy), 4 = response (second copy)
- The boolean expression in Step 3 is optimized using Karnaugh map reduction to minimize logical operations
- The algorithm avoids explicit loops by leveraging broadcasting and logical operations
- For CPU tensors, we temporarily move computation to GPU before returning results to the original device

Algorithm 1 Efficient Custom Attention Mask Generation

```
Require: Role IDs tensor $\mathbf{R} \in \{0, 1, 2, 3, 4\}^{B \times L}$ where $B$ is batch size, $L$ is sequence length
Ensure: 4D attention mask $\mathbf{M} \in \mathbb{R}^{B \times 1 \times L \times L}$
// Step 1: Compute turn IDs via cumulative sum
$\mathbf{R}_{\text{shift}} \leftarrow \text{roll}(\mathbf{R}, \text{shift} = 1, \text{dim} = 1)$
$\mathbf{R}_{\text{shift}}[:, 0] \leftarrow 0$
$\text{turn\_increment} \leftarrow (\mathbf{R} \neq 0) \wedge (\mathbf{R} = 1) \wedge (\mathbf{R}_{\text{shift}} \neq 1)$
$\mathbf{T} \leftarrow \text{cumsum}(\text{turn\_increment}, \text{dim} = 1)$
$\mathbf{T}[\mathbf{R} = 0] \leftarrow 0$ {Zero out padding positions}

// Step 2: Create base causal non-padding mask
$\mathbf{i} \leftarrow [0, 1, \dots, L - 1]$
$\text{non\_pad} \leftarrow (\mathbf{R} \neq 0)$
$\mathbf{M}_{\text{base}} \leftarrow (\mathbf{i}[:, \text{None}] \geq \mathbf{i}[\text{None}, :]) \wedge \text{non\_pad}[:, :, \text{None}] \wedge \text{non\_pad}[:, \text{None}, :]$

// Step 3: Apply role-specific visibility constraints (K-map optimized)
$\text{turn\_equal} \leftarrow (\mathbf{T}[:, :, \text{None}] = \mathbf{T}[:, \text{None}, :])$
$\mathbf{R}_i \leftarrow \mathbf{R}[:, :, \text{None}]; \mathbf{R}_j \leftarrow \mathbf{R}[:, \text{None}, :]$
$\mathbf{M}_{\text{final}} \leftarrow \mathbf{M}_{\text{base}} \wedge \big[(\mathbf{R}_j = 1) \vee (\mathbf{R}_j = 4 \wedge \text{turn\_equal})$
                        $\vee (\mathbf{R}_j = 3 \wedge \mathbf{R}_i \neq 4) \vee (\mathbf{R}_j = 3 \wedge \neg\text{turn\_equal})$
                        $\vee (\mathbf{R}_j = 2 \wedge \text{turn\_equal} \wedge \mathbf{R}_i \neq 3)\big]$

// Step 4: Convert to 4D attention weights
$\mathbf{M} \leftarrow \text{where}(\mathbf{M}_{\text{final}}.\text{unsqueeze}(1), 0, -\infty)$
return $\mathbf{M}$
```

C.2 Experimental Setup

All training runs are initiated using llamafactory-cli in SFT mode. We apply QLoRA with 4-bit NF4 quantization, using a LoRA rank of 32 and a scaling factor of $\alpha = 64$. Training is performed for three epochs with bfloat16 (bf16) precision.

We enable the Liger kernel for improved efficiency. Each GPU processes a batch size of 2, with gradient accumulation over 4 steps. This setup yields an effective batch size of 64 across the 8-GPU node.

C.3 Comprehensive Results

We report the complete numerical results that support the figures in Section 3 in Tables 2, 3 and 4. We report two metrics for every configuration:

- **Throughput** ("samples per sec.") — the average number of *full conversations* processed per second.
- **Peak GPU memory** — the peak memory recorded during training.

C.3.1 Implementing K-Pass Processing

To obtain the results in Table 3 we extend our Optimised 1-Pass scheme to an intermediate K*-Pass* schedule. Assume a conversation contains N assistant turns $(h_1, t_1, r_1), \dots, (h_N, t_N, r_N)$.

(a) **Chunking the dialog.** We partition the conversation into K contiguous chunks, each containing $\lceil N/K \rceil$ turns (the last chunk may be shorter).

(b) **Selective token duplication.** Within the *current* chunk we apply the same response-token duplication as in Section 2.2: $r_i^{\text{in}}, r_i^{\text{out}}$. All earlier chunks act purely as context and therefore retain their original, non-duplicated responses. This progressively lowers the number of duplicated tokens as K increases, which is the main source of the memory savings reported in Table 3.

