Turbo Connection: Reasoning as Information Flow from Higher to Lower Layers

Mohan Tang [1] **Sidi Lu** [1]

Abstract

Complex problems, whether in math, logic, or planning, are solved by humans through a sequence of steps where the result of one step informs the next. In this work, we adopt the perspective that the reasoning power of Transformers is fundamentally limited by a fixed maximum number of steps along any latent path of computation. To address this, we introduce Turbo Connection (TurboConn), a novel architecture that overcomes the fixed-depth constraint by routing multiple residual connections from the higher-layer hidden states of each token t to the lower layers of token $t + 1$. Fine-tuning pre-trained LLMs with our method not only yields accuracy gains of 0.9% to over 10% on benchmarks like GSM8K, Parity, and multi-step arithmetic, but also demonstrates that the density of these backward connections is critical; our dense interaction significantly outperforms "sparse" alternatives that only pass a single hidden state or vector. Notably, TurboConn can be integrated into pre-trained LLMs to overcome task-specific plateaus: while a fine-tuned Qwen-3-1.7B achieves only 53.78% on Parity, adding our architectural modification enables the model to reach 100% accuracy, all without the necessity to retrain the full model from scratch or sophisticated curriculum learning. Our results provide strong empirical evidence that the depth of the computational path is a key factor in reasoning ability, also offering a new mechanism to enhance LLMs without significantly affecting generation latency.

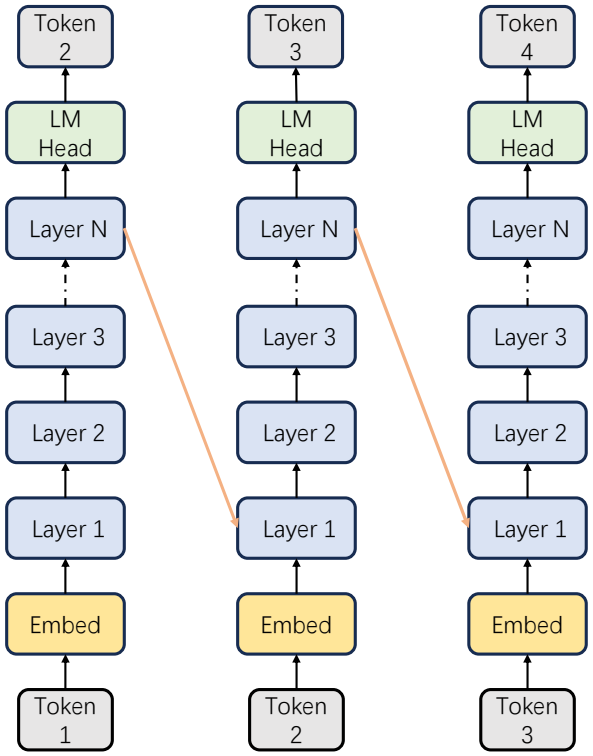

Figure 1. Modified Transformer architecture with downward connections (orange arrows) from higher to lower decoder layers.

1. Introduction

While Transformer-based Large Language Models (LLMs) have advanced significantly in recent years, they continue to exhibit shortcomings on tasks that demand complex reasoning. The popular Chain-of-Thought (CoT) framework (Wei et al., 2022) addresses this by allocating dynamic computation through intermediate steps. However, this approach places a considerable strain on computational resources and often requires specialized training data (DeepSeek-AI et al., 2025).

A growing area of research (Dehghani et al., 2019; Geiping et al., 2025; Fan et al., 2025a) focuses on performing additional computation in the depth dimension of a Transformer rather than along the token sequence dimension. This is achieved by recursively applying Transformer layers, which enables increased latent reasoning. In this approach, "thinking" occurs within the model's hidden states instead of being explicitly projected onto tokens, potentially allowing for more complex information processing. A key advantage of this approach is that it allows training on unlabeled

[1]UCLA. Correspondence to: Mohan Tang <tangmohanp@outlook.com>.

Preprint. July 14, 2026.

pre-training data (Geiping et al., 2025; Zeng et al., 2025). However, it still requires increased time and GPU costs during training, as well as longer inference times that scale proportionally with the number of recursion steps. These factors pose considerable scalability concerns for future development.

Is enhancing reasoning solely a matter of increasing the total amount of computation? We argue that the answer is no. In this work, we aim to increase the effective depth—defined as the maximum length of the computational path available to process information—with negligible impact on total floating-point operations. In standard Transformers, information flows strictly from lower to higher layers. As a result, the maximum length of the computational path is always bounded by a fixed number (proportional to the depth of the model). Because the model's depth is fixed regardless of input length, this architecture imposes limitations on the Transformer's computational power. Aligning with this perspective, Merrill & Sabharwal (2023) demonstrated that finite-depth Transformers are confined to the uniform TC^0 complexity class, a class of problems solvable by constant-depth circuits, suggesting they may lack the expressive power required for inherently sequential problems. Feng et al. (2023) also presents a viewpoint that CoT works by increasing the "effective depth" of Transformers as the outputs are repeatedly looped back to the input.

Overcoming this constraint, we introduce a novel architecture featuring connections from higher layers back to lower layers. Specifically, the hidden state from a higher layer of token t is fed into a lower layer processing token $t+1$. This modification allows the effective reasoning depth to grow linearly with the sequence length, breaking the fixed-depth constraint of standard Transformers with only a marginal increase in total computation. We refer to our method as Turbo Connection (TurboConn). The $t \to t+1$ design is a deliberate choice that leverages the autoregressive nature of the Transformer decoder to avoid the circular dependencies that would arise from self-connections (e.g., $t \to t$), a point we discuss in detail in Section 3.2. The primary trade-off is the loss of full parallelism during training and the prefill stage of inference, as processing degenerates to group-sequential evaluations. Nevertheless, this has little impact on GPU memory usage or the speed of autoregressive generation.

A key advantage of TurboConn is that it requires no modifications to the standard loss function, making it compatible with existing pre-training objectives, though our experiments are focused on the fine-tuning stage due to resource constraints. For this work, we focus on evaluating its effectiveness by fine-tuning Llama 3 1B and 8B models (Grattafiori et al., 2024) and Qwen 3 1.7B model (Yang et al., 2025) on several reasoning-intensive datasets. We find that our architecture consistently outperforms the standard Transformer architectures across these tasks.

