Generalized Parallel Scaling with Interdependent Generations

Harry Dong[1,2], **David Brandfonbrener**[1], **Eryk Helenowski**[1], **Yun He**[1], **Mrinal Kumar**[1], **Han Fang**[1], **Yuejie Chi**[1,3], **Karthik Abinav Sankararaman**[1]
[1]Meta, [2]Carnegie Mellon University, [3]Yale University
harryd@andrew.cmu.edu, karthikabinavs@meta.com

Abstract

Parallel LLM inference scaling involves sampling a set of $N > 1$ responses for a single input prompt. However, these N parallel responses tend to be generated independently from each other, partitioning compute resources and leaving potentially useful information in one generation untapped by others. This is in contrast to response length scaling where past computation is used in all future steps. For higher quality responses and response sets, we propose `Bridge` to generate *interdependent responses in parallel* by rethinking batched LLM hidden states as holistic tensors rather than independent slices. With only a small amount (2.8%-5.1%) of new parameters, `Bridge` improves the relative mean accuracy gains from reinforcement learning with verifiable rewards by up to 50% and boosts consistency of correct responses. Trained once, `Bridge` scales to any generation width, all with greater performance than independent generations, unlocking a more general mode of parallel scaling that effectively leverages information between sequences, compatible with any post-generation aggregation technique.

1 Introduction

Scaling inference-time compute has given large language models (LLMs) substantial leaps in performance on difficult tasks. Many scaling methods concentrate resources to generate a *single* high-quality response such as with chains-of-thought (CoTs) [Wei et al., 2022] and decompositions of a problem into parallel substeps [Rodionov et al., 2025, Yang et al., 2025b]. However, there are also instances where a high-quality *set* of responses for each input is needed, such as in the case of output synthesis, best-of-N selection, and synthetic data generation. Scaling this in a parallel manner is traditionally done by sampling independent generations. Consequently, each generation is ignorant of the other rollouts, despite answering the same prompt. Independent generations for the same prompt leave potentially useful information derived from other responses unutilized, limiting the performance ceiling. In contrast, sequentially scaling CoTs ensures each sampled token can play a role in the final output. Motivated by the potential

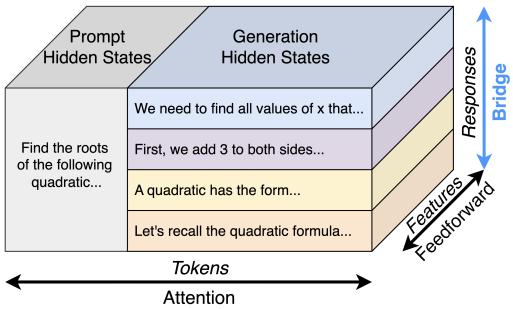

Figure 1: LLM hidden states are 3-D tensors, where attention and feedforward blocks explicitly transfer information between tokens and features, respectively. By instead treating parallel scaling generations as a single tensor rather than independent slices, our method, `Bridge`, operates along the batch axis, so that tokens from all sequences that share the same prompt can share information throughout generation.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

of shared information across parallel generations stemming from the same prompt, we aim to leverage these interactions to enhance and generalize parallel inference scaling.

There has been progress in integrating some form of parallel dependence for inference. A line of work explores breaking down reasoning steps into parallel paths with great success [Rodionov et al., 2025, Hsu et al., 2025, Pan et al., 2025, Jin et al., 2025, Yang et al., 2025b]. In these cases, parallel computation is funneled into a single output, useful for generating one high-quality response but not a high-quality *set* of responses. Even so, they highlight the potential of mid-generation interactions between sequences. We seek to extend parallel scaling with *interdependence*, which allows all N output sequences for one prompt to use all the compute and information available, not just a single isolated partition. *Thus, the challenge is finding a totally parallel method that uses N simultaneous threads to generate N responses with interdependence without extensive post-training*.

Looking at the operations on LLM hidden states reveals a clue to overcome these challenges (Figure 1). For batch size B, sequence length S, and hidden dimension D, the hidden states per forward pass has 3-D shape $B \times S \times D$. Attention and feedforward blocks blend information throughout each $S \times D$ slice with the batch dimension kept independent. Even minor inter-sample interactions like Batch Normalizations [Ioffe and Szegedy, 2015] were substituted with Layer Normalizations [Ba et al., 2016]. While this is natural for highly heterogeneous batches where samples with wildly different inputs can be fed together in the same forward pass without interference, *parallel scaling, which draws many responses from a single input, exhibits uniquely homogeneous structure since each output stems from the same input.* Hence, there is the potential for useful information transfer during the generation process which we exploit.

We introduce `Bridge` (**B**atch **r**easoning with **i**nter**d**ependent **ge**nerations), a method that shares information across tokens that stem from the same prompt in a batch for parallel scaling with interdependent generations. With a minor architectural change to LLMs, each token generated in a batch can depend on tokens in other generation threads with the same prompt. In turn, our method improves reasoning performance evaluated both at the individual response level (accuracy) and response set level (G-Pass@k_τ [Liu et al., 2025]). Furthermore, our method focuses on generation, so any post-generation aggregation technique can be used. We push the following advancements towards parallel scaling:

1. **Parallelism with Dependence:** Instead of generating in isolated silos, `Bridge` allows information to flow between sequences while maintaining complete generation parallelism. Thus, inference compute is pooled together for all tokens, rather than being partitioned. We show `Bridge` significantly increases the final performance after reinforcement learning with verifiable rewards (RLVR) on 7 math benchmarks using multiple reasoning models.

2. **Low Cost:** By adding only 2.8% to 5.1% additional parameters, and warming up on a small supervised fine tuning (SFT) dataset (e.g. GSM8K [Cobbe et al., 2021]), `Bridge` already significantly improves the effectiveness of RLVR.

3. **Versatility:** `Bridge` has no restriction on the width of parallelism and is robust to train-time and test-time width discrepancies. Trained once, all tested widths outperform independent generations in terms of accuracy, coverage, and consistency. Furthermore, `Bridge` does not rely on any heuristics or interventions at any point in the generation process.

