LANEROPE: POSITIONAL ENCODING FOR COLLABORATIVE PARALLEL REASONING AND GENERATION

Gabriele Cesa, Thomas Hehn, Aleix Torres-Camps, Àlex Batlle Casellas
Qualcomm AI Research*
gcesa@qti.qualcomm.com

Jordi Ros-Giralt, Arash Behboodi & Tribhuvanesh Orekondy
Qualcomm AI Research*

ABSTRACT

Parallel LLM test-time scaling techniques (e.g., best-of-N) require drawing $N > 1$ sequences conditioned on the same input prompt. Such techniques demonstrate improved accuracy and since the N generations are batched, they better utilize compute cores. However, traditionally, each sequence in the batch is generated independently and hence do not reuse compute, intermediate generations, or observations between sequences. In this paper, we propose LaneRoPE to enable coordination and collaboration between $N > 1$ sequences at generation time. LaneRoPE involves two key ideas: (a) an inter-sequence attention mask to make sampling of sequences dependent on one another; and (b) a RoPE extension to inject positional information that captures relative positions between tokens, both within and outside a particular sequence. We evaluate our approach on mathematical reasoning tasks and find promising results: LaneRoPE enables collaboration among sequences, yielding additional accuracy gains under limited generated sequence length. Importantly, since LaneRoPE enables coordination with minimal changes to the underlying LLM architecture and introducing negligible new learnt parameters, it is appealing to rapidly introduce collaborative reasoning in existing LLM inference pipelines.

1 INTRODUCTION

Parallel test-time scaling methods show how accuracy can be reliably improved with additional inference-time compute (Cobbe et al., 2021; Brown et al., 2024; Wu et al., 2025; Snell et al., 2024; Wang et al., 2022) and thereby make this a powerful technique to boost a fixed LLM's performance without retraining. The crux of such methods is sequentially generating *multiple* responses (each potentially with its own reasoning trace) for a given input, and optionally ranking and aggregating responses with a reward model. The multiple responses are typically generated in parallel during the same forward pass (i.e., batched inference), and benefit from a hardware efficiency standpoint, since this amortizes memory transfers, minimizes kernel launch overheads, and better utilizes compute cores at each generation step. Despite its efficiency, parallel inference is largely *uncoordinated*: for a given input, the output sequences are generated independently, which prohibits exploiting inherent parallelizable decomposable structures (Yang et al., 2025) and limits re-using reasoning or intermediate outputs and generating diverse responses.

Building on top of powerful parallel inference, how can we make multiple sequences coordinate with each other during generation? Two families of approaches have gained traction, each with their inherent trade-offs. The first family of approaches leverage 'explicit branching' (Gandhi et al., 2024; Pan et al., 2025; Yang et al., 2025; Zheng et al., 2025) mechanisms: at inference time, a single sequence adaptively fans-out to multiple search paths (e.g., one path per decomposed subproblem) and funneling the paths to result in a single response. Although they encourage the LLM to explicitly

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

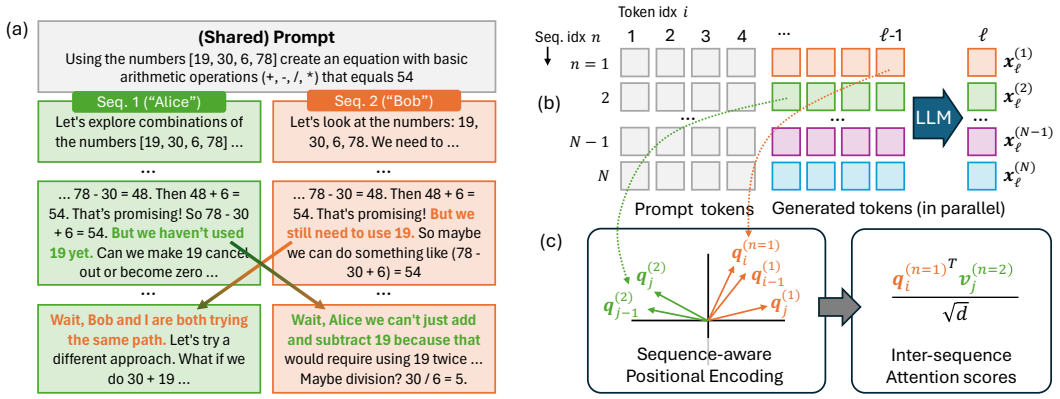


Figure 1: **LaneRoPE: High-level introduction and intuition.** (a) We investigate the problem of *collaborative reasoning*, where given a single input prompt, parallel sequences can reason by conditionally attending to other sequences mid-generation; (b) Our work assumes the tokens across sequences are generated in parallel with the same model, and thereby benefit from *batched efficiency*; (c) A key contribution of our work is introducing *cross-lane attention*, such that attentions can be calculated across sequences (i.e., *lanes*). To enable this, we introduce a novel position encoding scheme to account for relative distances between tokens of different sequences.

decompose tasks, the dynamic nature of how variable number of parallel processes are spawned and merged introduces challenges in predicting runtime compute resources and more importantly, they depend on specialized run-time engines. The second family of 'collaborative' approaches (Hsu et al., 2025; Rodionov et al., 2025; Dong et al., 2025), under which our approach falls, introduce a more flexible means of coordination primarily by enabling shared visibility (1a) into intermediate generations of all parallel sequences. These techniques are practically attractive, since they rely on a fixed number of generation sequences (i.e., constant batch size; see Fig. 1b) and can hence largely leverage batching support by both LLM inference engines and the underlying hardware. In particular, existing techniques (Hsu et al., 2025; Rodionov et al., 2025) exploit an interesting insight that multiple sequences can be 'virtually ordered' into a single sequence by reparameterizing the positions or positional encodings. However, we argue that this is inherently a multi-sequence generation problem and inference needs to accurately reflect positions both along the token dimension (as typically done) *and* sequence (or *lane*) dimension. In the rest of this manuscript, we use *lane* and *sequence* interchangeably to emphasize batch elements generated simultaneously and inter-dependently.

In this work, we propose LaneRoPE to better enable parallel reasoning and generation. Similar to parallel test-time scaling techniques (Wang et al., 2022; Wu et al., 2025) and collaborative reasoning (Hsu et al., 2025; Rodionov et al., 2025; Dong et al., 2025), we assume a constant number of sequences (Fig. 1b) generated at inference. Our approach involves two key insights. First, we make intermediate reasoning traces and outputs visible across sequences by conditioning the next token on previous tokens from *all sequences*. We achieve this by using causal cross-sequence attention masks and generalizing attention score calculations to extend to inter-sequence tokens. Second, we extend rotary positional encoding 'RoPE' (Su et al., 2021) to additionally capture relative distances among sequences (Fig. 1b), allowing us to learn the attention structure across parallel generations in a dynamic and task-dependent manner. Our formulation is expressive enough to capture a range of parallel inference techniques, ranging from independent sampling (e.g., self-consistency) to the fully dense attention masks used in GroupThink (Hsu et al., 2025).

Our evaluation of LaneRoPE on standard mathematical reasoning datasets (e.g., AIME, MATH500) and multiple open-weight LLMs show promising results. Overall, we find that the LLMs demonstrate a reliable accuracy improvement across multiple benchmarks: in particular, a `DeepSeek-R1-Distill-Qwen-7B` model shows an average `Pass1` improvement of over 20% when equipped with LaneRoPE. The flexibility in our formulation enables LaneRoPE to be integrated into existing models with minimal modification and, importantly, in contrast to prior work (Hsu et al., 2025; Rodionov et al., 2025), admits optional finetuning. Furthermore, for finetuning, we

introduce very few additional parameters ($0.7 - 1.3\%$ additional) over base models and these can be fit with simple SFT. In summary, we find LaneRoPE takes a promising step towards enabling parallel *collaborative* reasoning by fundamentally exploiting relative positions of tokens, both within and outside a particular sequence.