To summarize our contributions:

1. We propose a novel method that increases the reasoning ability of large language models, without:
   - additional latency at inference-time autoregressive generation,
   - increased GPU memory consumption during training or inference, or
   - the need for human-annotated chain-of-thought data.
2. Across different models and tasks, TurboConn yields accuracy gains ranging from 0.9% to over 10% on reasoning benchmarks.
3. We show that TurboConn enables a qualitative shift in model behavior, including better *length generalization* on the Parity task and emergence of *discriminative filtering*.
4. Our results provide strong empirical evidence that the lack of depth in latent computation indeed limits the reasoning ability of LLMs. Having a fixed computational depth for all tokens is likely a necessary consequence of the standard parallel training paradigm for Transformers, where all tokens in a sequence are processed simultaneously. This finding could motivate more research to move beyond purely parallel designs and explore sequential or non-parallel training paradigms to unlock deeper reasoning.

2. Related Work

2.1. Chain-of-Thought

People observe that large language models benefit from generating more tokens to "think step by step" (Wei et al., 2022; Kojima et al., 2022). Theoretically, chain-of-thought has been shown to be able to greatly enhance LLMs in terms of computational power (Merrill & Sabharwal, 2024; Li et al., 2024). To train models to produce these reasoning chains, researchers have explored several paradigms. The most direct method is supervised fine-tuning on chain-of-thought data (Liu et al., 2025a; Yue et al., 2023; Ahmad et al., 2025). A more challenging but highly sought-after goal is to elicit CoT reasoning using only final outcomes for supervision (Zelikman et al., 2022; Zhu et al., 2023; Singh et al., 2024). The most prominent current approach in this domain is reinforcement learning from outcome-based rewards (OpenAI et al., 2024; DeepSeek-AI et al., 2025; Team et al., 2025; Liu et al., 2025b). Recent work has pushed these boundaries even further. For instance, some methods aim to train CoT behavior on unlabeled data using only the standard next-token prediction objective (Zelikman et al., 2024).

While the standard CoT framework relies on generation of

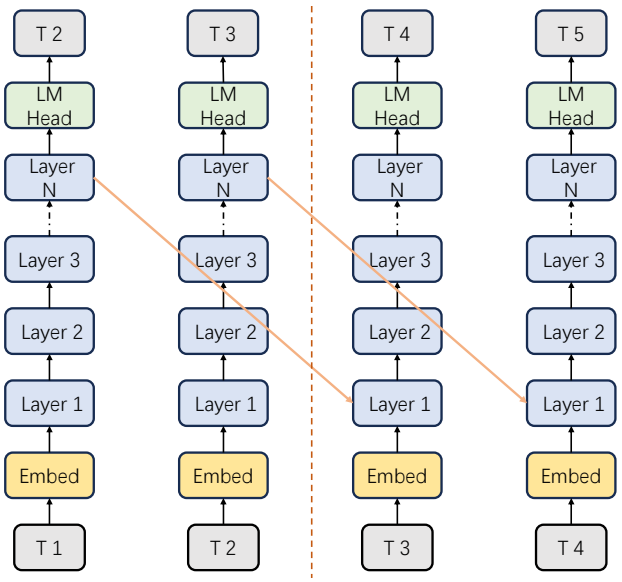

Figure 2. Grouping strategy for downward connections. Example shown for group of 2.

discrete tokens, others have explored continuous Chain-of-Thought, where the discrete reasoning tokens are replaced with continuous representations derived from the model's hidden states (Hao et al., 2024). This method enables continuous reasoning by passing the top-layer hidden states back to the initial embedding layer on dedicated ⟨pause⟩ tokens inserted between problems and answers. However, training such continuous thinking is challenging, requiring extensive curriculum learning from existing CoT datasets.

2.2. Looped/Universal Transformers

Looped or universal Transformers apply the same set of Transformer layers recursively in the depth dimension before producing a final output. This architectural paradigm has shown advantages in reasoning tasks, as demonstrated in several experiments (Dehghani et al., 2019; Saunshi et al., 2025). Theoretically, looped Transformers have been proven capable of solving various logical problems of interest (Giannou et al., 2023; Fan et al., 2025b). More recently, variants of this approach have been successfully scaled up to the billion-parameter range and trained on large, general-purpose pre-training datasets (Geiping et al., 2025; Zeng et al., 2025).

2.3. On Relationship between Depth and Reasoning

Merrill & Sabharwal (2023) theoretically showed that finite-depth Transformers are confined to the uniform TC^0 complexity class, which is believed to have limited computational power. This theoretical constraint has practical implications: assuming the standard conjecture that $L \neq P$, Transformers cannot solve problems such as linear equation solving or universal context-free grammar recognition. Feng et al. (2023) also shows that bounded-depth Transformers cannot solve certain basic mathematical tasks unless the model size grows super-polynomially with respect to the input length and explains why CoT works through the lens of increased "effective depth."

Empirical evidence also highlights the critical role of depth in how models perform multi-step reasoning. For instance, LLMs have been shown to resolve two-hop problems by handling the first hop in lower layers before processing the second hop in higher layers (Biran et al., 2024). Similarly, intensive training on multi-hop reasoning tasks leads to the gradual strengthening of intermediate representations in the middle layers, which are then used by subsequent layers to reach a final answer (Wang et al., 2025). Our work is distinguished from these approaches by isolating the effect of depth alone, without a significant increase in computation, to empirically show its importance.

Particularly relevant is the "back-patching" experiment by Biran et al. (2024), which showed that manually injecting a hidden state from a higher layer back into a lower layer can correct reasoning failures. Our work can be seen as operationalizing this insight: instead of using it as a post-hoc analysis or intervention that requires a second, full forward pass, we integrate this top-to-bottom connection directly into the model's architecture.

Among novel architectural designs, Fan et al. (2021) enables top-down information flow by replacing the Transformer's standard key-value memory with a single key-value set per token that encapsulates information across all layers. Similarly, Staircase Attention (Ju et al., 2022) introduces a recurrent architecture that applies a shared Transformer block iteratively over overlapping token groups, feeding hidden states from previous steps back into the core network as new tokens are processed. While both methods also provide feedback from higher to lower layers, our work is uniquely designed to augment existing, large-scale pre-trained LLMs. TurboConn successfully leverages expensive, pre-trained features of LLMs by using zero-initialized additive connections.