Our extensive experiments on multiple models and tasks show that `Bridge` effectively shares information across multiple generations for the same input. For example, our method improves the relative benefit of RLVR on DeepSeek-R1-Distill-Qwen-7B by 50% averaged over 7 tasks, compared to the next best method. With the same model, `Bridge` also increases the rate at which all responses to a single competition math problem are correct from 15.3% to 18.1%.

Paper Organization. In the next section (Section 2), we cover relevant background on test-time scaling with an emphasis on parallel scaling. Then, we introduce `Bridge` in Section 3, detailing the algorithm, the training pipeline, and its implications. We demonstrate `Bridge`'s efficacy on a variety of math reasoning datasets, evaluated both on sample-wise accuracy and on global response set quality in Section 4. We go further and provide a thorough investigation of our method including varying the generation width, sequence length extrapolation, and learned features in Section 4.3.

2 Background & Related Works

Test-time Scaling. In part due to the success of scaling LLM training [Kaplan et al., 2020, Hoffmann et al., 2022], there has been a growing interest in quantifying how far scaling LLM inference-time compute can push performance, especially on difficult reasoning tasks. The main axes of inference scaling are generation length and the number of generations. To scale generation length, LLMs are encouraged to produce long CoTs before arriving at a final answer [Guo et al., 2025, Yang et al., 2025a, Muennighoff et al., 2025] which is usually of higher quality than shorter CoTs. In the case of extreme generation lengths, there appears to be diminishing or even negative returns, suggesting a limitation of current models to scale along this axis [Gema et al., 2025]. To scale along the number of generations axis, LLMs can output multiple responses for a single query, increasing the probability of a high quality response being generated [Brown et al., 2024, Snell et al., 2024, Wu et al., 2024, Manvi et al., 2024, Sun et al., 2024, Dong et al., 2025]. However, independent generations divide computational resources among themselves, oblivious to each other's progress, which leads to less significant performance gain with additional compute than length scaling [Mirtaheri et al., 2025]. To promote parallel exploration during training, training with a Pass@k objective has shown promise which could be an interesting extension to our method [Chen et al., 2025b].

Post-generation Synthesis. Instead of selecting one response from a pool of candidate responses, some works investigate ways to synthesize multiple responses together. One way is to take an unweighted or weighted majority vote across responses [Wang et al., 2022, Uesato et al., 2022, Lightman et al., 2023, Li et al., 2023], but this is geared mainly for discrete answers, and an effective synthesis of reasoning traces remains unclear. There are also approaches where multiple responses are concatenated and fed into an LLM to extract or combine information [Chen et al., 2023, Qi et al., 2025, Zhao et al., 2025]. *Our work's focus is on the generation phase, so many of these post-generation aggregation techniques can be seamlessly integrated.*

Mid-generation Synthesis/Pruning. There has also been some work in developing techniques to share information across outputs mid-generation. For instance, Hogwild! Inference [Rodionov et al., 2025] and Group Think [Hsu et al., 2025] share key-value caches across generation runs to collaborate and decompose tasks into subtasks. Similarly, some methods concatenate outputs of parallel processes to aid in the main decoding thread [Pan et al., 2025, Jin et al., 2025, Yang et al., 2025b, Macfarlane et al., 2025]. Training from scratch, ParScale [Chen et al., 2025a] fans outs an input into multiple paths within the model architecture then aggregates them to predict the next token. Whereas these previous methods funnel resources of N parallel processes to produce one output, our method uses N parallel processes to simultaneously generate N high quality outputs. This way, our design integrates inter-output dependency mid-generation while producing different responses simultaneously, flexibly suitable for post-generation synthesis, RLVR training, and synthetic data generation. Another line of work involves stopping unpromising outputs mid-generation to devote more resources to other parallel generations [Fu et al., 2025, Sun et al., 2024]. These works show excellent reductions in compute, and composing them with our method is of interest for future work.

3 `Bridge`: Connecting Generation Paths

Sharing information between samples mid-generation in the latent space gives rise to a couple technical challenges. One is finding an effective and efficient way to achieve this. Attention and feedforward blocks already pose serious static and dynamic memory bottlenecks, which we want to avoid accentuating while still improving accuracy. Second, we also need versatility to allow for any number of parallel generations at test-time. `Bridge` overcomes these challenges with small attention-like blocks that fit into any LLM. We begin with a description of `Bridge`, its connections, and its implications in Section 3.1, followed by SFT (Section 3.2) and RLVR (Section 3.3) details.

3.1 `Bridge` Architecture

We introduce `Bridge`, a new transformer [Vaswani et al., 2017] block that introduces dependence between samples in a batch. At a high level, `Bridge` performs attention between tokens, which share the same prompt and do not come from completed generations, in a batch at each timestep. We summarize our method in Figure 2 and Algorithm 1.

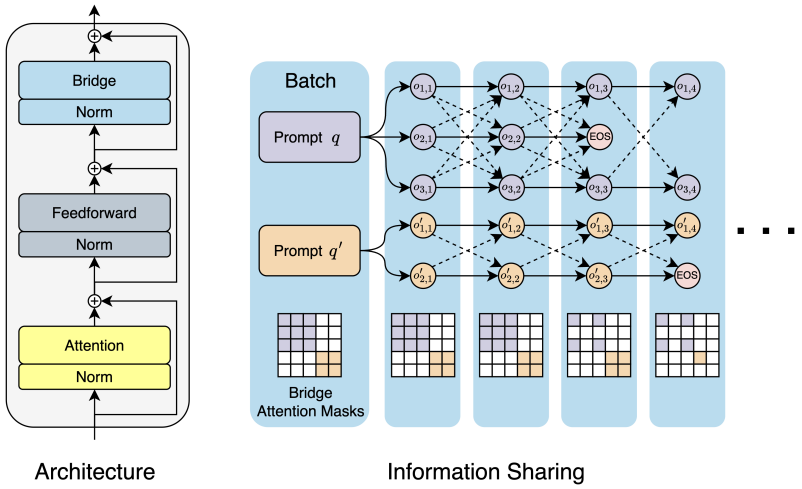

Figure 2: Our method design. **(Left)** A `Bridge` block and input normalization layer are added after each feedforward block. **(Right)** A timestep's tokens stemming from the same input prompt attend to each other in `Bridge` blocks, denoted by the arrows. Dotted arrows illustrate all the locations of information transfer to different sequences in a Markovian fashion (token features only at the current timestep are shared to predict the next timestep's tokens). Attention is masked for tokens from different prompts and from completed generations. White squares are masked cells.