Contributions We summarize our contributions as:

- A novel method to enable *fine-grained token-level collaboration* during batched parallel generation of language models. Notably, our method encompasses some existing techniques but its superior flexibility and generality allow for further fine-tuning.
- A recipe to synthesize *collaborative reasoning traces* which can be used for supervised fine-tuning (SFT).
- A fine-tuning pipeline to unlock and strengthen the collaborative reasoning capabilities of pre-trained models.

2 Related Works

LLMs, Reasoning and Test-time Search. Large language models (LLMs) have attracted significant attention largely attributed to remarkable performance in a range of tasks. A range of these tasks (e.g., math, coding) typically test the LLM's ability in generalization, abstractions, and multi-step problem solving. Especially for such tasks, reasoning (Wei et al., 2022; Kojima et al., 2022) has proven to be a powerful paradigm and significantly improved success rates at solving challenging problems. This class of techniques can further be extended to explore various reasoning paths before concluding with a final solution. The reasoning search paths can take various structures, but can broadly be classified into sequential (Muennighoff et al., 2025), graph-like (Yao et al., 2023; Besta et al., 2024), or parallel (Wang et al., 2022; Brown et al., 2024). In this work, we primarily study reasoning as a parallel search problem.

Parallel Reasoning: Independent and Inter-dependent. A popular strategy to scale compute at test-time to improve accuracies is with parallel generations (Wang et al., 2022; Cobbe et al., 2021). For a given problem, multiple sequences (potentially chain-of-thought reasoning followed by a solution) from the same LLM are sampled in parallel and ranked (e.g., using a reward model) to determine the final solution. Prior works (Wang et al., 2022; Cobbe et al., 2021; Brown et al., 2024; Snell et al., 2024; Wu et al., 2025) primarily demonstrate effectiveness where the sequences are sampled *independently*. This however leads to inefficient use of compute, since there is little collaboration between sequences and risks redundant compute between sequences. A new recent line of work explores whether the sequences can be sampled *inter-dependently* (Dong et al., 2025), such that LLM can decide to adaptively parallelize and coordinate *during* generation. This line of inter-dependent generation broadly falls under two categories: (a) 'adaptive branching' (Jin et al., 2025; Pan et al., 2025; Qi et al., 2025; Yang et al., 2025; Chen et al., 2025; Zheng et al., 2025; Wen et al., 2025; Lian et al., 2025) where at generation time, special tokens (e.g., `fork` and `merge`) signal the inference engine to spawn or merge multiple parallel sequence generations; and (b) 'fine-grained collaboration', as in Hsu et al. (2025); Rodionov et al. (2025) and recently Dong et al. (2025), where sequence generation is always parallel (constant batch size) and each sequence can attend to part or all of the tokens generated by other sequences using dynamic attention masks. Our approach falls into the second category, and unlike previous works we introduce a flexible framework to positionally encode cross-sequence tokens which additionally account for relative inter-sequence distances.

Positional Encoding in LLMs Large language models (LLMs) incorporate *positional encodings* to inject sequence order information into transformer architectures, since the self-attention mechanism alone is permutation-invariant (Vaswani et al., 2017). The original Transformer used fixed sinusoidal position encodings (Vaswani et al., 2017), while subsequent models like BERT adopted learned positional embeddings (Devlin et al., 2019), both providing each token with an absolute position bias. To improve generalization beyond fixed-length contexts, *relative positional encodings* were introduced to encode pairwise token distances directly in self-attention (Shaw et al., 2018), allowing the model to focus on relative order. More recent approaches modify how positional information enters the attention mechanism to enable better extrapolation to longer sequences. For example, *rotary positional encodings* (RoPE) rotate query/key vectors by angle proportional to token index,

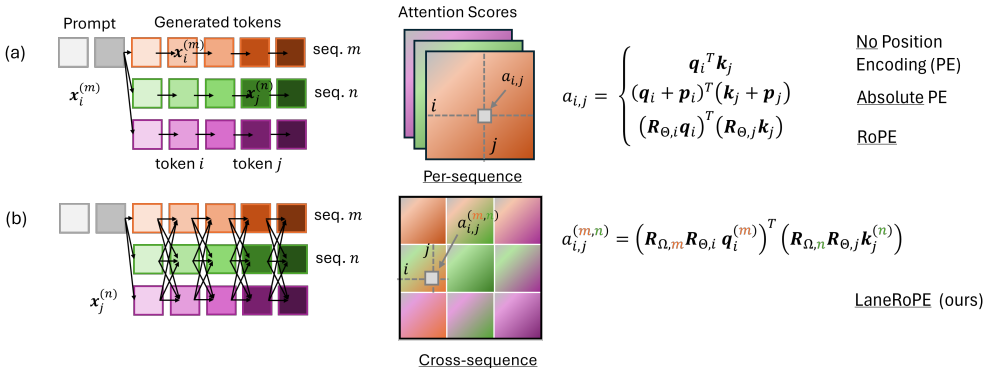


Figure 2: Comparison of RoPE (Su et al., 2021) and LaneRoPE for parallel inference. (a) Parallel inference (e.g., best-of-N) relies in generating sequences independently. As a result, positional encodings and attention scores are defined only per-sequence. (b) With LaneRoPE, tokens are generated by causally attending to tokens from all sequences. We achieve this by introducing cross-sequence attention scores with a novel position encoding scheme to account for inter-sequence relative distances.

effectively combining absolute and relative positional information (Su et al., 2021). Similarly, *ALiBi* (Attention with Linear Biases) eliminates explicit embeddings and instead adds a distance-dependent bias to attention scores, allowing models trained on shorter sequences to generalize to much longer inputs (Press et al., 2022). In our work, we extend RoPE (Su et al., 2021) to additionally model relative distances between sequences.

3 LaneRoPE: Cross-lane positional encoding

In this section, we present our approach LaneRoPE to enable collaborative parallel inference. We begin by covering preliminaries, and then later technically detail our approach.

3.1 Preliminaries

The self-attention layer of a standard transformer (Vaswani et al., 2017) takes as inputs a set of tokens $\mathbb{T} = \{t_i\}_{i=1}^{L}$. Each token is represented by its corresponding embedding $\mathbb{E} = \{\boldsymbol{x}_i\}_{i=1}^{L}$, where $\boldsymbol{x}_i$ is a d^{emb}-dim embedding of the i-th token without positional information. The self-attention layer converts the embeddings into query, key, and value representations, typically via a linear transformation over positionally-encoded embeddings:

$$\boldsymbol{q}_i = f_q(\boldsymbol{x}_i, i),\ \boldsymbol{k}_j = f_k(\boldsymbol{x}_j, j),\ \boldsymbol{v}_j = f_v(\boldsymbol{x}_j, j) \tag{1}$$

The attention logits $a_{i,j}$ and outputs $\boldsymbol{o}_i$ are calculated as:

$$a_{i,j} = \boldsymbol{q}_i^\top \boldsymbol{k}_j \tag{2}$$

$$\alpha'_{i,j} = \text{softmax}_j\left(a_{i,j}/\sqrt{d}\right), \quad \boldsymbol{o}_i = \sum_{j=1}^{L} \alpha'_{i,j} \boldsymbol{v}_j \tag{3}$$