3. Method

3.1. Background and Notations

Consider the following formalization of the Transformer architecture. Let $\boldsymbol{h}_l^{(i)}$ be the hidden state of token i at layer l for $l = 0, \ldots, L$. Given input tokens $t_0,\ t_1,\ t_2, \ldots,\ t_k$, obtain their embeddings

$$\boldsymbol{h}_0^{(i)} = \boldsymbol{e}_i = \mathrm{Embed}(t_i), \qquad i = 0, \ldots, k.$$

For $l = 1, \ldots, L$ and $i = 0, \ldots, k$,

$$\boldsymbol{h}_l^{(i)} = \text{LayerBlock}_l\big(\boldsymbol{h}_{l-1}^{(i)};\, \boldsymbol{h}_{l-1}^{(0)}, \boldsymbol{h}_{l-1}^{(1)}, \ldots, \boldsymbol{h}_{l-1}^{(i-1)}\big)$$

Here $\text{LayerBlock}(\cdot)$ encapsulates the self-attention, MLP, positional encodings, residuals, etc.

The final logits are

$$P^{(i)} = \text{lm\_head}\big(\boldsymbol{h}_L^{(i)}\big), \qquad i = 0, \ldots, k.$$

3.2. Our Approach

We introduce a novel modification to the standard Transformer architecture by incorporating downward connections from higher layers to lower layers, as shown in Figure 1.

In contrast to our approach, an intuitive design might be to feed a token's higher layers back into its own lower layers ($t \to t$). However, this creates a circular dependency in the computational graph, a logical impossibility in a single forward pass. To resolve this loop, the model would need to re-run its layers multiple times for a single token, a process that dramatically increases the required computation at inference time. In contrast, TurboConn avoids this problem. The causal nature of the Transformer decoder means that token t (past) can influence token $t + 1$ (future) without creating any circular dependency in the computational graph.

Formally, for $l = 1, \ldots, L$ and $i = 1, \ldots, k$, we modify the hidden state computation as such:

- When a connection from layer s to layer l exists:

$$\boldsymbol{h}_l^{(i)} = \text{LayerBlock}_l\big(\boldsymbol{h}_{l-1}^{(i)}; \boldsymbol{h}_{l-1}^{(0)}, \ldots, \boldsymbol{h}_{l-1}^{(i-1)}\big) + \alpha \cdot \mathcal{D}\big(\boldsymbol{h}_s^{(i-1)}\big)$$

- Otherwise (when no connection $(s \to l)$ exists):

$$\boldsymbol{h}_l^{(i)} = \text{LayerBlock}_l\big(\boldsymbol{h}_{l-1}^{(i)}; \boldsymbol{h}_{l-1}^{(0)}, \ldots, \boldsymbol{h}_{l-1}^{(i-1)}\big)$$

$\mathcal{D}(\cdot)$ is a learnable linear projection specific to each connection. We can initialize $\mathcal{D}(\cdot)$ to be all zeros when finetuning, so that the model behaves the same as the original LLM at the start of training. The multiplier α scales the strength of downward connections. It does not affect the model's theoretical representational power, but encourages greater utilization of the connections during training. By default we set α to 100. The effect of different multiplier values is analyzed in Section 4.4. Note that there can be multiple connections, as shown in Figure 3.

Because $\boldsymbol{h}_l^{(i)}$ depends only on tokens $0, \ldots, i$, the computation can proceed token-by-token from left to right. TurboConn preserves the standard language modeling paradigm and can be trained using the conventional autoregressive loss:

$$\mathcal{L} = -\sum_{i=1}^{k} \log P(t_i \mid t_0, t_1, \ldots, t_{i-1})$$

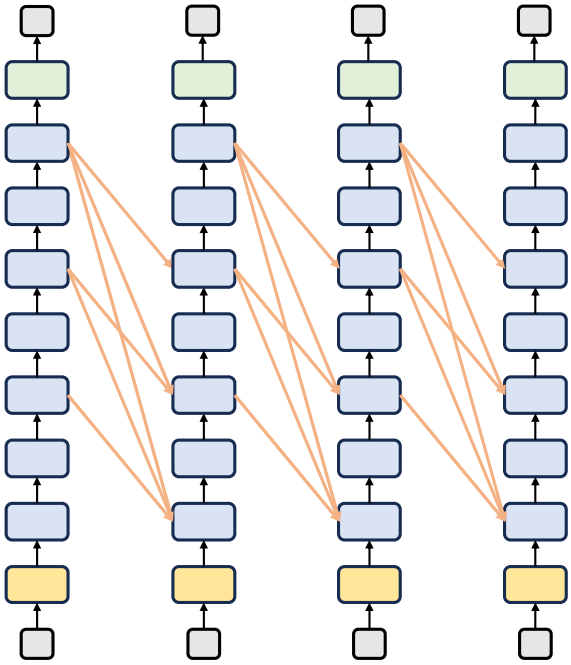

Figure 3. This diagram illustrates the dense connectivity pattern utilized in our experimental setup, depicted here for a group size of 1.

3.2.1. Analysis of Depth

As an example, consider the case that we have a connection from the highest layer to the lowest layer. In standard Transformers, the maximum computational path length is L (corresponding to the number of layers). With TurboConn, this path length extends to kL, where k is the sequence length. The maximum depth of reasoning now scales linearly with the number of input tokens. While this does not enable the model to solve all arbitrarily complex problems, the shift from a constant to a linear-depth computational model is a fundamental breakthrough compared to the standard Transformer.

3.3. Analysis of Computational Cost

The downward connections introduce sequential dependencies between tokens, breaking the parallelism typically available during training. This requires sequential processing of tokens during the forward pass, potentially increasing training latency. Similarly, the prefill stage during inference degenerates to sequential processing, introducing a

potential latency bottleneck, though this can be mitigated by evaluating tokens in larger groups. However, this sequential constraint does not impact the generation phase at inference time, as autoregressive generation inherently processes tokens one by one.

Regarding memory requirements, TurboConn introduces only marginal additional hidden states that need to be stored, resulting in minimal increase in GPU memory usage. The memory footprint remains comparable to standard Transformers.

3.4. Grouping

To mitigate the computational cost of sequential processing, we introduce a grouping mechanism that enables more efficient parallel computation. As shown in Figure 2, we group tokens together and send information back in groups rather than processing individual tokens sequentially. This approach processes tokens in groups of size g, where each group receives downward connections simultaneously. For layers $l = 1, 2, \ldots, L$ and token positions $i = g, g+1, \ldots, k$, we modify the hidden state computation as follows: when a connection from layer s to layer l $(s \to l)$ exists:

$$\boldsymbol{h}_l^{(i)} = \text{LayerBlock}_l\big(\boldsymbol{h}_{l-1}^{(i)}; \boldsymbol{h}_{l-1}^{(0)}, \ldots, \boldsymbol{h}_{l-1}^{(i-1)}\big) + \alpha \cdot \mathcal{D}\big(\boldsymbol{h}_s^{(i-g)}\big)$$

In this way, $\boldsymbol{h}^{(i)}$ does not depend on hidden states at tokens $i-g+1, i-g+2, \ldots, i-1$, so computations within each group can be performed in parallel. With group size g, the computational depth becomes kL/g (assuming there is a connection from top to bottom), and the number of sequential steps reduces from k to k/g. This reduces training latency while decreasing computational power. The group size can be chosen based on the specific use case requirements. Applications requiring deep sequential reasoning may benefit from smaller group sizes, while those prioritizing training efficiency may prefer larger groups.