We first describe self-attention layers. Define hidden states $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{B \times S \times D}$ for batch size B, sequence length S, and hidden dimension D. Let $[\boldsymbol{\mathcal{X}}]_{b,\cdot,\cdot}$ and $[\boldsymbol{\mathcal{X}}]_{\cdot,s,\cdot}$ be the b-th and s-th 2-D slices along the batch and sequence axes, respectively. Self-attention, parameterized by $\boldsymbol{W}_{\text{Attn,Q}}, \boldsymbol{W}_{\text{Attn,K}} \in \mathbb{R}^{D \times D_{\text{QK}}}$ and $\boldsymbol{W}_{\text{Attn,V}}, \boldsymbol{W}_{\text{Attn,O}}^{\top} \in \mathbb{R}^{D \times D_{\text{VO}}}$, is calculated independently for each sample b:

$$\boldsymbol{Q}_{\text{Attn},b} = [\boldsymbol{\mathcal{X}}]_{b,\cdot,\cdot}\boldsymbol{W}_{\text{Attn,Q}}, \qquad \boldsymbol{K}_{\text{Attn},b} = [\boldsymbol{\mathcal{X}}]_{b,\cdot,\cdot}\boldsymbol{W}_{\text{Attn,K}}, \qquad \boldsymbol{V}_{\text{Attn},b} = [\boldsymbol{\mathcal{X}}]_{b,\cdot,\cdot}\boldsymbol{W}_{\text{Attn,V}},$$

$$[\text{Attn}(\boldsymbol{\mathcal{X}})]_{b,\cdot,\cdot} = \underbrace{\text{Softmax}(\text{Mask}_{\text{Attn}}(\boldsymbol{Q}_{\text{Attn},b}\boldsymbol{K}_{\text{Attn},b}^{\top}))}_{\in \mathbb{R}^{S \times S}}\boldsymbol{V}_{\text{Attn},b}\boldsymbol{W}_{\text{Attn,O}}. \tag{1}$$

`Bridge` blocks are similar, but attention between samples is calculated independently for each token index s. Letting $\boldsymbol{W}_{\texttt{Bridge,Q}}, \boldsymbol{W}_{\texttt{Bridge,K}} \in \mathbb{R}^{D \times D_{\text{QK}}}$ and $\boldsymbol{W}_{\texttt{Bridge,V}}, \boldsymbol{W}_{\texttt{Bridge,O}}^{\top} \in \mathbb{R}^{D \times D_{\text{VO}}}$,

$$\boldsymbol{Q}_{\texttt{Bridge},s} = [\boldsymbol{\mathcal{X}}]_{\cdot,s,\cdot}\boldsymbol{W}_{\texttt{Bridge,Q}}, \ \boldsymbol{K}_{\texttt{Bridge},s} = [\boldsymbol{\mathcal{X}}]_{\cdot,s,\cdot}\boldsymbol{W}_{\texttt{Bridge,K}}, \ \boldsymbol{V}_{\texttt{Bridge},s} = [\boldsymbol{\mathcal{X}}]_{\cdot,s,\cdot}\boldsymbol{W}_{\texttt{Bridge,V}},$$

$$[\texttt{Bridge}(\boldsymbol{\mathcal{X}})]_{\cdot,s,\cdot} = \underbrace{\text{Softmax}(\text{Mask}_{\texttt{Bridge}}(\boldsymbol{Q}_{\texttt{Bridge},s}\boldsymbol{K}_{\texttt{Bridge},s}^{\top}))}_{\in \mathbb{R}^{B \times B}}\boldsymbol{V}_{\texttt{Bridge},s}\boldsymbol{W}_{\texttt{Bridge,O}}. \tag{2}$$

There are 3 key differences between usual self-attention and `Bridge` beyond a transposition of $\boldsymbol{\mathcal{X}}$:

- Instead of a decoder mask, `Bridge` applies an attention mask that omits attention to tokens from sequences stemming from different prompts and sequences that have completed generation. See Figure 2 for an example.
- No positional encoding is used to preserve sample position invariance.
- Without attention to previous tokens, `Bridge`'s Markovian design *does not maintain a key-value cache*.

We place a `Bridge` block after each feedforward block with a residual stream and input normalization layer that mimics existing blocks, shown in Figure 2. `Bridge` is active during the prefill stage too, but since all hidden states for the same input are identical, `Bridge` blocks act as linear layers.

Connection to Efficient Attention for Tensors. `Bridge` unlocks the ability for an LLM to treat a batch of LLM hidden states as a 3-D ($B \times S \times D$) structure rather than a stack of independent 2-D slices. In this way, the inputs are analogous to images, and the decoding process is like autoregressively generating additional columns. With this interpretation, `Bridge` applying attention

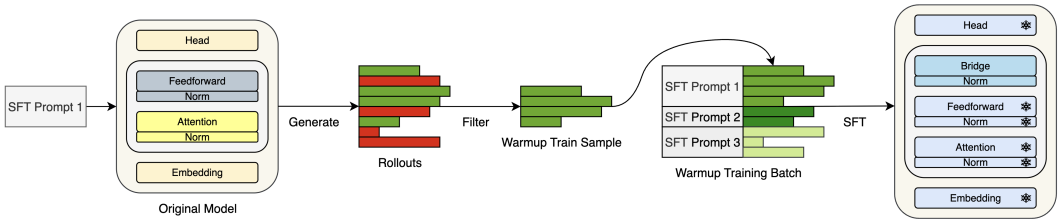

Figure 3: Warm up procedure. The original LLM generates candidate traces which are filtered by correctness and compiled into a dataset. SFT on this generated dataset only updates new parameters. The P-Match baseline substitutes `Bridge` blocks with MLPs matched in parameter count.

operations on different axes of an input is similar to axial attention [Ho et al., 2019] which was introduced first in computer vision to accelerate encoder attention but has since seen wide success in various applications such as in medicine [Azad et al., 2024], materials science [Dong et al., 2023], and algorithm discovery [Fawzi et al., 2022].