Positional Encodings and RoPE Positional encodings are typically injected when extracting query and key representations in Eq. 1, so that the outputs in Eq. 3 are position-dependent. A traditional choice (Vaswani et al., 2017) is using *absolute* positional encodings as $f_{\{q,k\}}(\boldsymbol{x}_i, i) = \boldsymbol{W}_{\{q,k\}}(\boldsymbol{x}_i + \boldsymbol{p}_i)$, with $\boldsymbol{p}_{i,2t} = \cos\left(\frac{i}{10000^{2t/d}}\right)$ and $\boldsymbol{p}_{i,2t+1} = \sin\left(\frac{i}{10000^{2t/d}}\right)$. However, we often want attention scores to depend on relative offsets between tokens because most token interactions are *translation-invariant*. RoPE (Su et al., 2021) enforces relative offsets, such that attention logits $a_{i,j}$ between $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ (Eq. 2) depends on positions only through $i - j$:

$$f_q(\boldsymbol{x}_i, i)^T f_k(\boldsymbol{x}_j, j) = g(\boldsymbol{x}_i, \boldsymbol{x}_j, i - j) \tag{4}$$

RoPE achieves this property by splitting dimensions into $d/2$ independent 2D planes and rotating each plane with its own frequency θ_ℓ. This action can be represented as a linear projection using a block-diagonal rotation matrix:

$$f_{\{q,k\}}(\boldsymbol{x}_i, i) = \boldsymbol{R}_{\Theta,i}\left(\boldsymbol{W}_{\{q,k\}}\boldsymbol{x}_i + \boldsymbol{b}_{\{q,k\}}\right) \tag{5}$$

with

$$\boldsymbol{R}_{\Theta,i} = \operatorname{diag}\left(R(i\theta_1),\, R(i\theta_2),\, \ldots,\, R(i\theta_{d/2})\right) \tag{6}$$

$$R(i\theta_\ell) = \begin{bmatrix} \cos(i\theta_\ell) & \sin(i\theta_\ell) \\ -\sin(i\theta_\ell) & \cos(i\theta_\ell) \end{bmatrix} \quad \ell = 1, \ldots, d/2$$

such that

$$\boldsymbol{q}_i^\top \boldsymbol{k}_j = (\boldsymbol{W}_q\boldsymbol{x}_j + \boldsymbol{b}_q)\boldsymbol{R}_{\Theta,i-j}(\boldsymbol{W}_k\boldsymbol{x}_i + \boldsymbol{b}_k)\,.$$

Parallel Test-time Scaling Parallel test-time scaling techniques (Brown et al., 2024; Snell et al., 2024; Wu et al., 2025) have gained interest recently, since accuracy of the same base LLM can generally be improved at inference-time. The techniques typically follow a proposal-verifier strategy: for a single input prompt $\boldsymbol{q}$, multiple ($N > 1$) sequences $\{(\boldsymbol{x}_{n,1}, \ldots, \boldsymbol{x}_{n,L})\}_{n=1}^N$ are first sampled from the base 'proposer' LLM. The quality of sequences are then ranked, popularly by 'majority vote' (Wang et al., 2022) (i.e., frequency of solutions), or by using a 'verifier' (Cobbe et al., 2021; Lightman et al., 2024). Importantly, prior works predominantly rely on *independently* sampling $N > 1$ sequences.

3.2 LaneRoPE: Sequence-aware Positional Encoding

Notation: multi-sequence generation We consider the case where for the same input prompt, a set of $N > 1$ sequences $\{\boldsymbol{x}^{(n)}\}_{n=1}^N$ (with the n-th sequence containing an ordered sequence of tokens $\boldsymbol{x}^{(n)} = [\boldsymbol{x}_1^{(n)}, \boldsymbol{x}_2^{(n)}, \ldots]$) are auto-regressively generated in parallel. While the standard assumption is that each sequence is generated independently (Eq. 7),

$$p(\boldsymbol{x}_{i+1}^{(n)} \mid \texttt{prompt},\ \boldsymbol{x}_{1:i}^{(n)}) \tag{7}$$

we consider inter-dependent generation (Eq. 8)

$$p(\boldsymbol{x}_{i+1}^{(n)} \mid \texttt{prompt},\ \{\boldsymbol{x}_{1:i}^{(m)}\}_{m=1}^N) \tag{8}$$

To do so, we adapt the attention mask in our architectures to implement a *causal inter-sequence attention* mechanism, which allows tokens in one sequence m to attend to another sequence n:

$$a_{i,j}^{(m,n)} = {\boldsymbol{q}_i^{(m)}}^\top \boldsymbol{k}_j^{(n)} \tag{9}$$

$$\alpha'^{(m,n)}_{i,j} = \operatorname{softmax}_j\left(a_{i,j}^{(m,n)}/\sqrt{d}\right) \tag{10}$$

Because tokens generated at the same time step across sequences share identical positional indices, standard positional encoding cannot distinguish their sequence identity.

Cross-sequence Positional Encoding To solve this problem, we draw inspiration from RoPE (Su et al., 2021), as a successful solution to capturing positional encoding within a sequence, and apply it along the sequences dimension:

$$\boldsymbol{q}_i^{(m)} = f_q(\boldsymbol{x}_i^{(m)}, i, m) \qquad (i\text{-token},\ m\text{-th sequence})$$

$$\boldsymbol{k}_j^{(n)} = f_k(\boldsymbol{x}_j^{(n)}, j, n) \qquad (j\text{-token},\ n\text{-th sequence}) \tag{11}$$

$$f_{\{q,k\}}(\boldsymbol{x}_i, i, m) = \boldsymbol{R}_{\Omega,m}\boldsymbol{R}_{\Theta,i}(\boldsymbol{W}_{\{q,k\}}\boldsymbol{x}_i^{(m)} + \boldsymbol{b}_{\{q,k\}}) \tag{12}$$

The block-diagonal orthogonal rotation matrices $\boldsymbol{R}_\Theta$ encoding the token position in RoPE (Eq. 6) are now applied together with similar block-diagonal rotation matrices $\boldsymbol{R}_\Omega$ encoding the index of the sequence a token belongs to, with frequencies $\Theta = \{\theta_t |\ t \in 1, 2, \ldots, d/2\}$ and $\Omega = \{\omega_t |\ t \in 1, 2, \ldots, d/2\}$. A useful property of this formulation is the associative property of the rotation matrices, i.e. $R(\omega_t m)R(\theta_t i) = R(\omega_t m + \theta_t i)$, which ensures LaneRoPE can be easily integrated into any architecture using RoPE, as the additional rotation matrices can be merged and applied together with the existing ones from RoPE. Moreover, we can further draw an analogy with Fourier analysis: while RoPE leverages a 1D Fourier basis to help the attention layer express flexible functions along the 1-dimensional token sequence, LaneRoPE leverages a 2D Fourier basis to express flexible functions over the 2-dimensional grid of tokens from different sequences.

3.3 LANEROPE: PROPERTIES AND INITIALIZATION STRATEGIES

Thanks to its flexibility, LaneRoPE can express some existing parallelization strategies, but can also go beyond if trained. This suggests using one (or a combination) of these strategies to *initialize LaneRoPE from pre-trained models*, enabling it to leverage their native powerful reasoning and cooperation capabilities.