3.5. Comparison with Universal Transformers

While both TurboConn and the Universal Transformer utilize recurrence to increase depth, the Universal Transformer is recurrent in depth (per token), whereas TurboConn is recurrent across the sequence (cross-token). To clarify the mechanical differences, we compare the two architectures under the assumption that both are configured to increase the effective reasoning depth by a factor of D.

3.5.1. Computational Efficiency and Scaling

To increase the effective reasoning depth by a factor of D, the Universal Transformer must perform D recursive iterations for **every token**. Its training and inference time scale linearly with D. In contrast, TurboConn scales the depth by utilizing the sequence dimension. Although it breaks full token-level parallelism, it preserves intra-group and tensor-wise parallelism. Therefore, the training time increases by a factor significantly less than D.

3.5.2. Memory Footprint

For the Universal Transformer to achieve a depth factor of D, the memory used must grow by a factor of D, as the number of states that must be stored for each recursion increases. Conversely, TurboConn introduces no significant change in memory cost.

3.5.3. Autoregressive Generation and Inference

The difference is also pronounced during the autoregressive generation phase. For the Universal Transformer, each generated token requires D passes through the recurrent layers, which increases the latency by a factor of D. TurboConn, however, involves little additional cost during generation. The backward connections are integrated into the single forward pass of the model, allowing for depth-scaling without the linear latency penalty of Universal Transformers.

4. Experiments

4.1. Reasoning Tasks

We evaluate the effectiveness of TurboConn on the following datasets. Each dataset consists of about 380K questions:

1. **Parity** (Fan et al., 2025a; Banino et al., 2021; Graves, 2017): Given a sequence of 0s and 1s, the task is to determine whether there is an odd number of 1s. We sample sequences ranging from 1 to 70 digits in length.

2. **Multi-step Arithmetic** (Srivastava et al., 2023): Randomly generated arithmetic expressions using digits 0-9, with the number of operands ranging from 1 to 30. The result is projected to modulo 10 for a more uniform distribution of answers.

3. **GSM8K** (Cobbe et al., 2021): A collection of grade school math problems. We use an enlarged dataset augmented by Deng et al. (2023). To evaluate performance without process supervision, we removed the chain-of-thought reasoning steps from the original dataset.

We train and evaluate on different splits of the same distribution. Further details on training data can be found in Appendix A.4.

4.2. Training Hyperparameters

We evaluate performance by fine-tuning pre-trained Llama 3 and Qwen 3 models, comparing a standard Transformer baseline against the Transformer augmented with TurboConn under identical training settings. Training uses a batch size of 64, resulting in approximately 6,000 iterations per epoch for each dataset. We train for a maximum of 3 epochs, following standard fine-tuning practices to prevent overfitting.

Each model has a dense connection setup with 15 to 45 connections. This configuration empirically provides more stable training.

Due to resource constraints, we use LoRA on attention and feed-forward parameters. To ensure fair comparison, we set the rank to r = 120 for TurboConn and r = 140 for the baseline, maintaining comparable parameter counts. The downward connection projections $\mathcal{D}(\cdot)$ are also implemented as low-rank linear maps.

We employ a cosine scheduler (Loshchilov & Hutter, 2017), where the lr increases and decreases in periods of 1000 steps. In each period, it is warmed up from 0 to the highest value in 100 steps, and then gradually decrease to 0 again following a Cosine function with period 900.

Additional training details are provided in Appendix A.

4.3. Main Results

We evaluate our method on Llama 3.2 1B and Llama 3.1 8B models (Grattafiori et al., 2024) and Qwen 3 1.7B model. We use a group size of 4 and set the connection multiplier to 100. Results are presented in Table 2. TurboConn demonstrates significant improvements over standard Transformers across all model sizes and datasets. Notably, the 1B model with TurboConn outperforms the 8B model without connections on the Parity task, suggesting that improved information flow can be more beneficial than increased computational scale for certain reasoning tasks. Moreover, adding TurboConn to pretrained LLM enables Qwen-3-1.7B to achieve 100% accuracy on Parity, where fine-tuning alone fails to resolve the underlying complexity and plateaus at 53.78%.

4.4. Effect of Group Sizes and Multiplier

In this section, we evaluate the effect of different group sizes g on both model performance and computational efficiency. Furthermore, we find that the choice of group size dictates the optimal setting for the connection multiplier (α). We conduct our analysis using the Llama 3.2 1B model across all three datasets.

To ensure consistent experimental setup, we remove a small number of overly long examples (ones exceeding 168 tokens using Llama 3.2 1B tokenizer) from GSM8K so that all experiments can be run with identical settings. Based on this threshold, only 44 examples were removed from the original dataset of 384,620. Each measurement is performed on a single 80G A100 GPU.

4.4.1. Training Latency Analysis

Table 1 presents computational efficiency analysis across different group sizes. The results demonstrate that larger group sizes lead to reduced training latency. Importantly, when we recurse for k/g times, the training latency increases by less than a factor of k/g. This occurs because our sequential token processing preserves other forms of parallelism, including batch-level parallelism and vectorized operations within the attention and feed-forward computations. As shown in Table 1, on the Parity task, group size 4 requires 38.81 recursions yet increases training time by only 4.87×. Moreover, Parity with group size 16 achieves 10× deeper reasoning paths with just 1.36× training overhead–delivering enhanced reasoning capability at near-baseline computational cost. This reveals a favorable trade-off between reasoning depth and computational efficiency, suggesting that our method can provide substantial improvements in model reasoning capabilities with manageable increases in training time.

4.4.2. Performance Analysis

The results are shown in Table 1. In general, we can see that smaller group sizes result in stronger performance. One issue we observed is that when using a large multiplier ($\alpha = 100$) and a large group size, the performance can sometimes degrade compared to the baseline. This suggests that while the downward connections pass valuable, high-level information from previous tokens, the reduced computational depth of larger groups may be insufficient to process this information effectively. The influx of strong signals from higher layers can destabilize the training process. When using a high multiplier, the strong signals from downward connections may also cause the model to over-rely on the connections for information propagation between tokens at the expense of its standard attention mechanisms.