Generation Interdependence. For B independent rollouts we sample the next token $o_{b,s+1}$ from

$$p(o_{b,s+1}|q, o_{b,1:s})$$

for sample b, timestep s, input prompt q, and previously generated tokens $o_{b,1:s}$. With `Bridge`, the next token distribution becomes

$$p(o_{b,s+1}|q, \{o_{b',1:s}\}_{b'=1}^{B})$$

for each sample b. Conditioned on past tokens, `Bridge` preserves independence between tokens at the same timestep, which allows next token sampling to still be performed in parallel:

$$(o_{b_1,s+1} \perp\!\!\!\perp o_{b_2,s+1})|\{o_{b',1:s}\}_{b'=1}^{B} \text{ for } b_1 \neq b_2.$$

3.2 SFT Warm up

While RLVR can be immediately applied with `Bridge` since these new blocks are initialized to have no contribution, we can also optionally warm them up with SFT for more sufficient training and better downstream performance. A desirable SFT dataset would include many reasoning traces to one prompt. To stay close to the original LLM's generation distribution, we create SFT datasets by first responding to prompts from an existing math dataset. Then, traces are filtered for correctness. During training, these correct traces are fed together in the same batch to warm up `Bridge` blocks with SFT. All other parameters are frozen. Figure 3 illustrates the warm up procedure, and Table 3 explores more in-depth on the benefits of warm up.

3.3 RLVR Objective

We train LLMs with `Bridge` using GRPO [Shao et al., 2024]. We use a variant specified by Yu et al. [2025] which performs token-level normalization to reduce length bias. Letting the group size be G, the advantage of the i-th output o_i to input q with reward r_i is $\hat{A}_i = \frac{r_i - \text{mean}(r_1,\ldots,r_G)}{\text{std}(r_1,\ldots,r_G)}$. Then, for clipping threshold ϵ, hyperparameter β, and policy π_θ parameterized by θ, the objective is

$$\mathcal{J}(\theta) = \frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{s=1}^{|o_i|}\left\{\min\left[R_{i,s}(\theta)\hat{A}_i, \text{clip}(R_{i,s}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\right] - \beta D_{\text{KL}}(\pi_\theta||\pi_{\theta_{\text{ref}}})\right\}, \tag{3}$$

where

$$R_{i,s}(\theta) = \frac{\pi_\theta(o_{i,s}|q, \{o_{j,1:s-1}\}_{j=1}^{G})}{\pi_{\theta_{\text{old}}}(o_{i,s}|q, \{o_{j,1:s-1}\}_{j=1}^{G})},$$

$$D_{\text{KL}}(\pi_\theta||\pi_{\theta_{\text{ref}}}) = \frac{\pi_{\theta_{\text{ref}}}(o_{i,s}|q, \{o_{j,1:s-1}\}_{j=1}^{G})}{\pi_\theta(o_{i,s}|q, \{o_{j,1:s-1}\}_{j=1}^{G})} - \log\frac{\pi_{\theta_{\text{ref}}}(o_{i,s}|q, \{o_{j,1:s-1}\}_{j=1}^{G})}{\pi_\theta(o_{i,s}|q, \{o_{j,1:s-1}\}_{j=1}^{G})} - 1.$$

The key differences from the GRPO objective (and its variants) and our objective are not formulaic but rather inherently induced from the architecture of `Bridge`. Namely, the ratio and KL divergence terms

now contain inter-sample dependence between relevant samples, breaking the original assumption of independent trajectories. By linking the advantages and logits in a group, the loss and gradients per output are intertwined with other outputs' that share the same prompt. In other words, gradients from all sequences, containing both positive and negative advantages, are backpropagated through each sequence because of `Bridge` blocks. Further considerations are discussed in Appendix C.

4 Experiments

We now showcase the benefit of `Bridge` across multiple models and math reasoning benchmarks. After describing our setup in Section 4.1, we first show that applying RLVR with `Bridge` blocks improves accuracy more than other methods. For instance, DeepSeek-R1-Distill-Qwen-7B with `Bridge` blocks observes a relative 50% further improvement with RLVR than the next best method (Section 4.2.1). Then, in Section 4.2.2, we demonstrate that `Bridge` also improves the output set quality across several metrics in terms of coverage and correctness consistency. Finally, in Section 4.3, we highlight some important characteristics of our method including the versatility of generation width, length extrapolation, benefit of warm up, and feature contributions.

4.1 Experimental Settings

Models and Baselines. We test `Bridge` on DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B, and DeepSeek-R1-Distill-Llama-8B, which we abbreviate to DS-Qwen-1.5B, DS-Qwen-7B, and DS-Llama-8B, respectively [Dubey et al., 2024, Yang et al., 2024, Guo et al., 2025]. We use 4 query and key-value attention heads for `Bridge`, each with the same dimension as the original model's head dimension. This only adds 5.1%, 2.8%, and 3.4% extra parameters on top of the original DS-Qwen-1.5B, DS-Qwen-7B, and DS-Llama-8B models, respectively. Table B in Appendix B lists the exact parameter counts. Our parameter-matched baseline which we call "P-Match" adds 2-layer MLPs of the same size in the same positions as `Bridge` blocks which serves to show the limited effect of just adding parameters. Matched in parameter count, P-Match and `Bridge` are also trained with the same warm up and RLVR pipeline. Both methods are initialized to have zero contribution.