Special Case: GroupThink (Hsu et al., 2025) GroupThink uses the same attention mask and generation strategy (Eq. 8) of LaneRoPE. To distinguish tokens from different sequences, given a maximum token budget K for each sequence ("sequence gap"), it "virtually orders" the tokens of N parallel sequences into a single sequence, assigning a token $\boldsymbol{x}_i^{(m)}$ to the location $Km + i$, which is then used for RoPE. Thanks to the translation-invariance of RoPE, the absolute location is irrelevant and tokens from the same sequence appear close in the attention, while other sequences' tokens in the Key-Values (KV) cache appear further away in the context. Note that this strategy forces a pre-trained model out-of-distribution in part, since each token attends also to tokens generated in other sequences and, in particular, tokens from a sequence with a higher index $n > m$ appear at a negative virtual relative position[1] when attended to by a token in the sequence m (which is unseen during pre-training due to the causal attention mask). Despite this, (Hsu et al., 2025) show some promising results, suggesting some level of robustness of pre-trained models to this artifact.

Next, we demonstrate that this strategy can be exactly implemented in LaneRoPE. Indeed, by using the properties of rotation matrices, the RoPE matrix $\boldsymbol{R}_{\Theta,Km+i}$ used to encode the token $\boldsymbol{x}_i^{(m)}$ can be decomposed as

$$\boldsymbol{R}_{\Theta,Km+i} = \boldsymbol{R}_{\Theta,Km}\boldsymbol{R}_{\Theta,i} = \boldsymbol{R}_{K\Theta,m}\boldsymbol{R}_{\Theta,i} = \boldsymbol{R}_{\Omega,m}\boldsymbol{R}_{\Theta,i} \tag{13}$$

where we defined $\Omega := \{\omega_t = K\theta_t\}$.

Special Case: Parallel Independent Sampling Assume a pre-trained model with weights $\boldsymbol{W}_{\{q,k\}}$ and $\boldsymbol{b}_{\{q,k\}}$ in the query and key projection layers at a certain attention head. The independent-sampling attention can be written in our generation strategy (Eq. 8) as

$$a_{i,j}^{(m,n)} = \left(\boldsymbol{W}_q\boldsymbol{x}_i^{(m)} + \boldsymbol{b}_q\right)^\top \left(\boldsymbol{W}_k\boldsymbol{x}_j^{(n)} + \boldsymbol{b}_k\right) + \beta(m-n) \tag{14}$$

where $\beta(x) \propto \delta_{[x=0]}$ replicates an attention bias: if $\beta(0) \gg 0$ such that this term prevails on the key-query inner product whenever $m - n \neq 0$, $a_{i,j}^{(m,n)}$ remains sufficiently large after softmax only if $m = n$. While we can not directly implement this attention bias β in LaneRoPE, this result can be well approximated in it. Indeed, we can consider a function β approximating this δ-function via a discrete Fourier transform and write it as[2]

$$\beta(x) := \sum_t^{F/2} \hat{\beta}_t^c \cos(\omega_t x 2\pi) + \hat{\beta}_t^s \sin(\omega_t x 2\pi) \tag{15}$$

for a discrete set of $F/2$ frequencies $\boldsymbol{\omega} = \{\omega_t\}_t^{F/2}$. By defining the vector $\hat{\boldsymbol{\beta}}$ of size F with coefficients $\{\sqrt{\hat{\beta}_t^{\{c,s\}}}\}_t$, the attention bias takes a convenient form

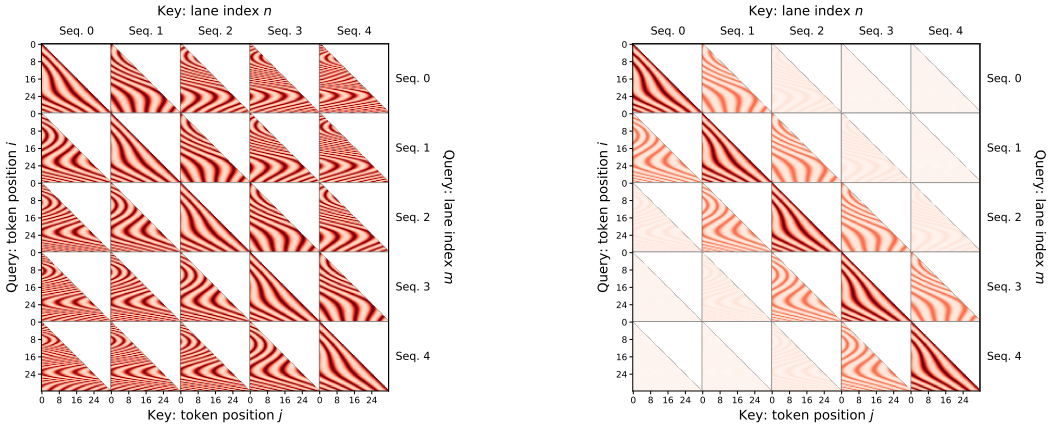
$$\beta(x) := \hat{\boldsymbol{\beta}}^T R_{\boldsymbol{\omega},x}\hat{\boldsymbol{\beta}} \tag{16}$$

Next, we can incorporate this inner product into LaneRoPE by adopting a larger architecture obtained by replacing the linear layer as:

$$\boldsymbol{W}'_{\{q,k\}} := \begin{pmatrix} \boldsymbol{W}_{\{q,k\}} \\ \boldsymbol{O} \end{pmatrix}, \; \boldsymbol{b}'_{\{q,k\}} := \begin{pmatrix} \boldsymbol{b}_{\{q,k\}} \\ \hat{\boldsymbol{\beta}} \end{pmatrix} \tag{17}$$

[1] For this reason, Hogwild! (Rodionov et al., 2025) takes a very similar approach but re-orders the sequences dynamically such that the sequence of the querying token is always last and, therefore, no negative relative positions emerge. However, this solution requires ad-hoc efficient implementations of the attention operator and differs from widely adopted transformers architectures.

[2] Note that β is symmetric, so the weights associated to the $\sin$ are 0. We can also safely assume all the weights $\hat{\beta}_t$ are non-negative.



(a) $\beta(0) = 0$, i.e. $|\hat{\boldsymbol{\beta}}|_2 = 0$

(b) $\beta(0) > 0$, i.e. $|\hat{\boldsymbol{\beta}}|_2 > 0$

Figure 3: Comparing the effect of the attention bias β incorporated into LaneRoPE on the attention scores. The strength of this effect is tuned by scaling the norm $|\hat{\boldsymbol{\beta}}|_2$. The figure shows also the effect of a GroupThink initialization, where tokens from different lanes are concatenated in a virtual position index (rows and columns indices). Note that the causal attention mask of each lane is preserved.

with $\mathbf{0} \in \mathbb{R}^F$ and $\boldsymbol{O} \in \mathbb{R}^{F \times d_{\text{emb}}}$ zero matrices, and correspondingly choosing rotary matrices

$$R_{\Theta',i} := \begin{pmatrix} R_{\Theta,i} & \\ & \boldsymbol{I} \end{pmatrix}, \; R_{\Omega',m} := \begin{pmatrix} R_{\omega',m} & \\ & R_{\omega,m} \end{pmatrix} \tag{18}$$

where $R_{\Theta,i}$ is the RoPE matrix of the pre-trained model while $R_{\omega',m}$ can be any LaneRoPE matrix[3]. The resulting projection layers have the same form as Eq. 12:

$$f_{\{q,k\}}(\boldsymbol{x}_i, i, m) = \boldsymbol{R}_{\Omega',m} \boldsymbol{R}_{\Theta',i} (\boldsymbol{W}'_{\{q,k\}} \boldsymbol{x}_i^{(m)} + \boldsymbol{b}'_{\{q,k\}}) \tag{19}$$

In practice, at initialization, we tune the strength of the attention bias by scaling $\hat{\beta}$ in Eq.17 by a large constant, i.e. directly scaling the norm $|\hat{\boldsymbol{\beta}}|_2$. For example, Fig. 3 illustrates the effect of this attention bias in LaneRoPE on the attention weights when changing its scale.