Conversely, setting the multiplier to a lower value ($\alpha = 1$) mitigates this issue, leading to consistent performance gains over the standard Transformer across all tested group sizes. On the other hand, this approach results in less significant improvements when group size is small, compared to using $\alpha = 100$. A smaller α value encourages the model to utilize the downward connections more subtly, resulting in steady, incremental advantages without destabilizing the training process. Therefore, the recommended approach is to pair a small group size with a large multiplier and a larger group size with a small multiplier.

Table 1. Training efficiency and performance analysis for Llama 3.2 1B across group sizes for multipliers $\alpha = 1$ and $\alpha = 100$. We report average sequence length, average number of recursions per training iteration, and relative training time per step compared to the baseline transformer across three reasoning tasks.

Dataset (Avg. Seq. Length)	Method	# Recursions	Time/Step (seconds) (× Transformer)	Acc (%) ($\alpha = 1$)	Acc (%) ($\alpha = 100$)
Parity (154.69 tokens)	Baseline	1.00	1.18 (1.00×)	92.87	–
	TurboConn (Group 4)	38.81	5.75 (4.87×)	98.94	**100.00**
	TurboConn (Group 6)	25.94	3.07 (2.60×)	98.68	**100.00**
	TurboConn (Group 8)	19.84	2.66 (2.25×)	98.83	51.42
	TurboConn (Group 16)	10.00	1.61 (1.36×)	98.85	51.40
Multi-step Arithmetic (125.75 tokens)	Baseline	1.00	1.01 (1.00×)	38.16	–
	TurboConn (Group 4)	31.81	4.46 (4.42×)	40.13	**42.66**
	TurboConn (Group 6)	21.37	2.29 (2.27×)	39.94	41.74
	TurboConn (Group 8)	16.13	1.82 (1.80×)	39.79	41.48
	TurboConn (Group 16)	8.17	1.45 (1.44×)	38.79	38.79
GSM8K (No CoT) (101.92 tokens)	Baseline	1.00	0.77 (1.00×)	7.20	–
	TurboConn (Group 4)	25.86	3.22 (4.18×)	7.67	**8.32**
	TurboConn (Group 6)	17.40	1.73 (2.25×)	7.87	7.95
	TurboConn (Group 8)	13.18	1.39 (1.81×)	7.55	8.17
	TurboConn (Group 16)	6.84	1.38 (1.80×)	7.41	6.94

4.5. Comparison to Single "Soft-Token" Feedback

Chain-of-thought (CoT) can also be conceptualized as a mechanism for passing information from higher to lower layers, by feeding back the information of a single discrete token into the input embedding of the next step. Similarly, methods like Coconut (Hao et al., 2024) utilize a continuous CoT approach, passing the highest hidden state back to the lowest layer, effectively passing information on the distribution of the next token. In contrast, our method is more "dense," passing hidden states from multiple layers back to multiple layers simultaneously. To determine if this architectural density is providing extra reasoning power, we compare our approach against a single "soft-token" feedback baseline.

We implement a feedback mechanism that passes only a single vector of information from the top of the model back to the input. Specifically, the input to the first layer for token i, denoted as $h_0^{(i)}$, is modified as follows:

$$h_0^{(i)} = (1 - \lambda)e_i + \lambda \sum_{j \in \mathcal{V}} P(x_i = j | x_{<i}) \text{Emb}(j)$$

where e_i is the original embedding of token i, Emb is the embedding matrix, and λ is a scaling factor (set to 0.1) representing the strength of the feedback. In this setup, $\lambda = 1$ would imply the prediction of the previous token completely overwrites the current input.

4.5.1. Results and Analysis.

As shown in Table 3, the single soft-token approach is insufficient to capture the reasoning gains of our dense architecture. These results suggest that multi-layer downward connections offer a more effective mechanism for unlocking the reasoning power of depth-scaling than the sparse feedback of a single vector or distribution.

4.6. Length Generalization Experiment

We hypothesize that models with TurboConn learn fundamentally different problem representations compared to the standard Transformer. This is particularly evident in the Parity task, where our method achieves perfect accuracy. To further verify this claim, we conducted a length generalization experiment using the Llama 3.1 8B model. TurboConn (with a group size of 1), "soft-token" method, and the baseline Transformer were all trained exclusively on parity sequences with a maximum length of 10. We then evaluated their performance on much longer sequences.

The results are shown in Figure 4. While all models achieved near-perfect accuracy on the training data, their generalization capabilities diverged significantly. As we can

Table 2. Performance comparison across model sizes and datasets, comparing standard Transformers baseline against Transformers augmented with TurboConn.

Model	Dataset	Method	Acc (%)
Llama 3.2 1B	GSM8K (No CoT)	Baseline	7.20
		TurboConn	**8.32**
	Multi-step-arithmetic	Baseline	38.16
		TurboConn	**42.66**
	Parity	Baseline	92.87
		TurboConn	**100.0**
Llama 3.1 8B	GSM8K (No CoT)	Baseline	23.92
		TurboConn	**24.82**
	Multi-step-arithmetic	Baseline	48.32
		TurboConn	**51.86**
	Parity	Baseline	89.40
		TurboConn	**100.0**
Qwen 3 1.7B	GSM8K (No CoT)	Baseline	15.90
		TurboConn	**20.31**
	Multi-step-arithmetic	Baseline	36.10
		TurboConn	**45.81**
	Parity	Baseline	53.78
		TurboConn	**100.0**

see from the plot, the LLM with TurboConn shows much better length generalization ability than Transformer and soft-token method. In particular, our method maintains perfect accuracy on sequences up to length 30. This suggests that TurboConn allows the model to learn a more robust and generalizable algorithm for the task, whereas the standard Transformer appears to overfit to the training distribution, leveraging its vast parameter count to solve the problem.

4.7. Emergence of Discriminative Filtering

Beyond improvements in top-1 accuracy, we observe that TurboConn results in a fundamental change to the output distribution of the models, and enables the emergence of *discriminative filtering*. We define this as the model's ability to confidently eliminate mathematically incorrect candidates from the output probability distribution.

To quantify this effect, we measure the number of "eliminated choices," defined as the count of digits $\{0, \ldots, 9\}$ assigned a probability below a threshold of $\tau = 0.001$ by the models after finetuning. As shown in Table 4, on the Multi-step Arithmetic task, Qwen-1.7B model augmented with TurboConn eliminates an average of **5.259** choices per token, more than doubling the **2.516** choices eliminated by the standard Transformer baseline.

Table 3. Comparison of our dense feedback method against standard Transformers baseline and the single soft-token feedback using Llama 3.2 1B.