Training. For the SFT warm up stage, we first use the original LLM to generate 8 response for each GSM8K [Cobbe et al., 2021] problem and then filter out incorrect responses and problems with one or fewer correct responses. We train only the additional parameters with `Bridge` and P-Match on this custom dataset for 5 epochs and keeping the best checkpoint according to the perplexity on 500 validation problems (and their corresponding set of correct reasoning traces). This checkpoint is inserted in the model for RLVR where we train the full model on DeepScaleR-Preview-Dataset [Luo et al., 2025] for 1000 gradient steps. DS-Qwen-1.5B is trained with generation width 8 while the others were trained with 4. The only reward is correctness of the generation. Our training hyperparameters are listed in Appendix A.

Evaluation. We evaluate `Bridge` on 7 math benchmarks: MATH-500 [Hendrycks et al., 2021, Lightman et al., 2023], AIME24, AIME25 [AIME, 2025], AMC23 [AMC, 2023], BRUMO25 [BRUMO, 2025], CMIMC25 [CMIMC, 2025], and HMMT_FEB25 [HMMT, 2025]. Evaluating MATH-500 on every 100 training steps, the checkpoint with the highest validation accuracy is used to test on the remaining 6 benchmarks. We evaluate across 4 responses per MATH-500 sample and 16 responses per sample from the other math benchmarks. Sampling temperature and top-p are set to 0.6 and 0.95, respectively. We set the generation width of `Bridge` to 8 for all tasks except MATH-500, which we set to 4 since we only evaluate on 4 responses per sample. We adapt our evaluations from the Lighteval framework [Habib et al., 2023].

4.2 Reasoning Performance

Here, we show the performance improvements of our method `Bridge` which leverages inter-sample information sharing for high quality generations. We evaluate performance both on per-output accuracy (Section 4.2.1) and macroscopically, on the set of outputs generated per prompt (Section 4.2.2).

Table 1: Accuracy comparison across math benchmarks. In each section, the 4 rows from top to bottom are the performance of the original model, RLVR applied on the original model, P-Match (extra MLPs) with SFT warm up and RLVR, and `Bridge` with SFT warm up and RLVR. The 2 rightmost columns show the average across all benchmarks and the average improvement over the original model. MATH-500, AMC23, BRUMO25, CMIMC25, and HMMT_ FEB25 are abbreviated to MATH, AMC, BRU, CMI, and HMMT, respectively.

Model	MATH	AIME24/25	AMC	BRU	CMI	HMMT	Avg	$\uparrow \Delta$
DS-Qwen-1.5B	73.65	14.58 / 13.75	50.47	18.12	4.53	8.33	26.20	0.00
RLVR only	78.75	16.04 / 17.71	60.78	18.54	3.75	7.50	29.01	2.81
P-Match	78.65	18.33 / 18.33	**61.25**	20.83	5.47	7.71	30.08	3.88
`Bridge`	**81.30**	**20.00 / 19.38**	**61.25**	**21.67**	**5.63**	**9.38**	**31.23**	**5.03**
DS-Qwen-7B	82.15	23.96 / 22.50	67.35	23.96	5.94	12.92	34.11	0.00
RLVR only	**88.15**	30.00 / 23.13	74.07	28.33	7.81	**13.96**	37.92	3.81
P-Match	86.80	29.58 / 24.79	70.00	27.08	6.25	11.25	36.54	2.43
`Bridge`	**88.15**	**33.75 / 25.63**	**77.97**	**30.21**	**10.00**	13.13	**39.83**	**5.72**
DS-Llama-8B	73.40	15.21 / 13.54	57.50	16.25	2.66	8.33	26.70	0.00
RLVR only	76.70	17.92 / 20.00	62.34	16.46	**6.09**	10.21	29.96	3.26
P-Match	78.00	22.08 / **20.83**	62.34	18.54	4.84	12.08	31.24	4.54
`Bridge`	**80.15**	**25.76** / 18.44	**65.44**	**19.06**	5.01	**12.19**	**32.29**	**5.59**

4.2.1 Accuracy

Beginning with standard accuracy (Pass@1), we compare the performance of the original model, original model with RLVR, P-Match with SFT and RLVR, and `Bridge` with SFT and RLVR on several math benchmarks. Results in Table 1 show that in nearly all cases and on average, `Bridge` obtains the highest accuracy compared to all other methods. *In particular, the average performance improvements of our method on the original model is 30%, 50%, and 23% relatively more than that of the next best method on DS-Qwen-1.5B, DS-Qwen-7B, and DS-Llama-8B models, respectively.* P-Match with parameter counts pegged to `Bridge` improves accuracy from just pure RLVR most of the time but is much more inconsistent, such as in the case of DS-Qwen-7B. This indicates that the superior performance of `Bridge` is not solely attributed to additional parameters. Furthermore, even though DS-Qwen-7B and DS-Llama-8B were trained with generation width 4, the evaluation results with width 8 are still stronger than the other independent sampling methods, showing the robustness of `Bridge`. In addition, the improvement by `Bridge` is greater for larger models, and scaling up to even larger ones remains of interest for future work.

4.2.2 Set Evaluations

Zooming out, we show `Bridge` also improves the consistency and coverage across multiple generation attempts. To evaluate the set of responses to a single input, we use the G-Pass@k_τ metric [Liu et al., 2025], which paints a more holistic picture of model potential (coverage) and consistency. Whereas Pass@k is the probability of a correct output in k responses, G-Pass@k_τ is the probability of $0 < \tau \leq 1$ fraction of k responses being correct. More formally, for n responses and c correct responses,

$$\text{Pass@}k = \mathbb{E}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right], \qquad \text{G-Pass@}k_\tau = \mathbb{E}\left[\sum_{j=\lceil \tau k \rceil}^{c} \frac{\binom{c}{j} \cdot \binom{n-c}{k-j}}{\binom{n}{k}}\right].$$

As $\tau \to 0$, G-Pass@k_τ is simply the coverage. On the other extreme, G-Pass@k_1 is the probability that all k responses are correct.