We emphasize that LaneRoPE is sufficiently expressive to reproduce independent sampling even without the additional dimensions we introduced and that this modification is only a convenient way to achieve independent sampling without modifying the pre-trained model's weights. In practice, few dimensions are sufficient[4] and the overhead is minimal. For example, 8 additional dimensions in all attention heads add only 0.1B new parameters in a `Qwen2.5-7B` architecture with 7.7B parameters.

Finally, while these two presented cases offer useful initialization strategies, LaneRoPE is not limited to them[5]. First, we typically adopt a combination of both strategies (e.g. using $\boldsymbol{\omega}' = K\Theta$ as in GroupThink). Second, $\boldsymbol{W}'_{\{q,k\}}$ and $\boldsymbol{b}'_{\{q,k\}}$ are learnable parameters which can be arbitrarily modified during training and, therefore, are not constrained to represent a fixed attention bias β.

4 Training LaneRoPE

To elicit cross-lane collaboration of a base model, we train the partial model by supervised finetuning (SFT) on a dataset that mimics collaborations.

[3] Any arbitrary frequency $\boldsymbol{\omega}'$ can be chosen since they only play a role when $m \neq n$, but that will be masked by the attention bias β. We typically use the GroupThink-initialization $\boldsymbol{\omega}' := K\Theta$, as discussed later.

[4] For N parallel sequences, $N + 1$ additional dimensions are always sufficient to represent any function on the N sequence indices with a discrete Fourier basis.

[5] $\hat{\boldsymbol{\beta}}$ could be initialized to approximate other functions β on the relative sequence indexes. Exploring these design choices could be an interesting future work.

Dataset Generation We construct this dataset using `Qwen3-30B-A3B-Thinking-2507` (Qwen-Team, 2025), using questions from the *DeepScaleR-Preview-Dataset* (Luo et al., 2025). To simulate collaboration across N lanes using a sequential LLM, we prompt the LLM to play the role of N different assistants in a step-by-step manner. An assistant's message at each step ends upon generation of a newline token or after 64 new tokens. After each step, an assistant is prompted to generate a new message by completing its own answer, pre-filled with the text it generated so far, which follows a user message providing the query and the text previously generated by other assistants. For the final SFT dataset, we separate the text generated by each assistant into individual traces such that each trace only contains the messages of one assistant. As a result, a model fine-tuned on this dataset requires a form of cross-sequence interaction to be able to fit the data. In total, we generate 8 conversations with two assistants for each of the 40315 questions of the dataset, resulting in about 322k raw conversations. See Apx. B for more details.

Dataset Curation To focus on quality over quantity in the SFT dataset, we filter the original generations based on a sequence of criteria. First, we only keep samples where all the assistants give the correct answer. Second, we employ a length limit of 35 messages per lane and remove all samples that reach that limit. Third, we prompt GPT-4o to judge the quality of the conversation of the assistants on a scale between 0 and 1, and only keep the sample if the average score across five repetitions is above 0.7. After this curation process, the final SFT dataset consists of 2721 conversations. See Apx. B.1 for an example of the SFT dataset.

Supervised Fine-tuning To train our models, we first initialize them from open-source pre-trained LLMs using a combination of the initialization strategies described earlier. In particular, we use a "sequence gap" $K = 4096$ to bias the model towards a GroupThink-like behavior but also set a strong positional bias ($|\hat{\boldsymbol{\beta}}|_2 = 1000.$) to encourage the model to attend to tokens in other sequences only when necessary. Then, we train our models for 2 epochs on this curated dataset. We only train model parameters that are directly affected by the modification of LaneRoPE, that is the weight and bias of linear layers of the keys and queries (Eq. 17), as well as the frequencies[6] Ω of the LaneRoPE itself.

5 Experiments

To assess the effectiveness of our proposed method, we conduct evaluations across several reasoning benchmarks and models that measure diverse problem-solving competencies.

5.1 Setup

Datasets Specifically, we benchmark performance on the MATH500 (Lightman et al., 2024), the AMC (Mathematical Association of America, 2023), and AIME 24 and 25 (Mathematical Association of America, 2024; 2025) datasets, all of which consist of challenging mathematical problems.

Models and Baselines We test our method on the fine-tuned reasoning models `DeepSeek-R1-Distill-Qwen` with 1.5B and 7B parameters (DeepSeek-AI, 2025). To compare to existing work, we utilize the official implementation of Hogwild! (Rodionov et al., 2025) and evaluate it on the same datasets and models as LaneRoPE. Further, we report the results from Dong et al. (2025) as a reference. To provide an indication of potential differences in hyperparameters and evaluation implementation, we include the base model performance on our and their evaluation. In all our experiments, we use LaneRoPE over $N = 2$ parallel sequences. Moreover, for each query in each benchmark, we sample $M = 16$ sequences: these are sampled independently in our baselines but, when using LaneRoPE we independently sample $M/N = 8$ groups of $N = 2$ inter-dependent completions (same for Hogwild!). Finally, all generations use a temperature of 0.6 and `top-p` value of 0.95 (as recommended in DeepSeek-AI (2025)) and are limited by 4096 maximum tokens.

[6]For numerical stability, we do not parameterize Ω directly; instead, we train a separate vector $\boldsymbol{\omega}$ of the same dimensionality and set $\Omega = K\,\Theta \cdot 2\big(2\sigma(\boldsymbol{\omega}) - 1\big)$, where $\sigma(\cdot)$ is the sigmoid function, Θ denotes the RoPE frequencies, and K is the GroupThink *sequence gap*. We initialize $\boldsymbol{\omega}$ such that $\Omega = K\Theta$.

Model	MATH-500	AIME24	AIME25	AMC23	Avg.
DS-Qwen-1.5B Dong et al. (2025)	73.65	13.75	13.44	50.00	37.71
+ Bridge Dong et al. (2025)	81.30	20.11	20.00	60.55	45.49
DS-Qwen-1.5B	66.33	10.21	12.50	41.56	32.65
+ HogWild!	58.15	7.50	6.67	43.12	28.86
(LaneRoPE) +SFT	67.08	13.35	11.86	46.58	34.72
DS-Qwen-7B Dong et al. (2025)	82.15	23.44	21.88	66.02	48.37
+ Bridge Dong et al. (2025)	88.15	32.19	25.41	77.65	55.85
DS-Qwen-7B	77.24	19.17	22.08	55.94	43.61
+ HogWild!	69.90	20.00	20.83	57.50	42.06
(LaneRoPE) +SFT	86.19	31.62	25.41	73.28	54.12

Table 1: `Pass@1` score averaged over 16 samples per query across different mathematical reasoning benchmarks. Dong et al. (2025) means these are the numbers reported by Dong et al. (2025). All other numbers are our independent runs.

5.2 Results

In Tab. 1, we report the `Pass@1` score averaged on $M = 16$ samples over different mathematical reasoning datasets. We first note that our fine-tuned LaneRoPE models show consistent improvements over the sequential reasoning baselines. In particular, we observe greater benefits in the larger 7B model, which is in line with the findings in Rodionov et al. (2025), i.e. that larger models have better collaboration capabilities.