Model	Dataset	Method	Acc (%)
Llama 3.2 1B	GSM8K (No CoT)	Baseline	7.20
		Soft-token	7.07
		TurboConn	**8.32**
	Multi-step Arithmetic	Baseline	38.16
		Soft-token	38.00
		TurboConn	**42.66**
	Parity	Baseline	92.87
		Soft-token	96.51
		TurboConn	**100.0**

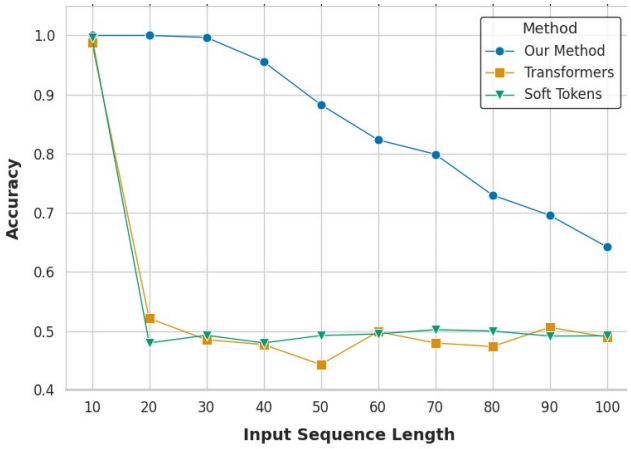

Figure 4. Length Generalization Performance. We evaluate a Llama 3.1 8B model, trained on Parity with up to 10-digit sequences, on its ability to generalize to longer input sequences.

A particularly striking result is that while increasing the model scale allows standard Transformers to match our accuracy, their discriminative power remains significantly lower. Qwen 3 1.7B model with TurboConn eliminates more choices (5.259) than the 8B baseline (4.500), despite achieving nearly identical accuracy (45.81% vs 45.61%). This signals a fundamental gap in this reasoning dimension that cannot be easily filled by parameter scaling alone, suggesting that the depth afforded by our method enables a more efficient internal verification process.

4.8. Synergy with Chain-of-Thought Reasoning

While TurboConn enhances the internal reasoning capabilities of models without explicit intermediate tokens, it is also compatible with standard Chain-of-Thought prompting.

Table 4. Comparison of accuracy and discriminative filtering on Multi-step Arithmetic after finetuning. "Eliminated Choices" represents the mean number of digit classes with probability $P < 0.001$. The result for TurboConn is reported using a group size of 4.

Model	Method	Acc (%)	Eliminated Choices
Qwen3-1.7B	Baseline	36.10	2.516
Qwen3-1.7B	**TurboConn**	**45.81**	**5.259**
Qwen3-4B	Baseline	43.68	4.609
Qwen3-8B	Baseline	45.61	4.500

One way to combine those methods is to augment the explicit reasoning of CoT with the latent reasoning ability of backward connections. We hypothesize that TurboConn can act as an error-correction mechanism for explicit reasoning chains, allowing the model to more accurately follow the "program" laid out in the CoT.

To test this synergy, we utilize the NuminaMath-CoT dataset (LI et al., 2024) and augmented GSM8K (Deng et al., 2023) dataset to generate CoT trajectories. We first fine-tune a Llama 3.2-1B model to produce reasoning chains for mathematical problems. We then compare models with TurboConn against the standard Transformers baseline by finetuning both to predict the final numerical outcome based on the same generated CoTs. This setup ensures that both models process the same explicit reasoning steps, isolating the effect of latent computation on final answer derivation. More details on this experiment are provided in Appendix A.7.

As shown in Table 5, TurboConn improves the model's ability to reach the correct conclusion from a given reasoning chain. These results suggest that TurboConn can effectively use latent computation to verify intermediate steps and correct potential errors within the explicit reasoning process, leading to more robust mathematical performance.

We note that the performance gains on GSM8K are less pronounced compared to those on NuminaMath. Since the training data for GSM8K is augmented using GPT-4, there is a significant distribution shift between the augmented training data and the original test/validation splits, potentially reducing the utility of the learned representations. We believe that our method would benefit more from access to sufficient higher-quality data that closely matches the target distribution.

4.8.1. Generalization to Competition Mathematics

To further evaluate the robustness of our method, we test the checkpoints trained on the NuminaMath dataset on recent competition mathematics problems. This evaluation set is an aggregate of problems from the 2023 AMC and 2024–2025 AIME competitions (Mathematical Association of America, 2023; 2024; 2025). We employ the same "correct existing CoT" setting described above, utilizing the Llama 3.2-1B model finetuned on NuminaMath to generate the initial CoT trajectories, and maintaining identical inference hyperparameters.

As shown in Table 5, on this combined competition dataset, TurboConn demonstrates a significant relative performance gain (4.45% vs. 2.00%). These results indicate that even when a 1B parameter model struggles to generate high-quality explicit reasoning chains for highly complex problems, TurboConn's latent reasoning mechanism is still capable of extracting value from imperfect trajectories to correct errors and improve final performance.

Table 5. Performance comparison using generated Chain-of-Thought trajectories. All results use Llama 3.2-1B; TurboConn uses a group size of 8.

Task	Method	Acc (%)
NuminaMath-CoT	Baseline + CoT	30.96
	TurboConn + CoT	**33.98**
GSM8K	Baseline + CoT	36.45
	TurboConn + CoT	**36.96**
AMC & AIME (Transferred)	Baseline + CoT	2.00
	TurboConn + CoT	**4.45**

5. Conclusions

In this work, we introduce TurboConn, a novel method that creates residual connections from higher to lower layers between sequential tokens, allowing the model's effective reasoning depth to scale linearly with sequence length. Our results demonstrated that this architectural change yields significant performance gains on logical reasoning tasks, with only a manageable increase in computational cost. This confirms empirically that extending effective depth alone is a viable strategy for overcoming the inherent reasoning bottlenecks of fixed-depth Transformers. It would be valuable to explore the potential of this architecture further. Future work could focus on integrating this architecture into the pre-training phase, which may foster the development of more foundational, general-purpose reasoning skills.

Acknowledgment

This work was conducted during the authors' studies at the University of California, Los Angeles. The authors acknowledge Peng's Language Understanding & Synthesis Lab at UCLA for providing the computational resources and infrastructure necessary for the experiments in this study.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Ahmad, W. U., Narenthiran, S., Majumdar, S., Ficek, A., Jain, S., Huang, J., Noroozi, V., and Ginsburg, B. Opencodereasoning: Advancing data distillation for competitive coding, 2025. URL https://arxiv.org/abs/2504.01943.