From Figure 4, `Bridge` achieves higher G-Pass@8_τ values for nearly all values of τ and models. This demonstrates that `Bridge` can achieve greater coverage without spreading out its responses to many incorrect answers. *In other words, not only do `Bridge` blocks increase the probability of a correct response in the response set more than the other methods, they also increase the frequency at which they occur.* Again, we note that Qwen-7B and DS-Llama-8B were trained with generation width 4 yet they generalize well to evaluation width 8.

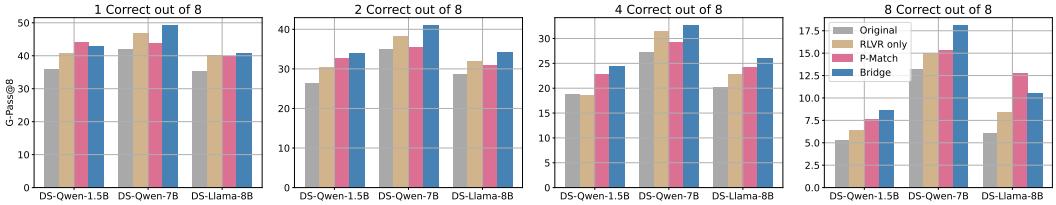

Figure 4: G-Pass@8_τ averaged across AIME24, AIME25, AMC23, BRUMO25, CMIMC25, and HMMT_FEB25. Each chart measures the minimum number of correct answers ($\tau \cdot k$) out of $k = 8$ simultaneous responses. `Bridge` has the greatest coverage ($\tau \cdot k = 1$) and answers correctly most consistently ($\tau \cdot k > 1$) in the vast majority of cases. Higher is better.

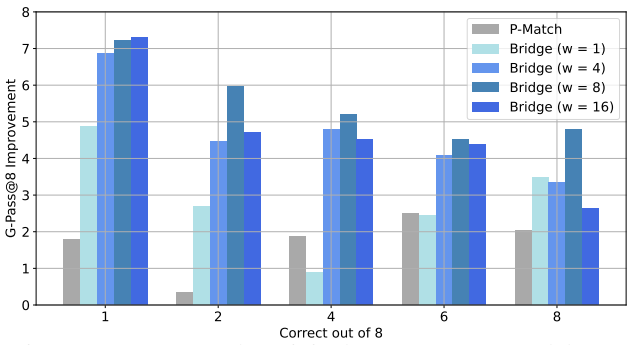

Figure 5: G-Pass@8_τ improvement upon the original DS-Qwen-7B model averaged across AIME24, AIME25, AMC23, BRUMO25, CMIMC25, and HMMT_FEB25 with relation to the evaluation generation width w of `Bridge`. The x-axis ($\tau \cdot k$) indicates the number of responses out of $k = 8$ that must be correct.

4.3 Ablations and Analysis

Generation Width. The design of `Bridge` allows complete flexibility in the number of parallel generations, or generation width w, due to the removal of positional encoding. Here, we show its generalizability to other widths on DS-Qwen-7B which was trained on a width of 4 with RLVR. In Table 2, in all cases where $w > 1$, `Bridge` outperforms P-Match in terms of task-wise and global average accuracy. We also investigate the effect of w on set quality in Figure 5. Again, we generally see a vast improvement upon the original model and P-Match with $w > 1$ for all G-Pass@8_τ settings. *These results show not only the benefit of sharing information via `Bridge` but also the generalizability to widths wider and thinner than its training width.* At the extreme of $w = 1$, equivalent to independent generations, results in average accuracy that falls between RLVR only and P-Match, indicating that `Bridge` blocks do not harm independent reasoning.

Table 2: Accuracy across 16 samples of varying `Bridge` generation widths, w, with DS-Qwen-7B which was trained at width 4 with RLVR. A `Bridge` width of 1 is equivalent to independent generation. Tasks are abbreviated as described in Table 1.

Method	AIME24	AIME25	AMC	BRU	CMI	HMMT	Avg	$\uparrow \Delta$
DS-Qwen-7B	23.96	22.50	67.35	23.96	5.94	12.92	26.11	0.00
RLVR only	30.00	23.13	74.07	28.33	7.81	**13.96**	29.55	3.44
P-Match	29.58	24.79	70.00	27.08	6.25	11.25	28.16	2.05
`Bridge` ($w = 1$)	28.54	25.21	73.75	27.29	8.44	11.04	29.05	2.94
`Bridge` ($w = 4$)	30.21	**25.63**	77.19	29.58	9.38	13.13	30.85	4.74
`Bridge` ($w = 8$)	**33.75**	**25.63**	**77.97**	30.21	**10.00**	13.13	**31.78**	**5.67**
`Bridge` ($w = 16$)	33.13	25.00	76.09	**30.63**	7.97	**13.75**	31.09	4.98

Generation Length. `Bridge` also shows strong generalizability along the length axis. We demonstrate its performance as we extrapolate beyond its training length of 4096, again measuring both individual and set performance. From Figure 6, our method scales smoothly and better than the other

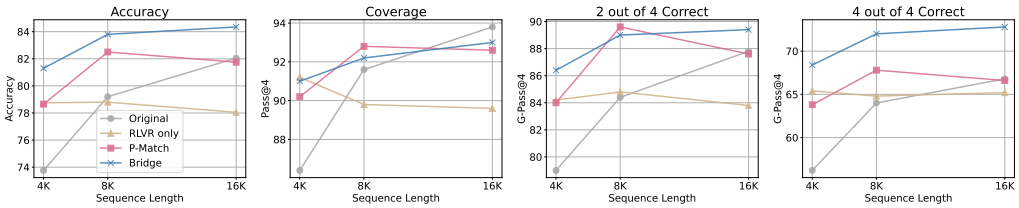

Figure 6: From left to right, DS-Qwen-1.5B MATH-500 accuracy, coverage, G-Pass@$4_{0.5}$, and G-Pass@4_1 as generation length increases. We generate 4 responses per input.

baselines in most cases. At the individual response level, our method achieves the highest accuracy across all generation lengths. At the set level, `Bridge` blocks increase the number of sets that *only* had correct answers by 6.0% compared to the next best at 16K generation length, illustrating our method's consistency to generate correct answers.