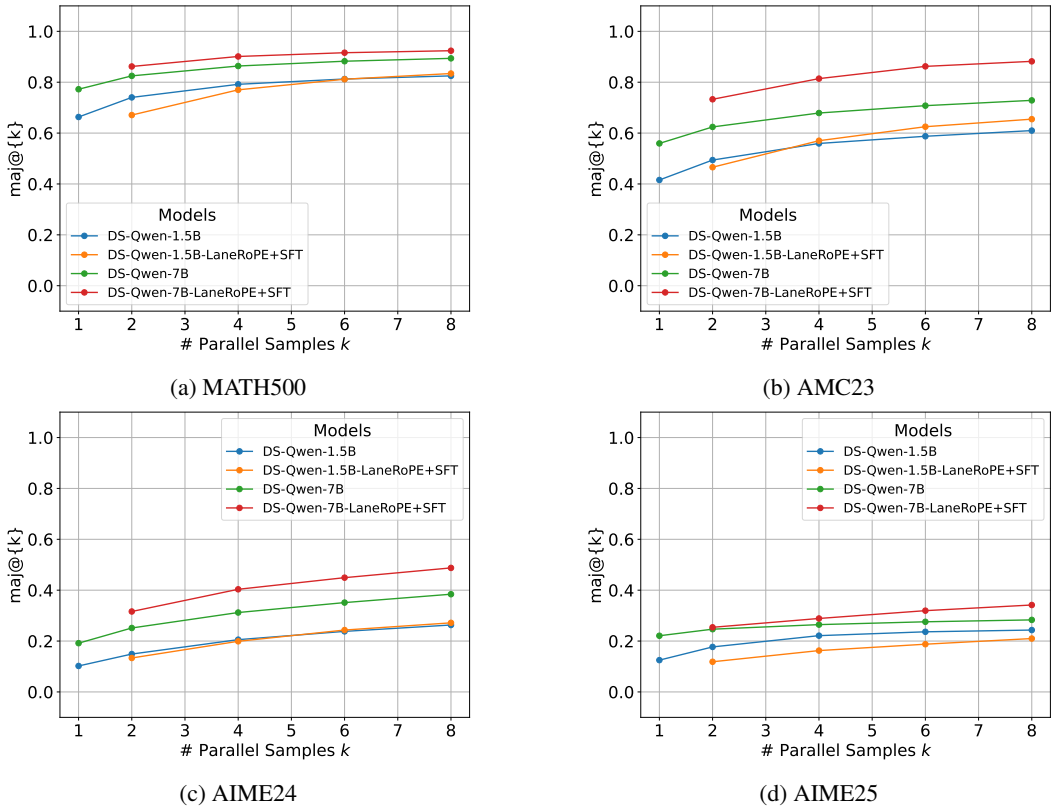
Tab. 1 also reports the results from the recent concurrent work from Dong et al. (2025), which also employs a form of cross-sequence attention and a combination of SFT and RL to reinforce parallel reasoning capabilities. Unfortunately, likely due to some inference hyperparameters differences, our baselines do not exactly match those reported in Dong et al. (2025) (see [DS-Qwen-7B] vs [DS-Qwen-7B Dong et al. (2025)]). Nevertheless, our fine-tuned model ([DS-Qwen-7B (LaneRoPE) + SFT]) typically matches the Bridge baseline, which was trained with a combination of SFT and RL, in terms of absolute performance and often surpasses it in terms of relative gains over the corresponding DS-Qwen-7B base model. Our hypothesis is that this gain on the larger model is thanks to our GroupThink-like initialization, which enables LaneRoPE to better leverage the native collaborative capabilities of foundational pre-trained models, even with little training.

We note that Hogwild! tends to worsen the base model performance. Despite relying on the official implementation, this does not warrant the conclusion that Hogwild! is an ineffective method. Nevertheless, this discrepancy suggests that certain factors may be affecting performance, most notably the use of a different underlying base model than the one originally evaluated. Such a change can alter the dynamics of the Hogwild! prompt and may require significant additional tuning or adaptation to achieve better results.

Finally, we question whether LaneRoPE is more effective than self-consistency based strategies like *majority voting* when considering larger parallel generation budgets k. To answer this we study combinations of LaneRoPE with *majority voting* in Fig. 4. While the smaller 1.5B model shows limited benefits with LaneRoPE over vanilla majority voting, the 7B model shows significant gains from using inter-dependent samples via LaneRoPE, rather than independent samples.

6 Conclusion

In this paper, we addressed the problem of collaborative parallel inference, where multiple sequences (e.g., within the same batch of a forward pass) can access each other's partial generation to condition future reasoning. In particular, we proposed a novel and flexible method - which encompasses some



(a) MATH500 (b) AMC23

(c) AIME24 (d) AIME25

Figure 4: Performance on different mathematical reasoning datasets as a function of the parallelization budget k, when a base model and LaneRoPE are combined with *majority voting*. All LaneRoPE models only leverage $N = 2$ inter-dependent samples and, therefore, generate $k/2$ pairs of completions.

previous approaches - to model token-level inter-sequence interactions and a data-generation recipe to create high quality collaborative reasoning traces for fine-tuning. Our empirical results show improved reasoning capabilities in parallel setups and open the opportunity for better parallel scaling strategies beyond independent sampling.

Limitations and Future Work We do not yet employ a dedicated training scheme for merging outputs from multiple lanes, and developing such mechanisms remains an open area for refinement. Our current experiments focus on $n = 2$ collaborating lanes, though the method itself is designed to extend naturally to a larger number of lanes. Finally, while a small SFT stage with synthetic data proved sufficient to unlock strong parallel reasoning capabilities, an additional refinement stage with Reinforcement Learning with Verifiable Feedback (RLVF), as done in Dong et al. (2025), could unlock novel collaborative behaviors and strategies in an unsupervised way and further strengthen the overall reasoning performance.

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Keyu Chen, Zhifeng Shen, Daohai Yu, Haoqian Wu, Wei Wen, Jianfeng He, Ruizhi Qiao, and Xing Sun. Aspd: Unlocking adaptive serial-parallel decoding by exploring intrinsic parallelism in llms. *arXiv preprint arXiv:2508.08895*, 2025. doi: 10.48550/arXiv.2508.08895. URL https://arxiv.org/abs/2508.08895.

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

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, 2019.

Harry Dong et al. Generalized parallel scaling with interdependent generations. *arXiv preprint arXiv:2510.01143*, 2025. URL https://arxiv.org/abs/2510.01143.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023. URL https://github.com/huggingface/lighteval.

Chan-Jan Hsu, Davide Buffelli, Jamie McGowan, Feng-Ting Liao, Yi-Chang Chen, Sattar Vakili, and Da-shan Shiu. Group think: Multiple concurrent reasoning agents collaborating at token level granularity. In *arXiv preprint arXiv:2505.11107*, 2025. doi: 10.48550/arXiv.2505.11107. URL https://arxiv.org/abs/2505.11107.

Tian Jin, Ellie Y. Cheng, Zack Ankner, Nikunj Saunshi, Jonathan Ragan-Kelley, Suvinay Subramanian, Blake M. Elias, Amir Yazdanbakhsh, and Michael Carbin. Learning to keep a promise: Scaling language model decoding parallelism with learned asynchronous decoding. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://icml.cc/virtual/2025/poster/44837.

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

Long Lian et al. Threadweaver: Adaptive threading for efficient parallel reasoning in language models. *arXiv preprint arXiv:2512.07843*, 2025. URL https://arxiv.org/abs/2512.07843.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://huggingface.co/agentica-org/DeepScaleR-1.5B-Preview, 2025.

Mathematical Association of America. American mathematics competitions (amc) 10/12 2023, 2023. URL https://maa.org/.