Banino, A., Balaguer, J., and Blundell, C. Pondernet: Learning to ponder, 2021. URL https://arxiv.org/abs/2107.05407.

Biran, E., Gottesman, D., Yang, S., Geva, M., and Globerson, A. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 14113–14130, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.781. URL https://aclanthology.org/2024.emnlp-main.781/.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Łukasz Kaiser. Universal transformers, 2019. URL https://arxiv.org/abs/1807.03819.

Deng, Y., Prasad, K., Fernandez, R., Smolensky, P., Chaudhary, V., and Shieber, S. Implicit chain of thought reasoning via knowledge distillation, 2023. URL https://arxiv.org/abs/2311.01460.

Fan, A., Lavril, T., Grave, E., Joulin, A., and Sukhbaatar, S. Addressing some limitations of transformers with feedback memory, 2021. URL https://arxiv.org/abs/2002.09402.

Fan, Y., Du, Y., Ramchandran, K., and Lee, K. Looped transformers for length generalization. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://doi.org/10.48550/arXiv.2409.15647.

Fan, Y., Du, Y., Ramchandran, K., and Lee, K. Looped transformers for length generalization, 2025b. URL https://arxiv.org/abs/2409.15647.

Feng, G., Zhang, B., Gu, Y., Ye, H., He, D., and Wang, L. Towards revealing the mystery behind chain of thought: a theoretical perspective. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc. URL https://dl.acm.org/doi/10.5555/3666122.3669222.

Geiping, J., McLeish, S., Jain, N., Kirchenbauer, J., Singh, S., Bartoldson, B. R., Kailkhura, B., Bhatele, A., and Goldstein, T. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025. URL https://arxiv.org/abs/2502.05171.

Giannou, A., Rajput, S., Sohn, J.-y., Lee, K., Lee, J. D., and Papailiopoulos, D. Looped transformers as pro-

grammable computers. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023. URL https://dl.acm.org/doi/10.5555/3618408.3618866.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay,

S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Graves, A. Adaptive computation time for recurrent neural networks, 2017. URL https://arxiv.org/abs/1603.08983.

Hao, S., Sukhbaatar, S., Su, D., Li, X., Hu, Z., Weston, J., and Tian, Y. Training large language models to reason in a continuous latent space, 2024. URL https://arxiv.org/abs/2412.06769.

Ju, D., Roller, S., Sukhbaatar, S., and Weston, J. Staircase attention for recurrent processing of sequences. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088. URL https://dl.acm.org/doi/10.5555/3600270.3601883.

LI, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S. C., Rasul, K., Yu, L., Jiang, A., Shen, Z., Qin, Z., Dong, B., Zhou, L., Fleureau, Y., Lample, G., and Polu, S. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Li, Z., Liu, H., Zhou, D., and Ma, T. Chain of thought empowers transformers to solve inherently serial problems, 2024. URL https://arxiv.org/abs/2402.12875.

Liu, Z., Chen, Y., Shoeybi, M., Catanzaro, B., and Ping, W. Acemath: Advancing frontier math reasoning with post-training and reward modeling, 2025a. URL https://arxiv.org/abs/2412.15084.

Liu, Z., Yang, Z., Chen, Y., Lee, C., Shoeybi, M., Catanzaro, B., and Ping, W. Acereason-nemotron 1.1: Advancing math and code reasoning through sft and rl synergy, 2025b. URL https://arxiv.org/abs/2506.13284.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts, 2017. URL https://arxiv.org/abs/1608.03983.

Mathematical Association of America. 2023 AMC problems and solutions, 2023. URL https://artofproblemsolving.com/wiki/index.php/2023_AMC_12A. Archived by the Art of Problem Solving.

Mathematical Association of America. 2024 AIME i and ii problems and solutions, 2024. URL https://artofproblemsolving.com/wiki/index.php/2024_AIME_I. Archived by the Art of Problem Solving.

Mathematical Association of America. 2025 AIME i and ii problems and solutions, 2025. URL https://artofproblemsolving.com/wiki/index.php/2025_AIME_I. Archived by the Art of Problem Solving.

Merrill, W. and Sabharwal, A. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023. doi: 10.1162/tacl_a_00562. URL https://aclanthology.org/2023.tacl-1.31/.

Merrill, W. and Sabharwal, A. The expressive power of transformers with chain of thought, 2024. URL https://arxiv.org/abs/2310.07923.

OpenAI, Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., Iftimie, A., Karpenko, A., Passos, A. T., Neitz, A., Prokofiev, A., Wei, A., Tam, A., Bennett, A., Kumar, A., Saraiva, A., Vallone, A., Duberstein, A., Kondrich, A., Mishchenko, A., Applebaum, A., Jiang, A., Nair, A., Zoph, B., Ghorbani, B., Rossen, B., Sokolowsky, B., Barak, B., McGrew, B., Minaiev, B., Hao, B., Baker, B., Houghton, B., McKinzie, B., Eastman, B., Lugaresi, C., Bassin, C., Hudson, C., Li, C. M., de Bourcy, C., Voss, C., Shen, C., Zhang, C., Koch, C., Orsinger, C., Hesse, C., Fischer, C., Chan, C., Roberts, D., Kappler, D., Levy, D., Selsam, D., Dohan, D., Farhi, D., Mely, D., Robinson, D., Tsipras, D., Li, D., Oprica, D., Freeman, E., Zhang,