Model Size	Run Setting	Samples per sec.	Peak Memory(GB)	Relative Speedup	Relative Peak Memory
4B	FA2-N-Pass(Baseline)	1.985	9	1.0	1.00
	FA2-Pack-N-Pass	6.241	9	3.1	1.00
	Flex Atten-N-Pass	1.286	9	0.6	1.00
	Flex Atten+Packing-N-Pass	4.550	9	2.3	1.00
	Flex-1-Pass	2.107	12	1.1	1.33
	Flex-Pack-1-Pass	6.552	12	3.3	1.33
8B	FA2-N-Pass(Baseline)	2.307	14	1.0	1.00
	FA2-Pack-N-Pass	4.522	14	2.0	1.00
	Flex-N-Pass	1.365	14	0.6	1.00
	Flex-Packing-N-Pass	3.561	14	1.5	1.00
	Flex-1-Pass	1.736	18.8	0.8	1.34
	Flex-Pack-1-Pass	5.484	18.8	2.4	1.34
32B	FA2-N-Pass(Baseline)	0.601	34	1.0	1.00
	FA2-Pack-N-Pass	1.299	34	2.2	1.00
	Flex-N-Pass	0.465	34	0.8	1.00
	Flex-Packing-N-Pass	1.078	34	1.8	1.00
	Flex-1-Pass	0.521	44	0.9	1.29
	Flex-Pack-1-Pass	1.578	44	2.6	1.29

Table 2: **Throughput and peak memory across execution strategies.** FA2 = FlashAttention 2; Flex = FlexAttention. Pack denotes dynamic sequence-packing; “1-Pass” is our proposed approach. Relative columns are computed with respect to the corresponding FA2–N-Pass baseline.

Model Size	K	Samples per sec.	Peak Memory(GB)	Relative Speedup	Relative Peak Memory
4B	N-Pass(baseline)	4.55	9	1.00	1.00
	6-Pass	3.89	10.8	0.85	1.20
	4-Pass	4.76	11.5	1.05	1.28
	2-Pass	5.91	11.8	1.30	1.31
	1-Pass	6.55	12	1.44	1.33
8B	N-Pass(baseline)	3.56	14	1.00	1.00
	6-Pass	3.13	16	0.88	1.14
	4-Pass	3.87	16.4	1.09	1.17
	2-Pass	4.87	17	1.37	1.21
	1-Pass	5.48	18.8	1.54	1.34
32B	N-Pass(baseline)	1.08	34	1.00	1.00
	6-Pass	0.88	39	0.82	1.15
	4-Pass	1.08	40	1.00	1.18
	2-Pass	1.37	41	1.27	1.21
	1-Pass	1.58	44	1.46	1.29

Table 3: **Speed–memory trade-off as a function of K.** Each dialogue is split into K equal-length chunks that are processed sequentially in a *single* forward/backward pass. $K = N$ corresponds to the per-turn baseline, while $K = 1$ is our single-pass method. All experiments use the FlexAttention backend with sequence packing (Flex-Pack), the configuration that achieved the best overall speed in our primary evaluation.

(c) **Attention and position IDs.** The custom attention mask and position-ID assignment described in Section 2.2 are applied *only* to the duplicated tokens of the active chunk. Context tokens keep the standard causal mask.

(d) **Loss computation.** The label mask is set to `1` for t_i and r_i^{out} *inside* the active chunk and `0` elsewhere, so each pass trains only on the new turns while reusing earlier content as fixed context.

Conceptually, the K-Pass schedule interpolates between the extremes:

	Run Setting	Samples per sec.	Peak Memory(GB)	Relative Speedup	Relative Peak Memory
Group 1	FA2-N-Pass(Baseline)	2.54	14	1.00	1
	FA2-Pack-N-Pass	6.93	14	2.73	1
	Flex-N-Pass	2.32	14	0.91	1
	Flex-Packing-N-Pass	4.94	14	1.94	1
	Flex-1-Pass	1.74	18.8	0.69	1.34
	Flex-Pack-1-Pass	6.43	18.8	2.53	1.34
Group 2	FA2-N-Pass(Baseline)	1.02	14	1	1
	FA2-Pack-N-Pass	2.39	14	2.34	1
	Flex-N-Pass	0.87	14	0.86	1
	Flex-Packing-N-Pass	2.10	14	2.06	1
	Flex-1-Pass	1.07	18.8	1.05	1.34
	Flex-Pack-1-Pass	2.86	18.8	2.80	1.34
Group 3	FA2-N-Pass(Baseline)	1.06	14	1	1
	FA2-Pack-N-Pass	2.28	14	2.15	1
	Flex-N-Pass	0.65	14	0.61	1
	Flex-Packing-N-Pass	1.75	14	1.65	1
	Flex-1-Pass	1.66	18.8	1.56	1.34
	Flex-Pack-1-Pass	2.81	18.8	2.65	1.34

Table 4: **Impact of conversation depth (Qwen-3 8B).** Group 1 (1–5 turns), Group 2 (6–7 turns), and Group 3 (8–16 turns). Our 1-Pass approach gains more speed as depth increases, in line with the theoretical $O(N^2)$ vs. $O(N^3)$ complexity gap.

- $K = N$ reproduces the per-turn baseline (no response duplication, minimal memory, maximal passes);
- $K = 1$ is our 1-Pass method (maximum duplication, single pass, fastest).