Mathematical Association of America. American invitational mathematics examination (aime) 2024, 2024. URL https://maa.org/.

Mathematical Association of America. American invitational mathematics examination (aime) 2025, 2025. URL https://maa.org/.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models. In *Conference on Language Modeling (COLM)*, 2025. URL https://arxiv.org/abs/2504.15466.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations (ICLR)*, 2022.

Jianing Qi, Xi Ye, Hao Tang, Zhigang Zhu, and Eunsol Choi. Learning to reason across parallel samples for llm reasoning. *arXiv preprint arXiv:2506.09014*, 2025. doi: 10.48550/arXiv.2506.09014. URL https://arxiv.org/abs/2506.09014.

Qwen-Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Gleb Rodionov, Roman Garipov, Alina Shutova, George Yakushev, Erik Schultheis, Vage Egiazarian, Anton Sinitsin, Denis Kuznedelev, and Dan Alistarh. Hogwild! inference: Parallel llm generation via concurrent attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2504.06261.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Vol. 2: Short Papers)*, pp. 464–468, 2018.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS 2017)*, pp. 6000–6010, 2017.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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

Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Parathinker: Native parallel thinking as a new paradigm to scale llm test-time compute. *arXiv preprint arXiv:2509.04475*, 2025. doi: 10.48550/arXiv.2509.04475. URL https://arxiv.org/abs/2509.04475.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025.

Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. Multiverse: Your language models secretly decide how to parallelize and merge generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2506.09991.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, and Dong Yu. Parallel-r1: Towards parallel thinking via reinforcement learning. *arXiv preprint arXiv:2509.07980*, 2025. doi: 10.48550/arXiv.2509.07980. URL https://arxiv.org/abs/2509.07980.

A Additional Experiments Details

In all our experiments, we instruct the models to provide an answer within \boxed{}. Hence, for each sequence, we extract the answer from the last occurrence of \boxed{} in the completion and, then, use the Lighteval framework (Habib et al., 2023) to score it.

When computing the majority voting scores in Fig. 4, from each sample of k sequences, we filter out sequences that do not contain any \boxed{} before computing the majority. When using LaneRoPE models with $N = 2$ lanes and a budget of k parallel sequences, we sample K/N independent pairs ($N = 2$) of inter-dependent sequences, i.e. maintaining the same batch size overall.

All generations use a temperature of 0.6 and `top-p` value of 0.95 (as recommended in DeepSeek-AI (2025)) and are limited by 4096 maximum tokens.

B Synthetize Collaborative Reasoning Traces

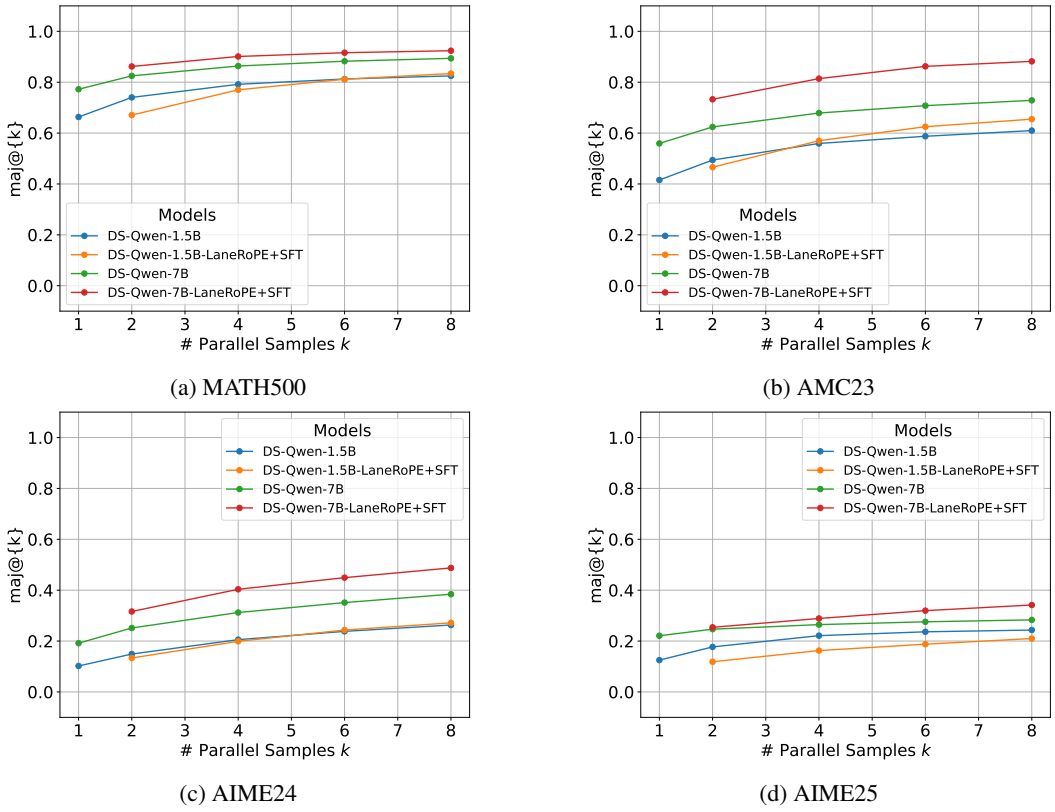
Below are the system prompt used to instruct the model on how the conversation will work and the user prompt used to communicate to the assistant the query and the text generated by the other assistants so far.

At each step, we prompt the model from scratch, pre-filling its own message with the text it generated in previous iterations and adding the text generated by the other assistants in the user prompt. An assistant's message at each step ends upon generation of a newline token or after 64 new tokens. Note also that, since the user message is constantly updated, there is no reuse of the Key-Value cache in consecutive messages. In total, we generate 8 conversations with two assistants for each of the 40315 questions of the dataset, resulting in about 322k raw conversations (pre-filtering). In the final data filtration phase, we prompt GPT-4o to judge the quality of the conversation of the assistants on a scale between 0 and 1, and only keep the sample if the average score across five repetitions is above 0.7.

System prompt

```
You are one of {N} helpful assistants (namely {assistants_list})
collaborating to solve a problem together, while writing their thoughts
in parallel.

Assistants collaborate without redundant work.
Each assistant's thoughts (i.e. what they write within <think> and
</think> tags) are visible to all other assistants while they are being
written.
```



(a) MATH500 (b) AMC23

(c) AIME24 (d) AIME25

Figure 5: Performance on different mathematical reasoning datasets as a function of the parallelization budget k, when a base model and LaneRoPE are combined with *majority voting*. All LaneRoPE models only leverage $N = 2$ inter-dependent samples and, therefore, generate $k/2$ pairs of completions.

```
In other words, as each assistant writes its thoughts, it can see all
the thoughts which are simultaneously being written by all other
assistants as reported in the last user message.
To be precise, you will see other assistant's current thoughts in
previous messages under a "### <assistant-name>'s current thoughts ###"
header.
Finally, the user will communicate you your name and, then, you will
write your own thoughts.

You will take into account what other assistants are doing. If another
assistant gives you suggestions, you should address them.
To keep the communication efficient, decompose long thoughts into short
and atomic thoughts in new lines.
This is also enforced by interrupting lines that are too long and
whenever a new sentence is started (hence, do not use the full stop "."
unless you are done with your thought).

Every sentence and line within the <think> and </think> tags should
focus on reaching the solution collaboratively as efficiently as
possible, since each assistant can only write 40 lines.
You should collaborate with each other by following diverse solution
strategies, doing different parts of the problem, double-checking each
other's results, trying different approaches, or any other means.
```

```
If you realize you are currently doing the same thing that another
assistant has already done or is in process of doing, you acknowledge
it and stop (e.g. Alice may say 'Wait, I was doing the same as Bob...')
and change to a different task right away, so as to avoid doing
redundant work.
You should also use the visible thoughts within the <think> tags to
decide how to best collaborate without doing the same work twice.
You should periodically check what other assistants are doing and
adjust your actions accordingly in order to collaborate as efficiently
as possible.

Finally, after the thinking process is concluded with the </think> tag,
you are expected to give the final answer very quickly, i.e. within few
tokens.
Hence, do NOT close the thinking process with the tag </think> before
you have completed the collaboration and reached an agreement on the
final answer with the other assistants.

It is extremely important you avoid doing the same work twice to make
best use of the limited thinking budget available, so communicate well
and efficiently!

From now on, you are {assistant_name}.
Now solve the next problem together. Keep track of who does what work
and communicate to avoid doing the same work twice.
```