Cold Start vs. Warm up. Warming up `Bridge` blocks with SFT prior to RLVR outlined in Section 3.2 leads to improvements in performance, shown in Table 3. The slight improvement implies that although it is prefered to warm up these new layers, it is not catastrophic if RLVR is applied directly from initialization.

Table 3: Accuracy comparison between cold start RLVR and RLVR with SFT warmed up `Bridge` blocks in DS-Qwen-7B. Tasks are abbreviated as described in Table 1.

	MATH	AIME24/25	AMC	BRU	CMI	HMMT	Avg
Cold Start	**88.70**	33.13 / **26.67**	77.19	29.80	7.04	12.09	39.23
Warmed up	88.15	**33.75** / 25.63	**77.97**	**30.21**	**10.00**	**13.13**	**39.83**

Feature Contribution. Having shown the improved performance brought by `Bridge`, we now briefly peer into the effect that it has on LLM hidden states. We measure this by finding the ratio between the output norm of each block with the corresponding residual norm of each token, with lower values suggesting relatively little effect on the residual features (Figure 7). Surprisingly, we find `Bridge` blocks contribute little compared to its counterpart in P-Match, despite having a significant impact on the performance.

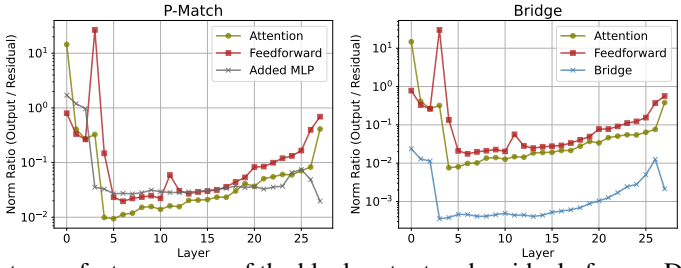

Figure 7: Ratio between feature norms of the block output and residual of every DS-Qwen-7B layer.

5 Conclusion

To generalize and enhance parallel inference scaling for LLMs, we introduce `Bridge`, a novel and inexpensive architectural addition to LLMs that allows parallel generations for the same input to share information with each other throughout the decoding process. We demonstrate that our method improves both single sample accuracy and set-wise quality across multiple models and several reasoning tasks. We achieve this by rethinking hidden states in parallel scaling as higher order tensors rather than disjoint slices. With this interpretation, this also plants the seeds for many exciting future directions such as observing the the effect of `Bridge` blocks during pretraining or mid-training, post-training with a Pass@k or another global objective, and quantifying the benefit on other modalities. Such directions will push parallel scaling as a much more effective axis of LLM inference scaling.

Acknowledgments and Disclosure of Funding

We thank Bradley Brown for frequent discussions about this project.

References

AIME. AIME problems and solutions, 2025. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

AMC. American mathematics competition, 2023. URL https://maa.org/student-programs/amc/.

Reza Azad, Ehsan Khodapanah Aghdam, Amelie Rauland, Yiwei Jia, Atlas Haddadi Avval, Afshin Bozorgpour, Sanaz Karimijafarbigloo, Joseph Paul Cohen, Ehsan Adeli, and Dorit Merhof. Medical image segmentation review: The success of u-net. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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.

BRUMO. Brown university math olympiad 2025, 2025. URL https://www.brumo.org/.

Mouxiang Chen, Binyuan Hui, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Jianling Sun, Junyang Lin, and Zhongxin Liu. Parallel scaling law for language models. *arXiv preprint arXiv:2505.10475*, 2025a.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025b.

CMIMC. Carnegie mellon informatics and mathematics competition, 2025. URL https://cmimc.math.cmu.edu/.

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

Harry Dong, Sean Donegan, Megna Shah, and Yuejie Chi. A lightweight transformer for faster and robust ebsd data collection. *Scientific Reports*, 13(1):21253, 2023.

Harry Dong, Bilge Acun, Beidi Chen, and Yuejie Chi. Scalable llm math reasoning acceleration with low-rank distillation. *arXiv preprint arXiv:2505.07861*, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025. URL https://arxiv.org/abs/2508.15260.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, et al. Inverse scaling in test-time compute. *arXiv preprint arXiv:2507.14417*, 2025.

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.

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`.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

HMMT. Hmmt, 2025. URL `https://www.hmmt.org/`.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

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

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. *arXiv preprint arXiv:2505.11107*, 2025.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

Tian Jin, Ellie Y Cheng, Zack Ankner, Nikunj Saunshi, Blake M Elias, Amir Yazdanbakhsh, Jonathan Ragan-Kelley, Suvinay Subramanian, and Michael Carbin. Learning to keep a promise: Scaling language model decoding parallelism with learned asynchronous decoding. *arXiv preprint arXiv:2502.11517*, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. Are your llms capable of stable reasoning?, 2025. URL `https://arxiv.org/abs/2412.13147`.

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, 2025. Notion Blog.

Matthew Macfarlane, Minseon Kim, Nebojsa Jojic, Weijia Xu, Lucas Caccia, Xingdi Yuan, Wanru Zhao, Zhengyan Shi, and Alessandro Sordoni. Instilling parallel reasoning into language models. In *2nd AI for Math Workshop @ ICML 2025*, 2025. URL `https://openreview.net/forum?id=a3o4b3hkwp`.

Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*, 2024.

Parsa Mirtaheri, Ezra Edelman, Samy Jelassi, Eran Malach, and Enric Boix-Adsera. Let me think! a long chain-of-thought can be worth exponentially many short ones. *arXiv preprint arXiv:2505.21825*, 2025.