User message

```
{QUERY}

Here are the current thoughts of the other assistant

### {agent_name}'s current thoughts ###

{other-assistant-thoughts}

### End of {agent_name}'s current thoughts ###


From now on, you are {agent_name}.

Don't forget to look at what the other assistants are doing. Coordinate
with them and avoid doing the same work twice to solve the problem
efficiently! If you have reached a conclusion and you know the answer,
interrupt the thinking process with </think>. Only then, provide the
final answer within the <answer> and </answer> tags.
```

B.1 SFT EXAMPLES

The following text boxes show a single SFT sample with two lanes, one for Alice and one for Bob. Note that we have removed or replaced several unicode characters of the chat template for Latex compatibility.

Alice's example

```
<|beginofsentence|><|User|>
You are one of 2 helpful assistants (namely Alice and Bob)
collaborating to solve a problem together, while writing their thoughts
in parallel.

Assistants collaborate without redundant work.
```

```
Each assistant's past and current thoughts are visible to all other
assistants while they are being written.
In other words, as each assistant writes its thoughts, it can see all
the thoughts which are simultaneously being written by all other
assistants.

You will see the other assistants unfinished steps. They will write
those steps concurrently with you. You will take into account what they
are doing. If another assistant gives you suggestions, you should
address them.
Other assistants will continue writing their thoughts in the background
while you will continue writing your own.
Since you and others both write your thoughts in parallel, you will
initially see only partial (unfinished) thoughts that others will
continue in parallel, while you write yours.
Other assistants may write new thoughts while you are writing yours.
You will use these partial thoughts to decide how best to collaborate
without doing the same work twice. You will periodically check what
other assistants are doing and you should adjust your actions based on
what they are doing so you collaborate efficiently with them.
If what you are currently doing is the same thing that another
assistant has already done or is in process of doing, you will stop
(e.g. Alice may say Wait, I was doing the same as Bob ...) and change
to a different task right away, so as to avoid doing redundant work.

Now solve the next problem together. Keep track of who does what work
and communicate well to avoid doing the same work twice.

From now on, you act as Alice.

A cell phone plan costs $20$ dollars each month, plus $5$ cents per
text message sent, plus $10$ cents for each minute used over $30$
hours. In January Michelle sent $100$ text messages and talked for
$30.5$ hours. How much did she have to pay?
Let's think step by step and output the final answer within \boxed{}.
<|Assistant|>
<think>
Okay, let's tackle this problem step by step.
First, the base cost is $20 per month.
Now, Michelle sent 100 text messages.
Each text is 5 cents, so 100 * $0.05.
Wait, Bob already mentioned the text messages.
Let me check the time part.
She talked for 30.5 hours.
The plan covers 30 hours, so the overage is 0.5 hours.
0.5 hours is 30 minutes.
Each minute over is 10 cents, so 30 * $0.10.
Bob is handling the time part, so I shouldn't repeat that.
Wait, the total would be base + text cost + overage.
Base: $20.
Texts: $5 (100*0.05).
Overage: $3 (30*0.10).
Total: 20+5+3=28.
Bob said 28, so that's consistent.
I need to make sure there's no miscalculation here.
Wait, 0.5 hours is 30 minutes, yes.
30 minutes * $0.10 = $3.
Texts: 100 * 0.05 = 5.
All adds up to $28.
No errors here.
So the answer is $28.
Since Bob already confirmed this, I agree with his calculation.
No need to redo anything.
Confirming once more to be safe.
```

```
Yes, all parts accounted for.
Final answer should be 28.
</think>
\boxed{28}<|endofsentence|>
```

Bob's example

```
<|beginofsentence|><|User|>
You are one of 2 helpful assistants (namely Alice and Bob)
collaborating to solve a problem together, while writing their thoughts
in parallel.

Assistants collaborate without redundant work.
Each assistant's past and current thoughts are visible to all other
assistants while they are being written.
In other words, as each assistant writes its thoughts, it can see all
the thoughts which are simultaneously being written by all other
assistants.

You will see the other assistants unfinished steps. They will write
those steps concurrently with you. You will take into account what they
are doing. If another assistant gives you suggestions, you should
address them.
Other assistants will continue writing their thoughts in the background
while you will continue writing your own.
Since you and others both write your thoughts in parallel, you will
initially see only partial (unfinished) thoughts that others will
continue in parallel, while you write yours.
Other assistants may write new thoughts while you are writing yours.
You will use these partial thoughts to decide how best to collaborate
without doing the same work twice. You will periodically check what
other assistants are doing and you should adjust your actions based on
what they are doing so you collaborate efficiently with them.
If what you are currently doing is the same thing that another
assistant has already done or is in process of doing, you will stop
(e.g. Alice may say Wait, I was doing the same as Bob ...) and change
to a different task right away, so as to avoid doing redundant work.

Now solve the next problem together. Keep track of who does what work
and communicate well to avoid doing the same work twice.

From now on, you act as Bob.

A cell phone plan costs $20$ dollars each month, plus $5$ cents per
text message sent, plus $10$ cents for each minute used over $30$
hours. In January Michelle sent $100$ text messages and talked for
$30.5$ hours. How much did she have to pay?
Let's think step by step and output the final answer within \boxed{}.
<|Assistant|>
<think>
Okay, let's tackle this problem step by step.
First, the base cost is $20 per month.
Now, Michelle sent 100 text messages.
The cost per text is 5 cents, so that's 100 * $0.05.
Wait, Alice already calculated the text messages part.
Let me check the time overage.
She talked for 30.5 hours, so over the 30-hour limit by 0.5 hours.
Convert 0.5 hours to minutes: 0.5 * 60 = 30 minutes.
Each minute over is 10 cents, so 30 * $0.10.
Alice mentioned the time part, so I don't need to repeat that.
Let me just confirm the calculations.
Texts: 100 * 0.05 = $5.
Time overage: 30 * 0.10 = $3.
Total cost: 20 + 5 + 3 = $28.
```

```
Wait, Alice already did the text cost.
She said texts are $5 and overage $3, so total $28.
I should check if my numbers match.
Yes, 100 text messages at 5 cents is $5.
0.5 hours over is 30 minutes, 30*0.10 is $3.
All adds up.
No mistakes here.
So the final answer should be $28.
I should confirm no other parts are missed.
The problem didn't mention other fees, so yes, total is 20 +5+3=28.
Looks correct.
Since Alice already worked through it and I'm just verifying, I can
agree.
No need to re-explain, just confirm the answer.
All right, so the answer is 28.
</think>
\boxed{28}<|endofsentence|>
```