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. *arXiv preprint arXiv:2501.19393*, 2025.

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. *arXiv preprint arXiv:2504.15466*, 2025.

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

Gleb Rodionov, Roman Garipov, Alina Shutova, George Yakushev, Vage Egiazarian, Anton Sinitsin, Denis Kuznedelev, and Dan Alistarh. Hogwild! inference: Parallel llm generation via concurrent attention. *arXiv preprint arXiv:2504.06261*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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.

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290*, 2024.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

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.

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, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. Multiverse: Your language models secretly decide how to parallelize and merge generation, 2025b. URL `https://arxiv.org/abs/2506.09991`.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. The majority is not always right: Rl training for solution aggregation, 2025. URL `https://arxiv.org/abs/2509.06870`.

A Training Hyperparameters

Table 4 lists the hyperparameters used for RLVR and SFT.

Table 4: Training hyperparameters.

Hyperparameter	RLVR	SFT & Warmup Data Generation
Mixed precision	BF16	BF16
Optimizer	AdamW	AdamW
KL penalty	1e-3	N/A
Learning rate	1e-6	2e-5
LR scheduler	constant	linear
LR warmup	10%	10%
Training steps	1000	varies
Batch size (rollouts $\times$ samples)	448	32
Rollouts per sample	8	[2,8]
Total samples	7000	varies
Max length	4096	2048
Steps per rollout	8	N/A
Temperature	0.6	0.6
Heads	4	4
Generation width	4 or 8	[2,4]

B Parameter Count Breakdown

`Bridge` has a low memory cost, adding relatively very few parameters (2.8% to 5.1%) to LLMs. Table 5 shows the exact parameter counts for each model.

Table 5: Distribution of parameters (B) across embedding/head, attention (Attn), feedforward (FF), and `Bridge` blocks.

Model	Embed & Head	Attn	FF	`Bridge`	Orig. Total	New Total
DS-Qwen-1.5B	0.47	0.15	1.16	0.09	1.78	1.86
DS-Qwen-7B	1.09	0.82	5.70	0.21	7.62	7.82
DS-Llama-8B	1.05	1.34	5.64	0.27	8.03	8.30

C Further GRPO Considerations

While our method inserts dependence between sequence and therefore their corresponding rewards, the sample permutation invariance of `Bridge` blocks means the unconditional rewards are still identically distributed. This implies

$$\mathbb{E}(\hat{A}_i) = \mathbb{E}\left(\frac{r_i - \frac{1}{n}\sum_{j=1}^{G} r_j}{\text{std}(r_1, \ldots, r_G)}\right) = \mathbb{E}\left(\frac{r_i - \frac{1}{n} \cdot n r_i}{\text{std}(r_1, \ldots, r_G)}\right) = 0,$$

preserving unbiasedness of the advantage. To preserve some notion of independence between rollouts for GRPO, one can generate multiple groups per prompt with `Bridge` and compute advantages between groups. Though this deserves exploration in future work, we do not do this here as it would be computationally expensive, and our single group setup is already empirically performative.

D `Bridge` Placement

Here, we examine in the architectural placement of `Bridge` blocks. In Table 6, we compare the resulting MATH500 accuracy after applying RLVR on DS-Qwen-1.5B with `Bridge` blocks added after attention blocks or after feedforward blocks, our chosen architecture for the experiments. Since

there is not a significant difference, this implies flexibility in placement, though we choose to stick with the one with the higher warmed up performance for our experiments.

Table 6: Effect on MATH-500 accuracy when inserting `Bridge` blocks after attention blocks vs. after feedforward blocks (chosen architecture) in DS-Qwen-1.5B.

Placement	Cold Start	Warmed up
After Attention	80.20	80.20
After Feedforward	80.15	81.30

E `Bridge` Pseudocode

Algorithm 1 sketches the pseudocode for `Bridge` blocks, following (2). Like normal self-attention, this can easily be extended to multiple heads.

Algorithm 1 `Bridge` Block

Input: $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{B \times S \times D}$
Parameters: $\boldsymbol{W}_{\mathrm{Q}}, \boldsymbol{W}_{\mathrm{K}} \in \mathbb{R}^{D \times D_{\mathrm{QK}}}; \boldsymbol{W}_{\mathrm{V}}, \boldsymbol{W}_{\mathrm{O}}^{\top} \in \mathbb{R}^{D \times D_{\mathrm{VO}}}$
Output: $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{B \times S \times D}$
$\boldsymbol{Q}_s \leftarrow [\boldsymbol{\mathcal{X}}]_{\cdot,s,\cdot} \boldsymbol{W}_{\mathrm{Q}}$ for $s = 1, \ldots, S$
$\boldsymbol{K}_s \leftarrow [\boldsymbol{\mathcal{X}}]_{\cdot,s,\cdot} \boldsymbol{W}_{\mathrm{K}}$ for $s = 1, \ldots, S$
$\boldsymbol{V}_s \leftarrow [\boldsymbol{\mathcal{X}}]_{\cdot,s,\cdot} \boldsymbol{W}_{\mathrm{V}}$ for $s = 1, \ldots, S$
Construct mask $\boldsymbol{M}_s \in \mathbb{R}^{B \times B}$ for $s = 1, \ldots, S$:
  $[\boldsymbol{M}_s]_{b_1, b_2} = 0$ if generations b_1, b_2 have the same prompt and are incomplete at token s.
  $[\boldsymbol{M}_s]_{b_1, b_2} = -\infty$ otherwise.
$[\boldsymbol{\mathcal{Y}}]_{\cdot,s,\cdot} \leftarrow \mathrm{Softmax}\left(\frac{\boldsymbol{Q}_s \boldsymbol{K}_s^{\top}}{\sqrt{D_{\mathrm{QK}}}} + \boldsymbol{M}_s\right) \boldsymbol{V}_s \boldsymbol{W}_{\mathrm{O}}$ for $s = 1, \ldots, S$
Return: $\boldsymbol{\mathcal{Y}}$