E., Wong, E., Proehl, E., Cheung, E., Mitchell, E., Wallace, E., Ritter, E., Mays, E., Wang, F., Such, F. P., Raso, F., Leoni, F., Tsimpourlas, F., Song, F., von Lohmann, F., Sulit, F., Salmon, G., Parascandolo, G., Chabot, G., Zhao, G., Brockman, G., Leclerc, G., Salman, H., Bao, H., Sheng, H., Andrin, H., Bagherinezhad, H., Ren, H., Lightman, H., Chung, H. W., Kivlichan, I., O'Connell, I., Osband, I., Gilaberte, I. C., Akkaya, I., Kostrikov, I., Sutskever, I., Kofman, I., Pachocki, J., Lennon, J., Wei, J., Harb, J., Twore, J., Feng, J., Yu, J., Weng, J., Tang, J., Yu, J., Candela, J. Q., Palermo, J., Parish, J., Heidecke, J., Hallman, J., Rizzo, J., Gordon, J., Uesato, J., Ward, J., Huizinga, J., Wang, J., Chen, K., Xiao, K., Singhal, K., Nguyen, K., Cobbe, K., Shi, K., Wood, K., Rimbach, K., Gu-Lemberg, K., Liu, K., Lu, K., Stone, K., Yu, K., Ahmad, L., Yang, L., Liu, L., Maksin, L., Ho, L., Fedus, L., Weng, L., Li, L., McCallum, L., Held, L., Kuhn, L., Kondraciuk, L., Kaiser, L., Metz, L., Boyd, M., Trebacz, M., Joglekar, M., Chen, M., Tintor, M., Meyer, M., Jones, M., Kaufer, M., Schwarzer, M., Shah, M., Yatbaz, M., Guan, M. Y., Xu, M., Yan, M., Glaese, M., Chen, M., Lampe, M., Malek, M., Wang, M., Fradin, M., McClay, M., Pavlov, M., Wang, M., Wang, M., Murati, M., Bavarian, M., Rohaninejad, M., McAleese, N., Chowdhury, N., Chowdhury, N., Ryder, N., Tezak, N., Brown, N., Nachum, O., Boiko, O., Murk, O., Watkins, O., Chao, P., Ashbourne, P., Izmailov, P., Zhokhov, P., Dias, R., Arora, R., Lin, R., Lopes, R. G., Gaon, R., Miyara, R., Leike, R., Hwang, R., Garg, R., Brown, R., James, R., Shu, R., Cheu, R., Greene, R., Jain, S., Altman, S., Toizer, S., Toyer, S., Miserendino, S., Agarwal, S., Hernandez, S., Baker, S., McKinney, S., Yan, S., Zhao, S., Hu, S., Santurkar, S., Chaudhuri, S. R., Zhang, S., Fu, S., Papay, S., Lin, S., Balaji, S., Sanjeev, S., Sidor, S., Broda, T., Clark, A., Wang, T., Gordon, T., Sanders, T., Patwardhan, T., Sottiaux, T., Degry, T., Dimson, T., Zheng, T., Garipov, T., Stasi, T., Bansal, T., Creech, T., Peterson, T., Eloundou, T., Qi, V., Kosaraju, V., Monaco, V., Pong, V., Fomenko, V., Zheng, W., Zhou, W., McCabe, W., Zaremba, W., Dubois, Y., Lu, Y., Chen, Y., Cha, Y., Bai, Y., He, Y., Zhang, Y., Wang, Y., Shao, Z., and Li, Z. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Saunshi, N., Dikkala, N., Li, Z., Kumar, S., and Reddi, S. J. Reasoning with latent thoughts: On the power of looped transformers, 2025. URL https://arxiv.org/abs/2502.17416.

Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Garcia, X., Liu, P. J., Harrison, J., Lee, J., Xu, K., Parisi, A., Kumar, A., Alemi, A., Rizkowsky, A., Nova, A., Adlam, B., Bohnet, B., Elsayed, G., Sedghi, H., Mordatch, I., Simpson, I., Gur, I., Snoek, J., Pennington, J., Hron, J., Kenealy, K., Swersky, K., Mahajan, K., Culp, L., Xiao, L., Bileschi, M. L., Constant, N., Novak, R., Liu, R., Warkentin, T., Qian, Y., Bansal, Y., Dyer, E., Neyshabur, B., Sohl-Dickstein, J., and Fiedel, N. Beyond human data: Scaling self-training for problem-solving with language models, 2024. URL https://arxiv.org/abs/2312.06585.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., Kluska, A., Lewkowycz, A., Agarwal, A., Power, A., Ray, A., Warstadt, A., Kocurek, A. W., Safaya, A., Tazarv, A., Xiang, A., Parrish, A., Nie, A., Hussain, A., Askell, A., Dsouza, A., Slone, A., Rahane, A., Iyer, A. S., Andreassen, A., Madotto, A., Santilli, A., Stuhlmüller, A., Dai, A., La, A., Lampinen, A., Zou, A., Jiang, A., Chen, A., Vuong, A., Gupta, A., Gottardi, A., Norelli, A., Venkatesh, A., Gholamidavoodi, A., Tabassum, A., Menezes, A., Kirubarajan, A., Mullokandov, A., Sabharwal, A., Herrick, A., Efrat, A., Erdem, A., Karakaş, A., Roberts, B. R., Loe, B. S., Zoph, B., Bojanowski, B., Özyurt, B., Hedayatnia, B., Neyshabur, B., Inden, B., Stein, B., Ekmekci, B., Lin, B. Y., Howald, B., Orinion, B., Diao, C., Dour, C., Stinson, C., Argueta, C., Ramírez, C. F., Singh, C., Rathkopf, C., Meng, C., Baral, C., Wu, C., Callison-Burch, C., Waites, C., Voigt, C., Manning, C. D., Potts, C., Ramirez, C., Rivera, C. E., Siro, C., Raffel, C., Ashcraft, C., Garbacea, C., Sileo, D., Garrette, D., Hendrycks, D., Kilman, D., Roth, D., Freeman, D., Khashabi, D., Levy, D., González, D. M., Perszyk, D., Hernandez, D., Chen, D., Ippolito, D., Gilboa, D., Dohan, D., Drakard, D., Jurgens, D., Datta, D., Ganguli, D., Emelin, D., Kleyko, D., Yuret, D., Chen, D., Tam, D., Hupkes, D., Misra, D., Buzan, D., Mollo, D. C., Yang, D., Lee, D.-H., Schrader, D., Shutova, E., Cubuk, E. D., Segal, E., Hagerman, E., Barnes, E., Donoway, E., Pavlick, E., Rodola, E., Lam, E., Chu, E., Tang, E., Erdem, E., Chang, E., Chi, E. A., Dyer, E., Jerzak, E., Kim, E., Manyasi, E. E., Zheltonozhskii, E., Xia, F., Siar, F., Martínez-Plumed, F., Happé, F., Chollet, F., Rong, F., Mishra, G., Winata, G. I., de Melo, G., Kruszewski, G., Parascandolo, G., Mariani, G., Wang, G., Jaimovitch-López, G., Betz, G., Gur-Ari, G., Galijasevic, H., Kim, H., Rashkin, H., Hajishirzi, H., Mehta, H., Bogar, H., Shevlin, H., Schütze, H., Yakura, H., Zhang, H., Wong, H. M., Ng, I., Noble, I., Jumelet, J., Geissinger, J., Kernion, J., Hilton, J., Lee, J., Fisac, J. F., Simon, J. B., Koppel, J., Zheng, J., Zou, J., Kocoń, J., Thompson, J., Wingfield, J., Kaplan, J., Radom, J., Sohl-Dickstein, J., Phang, J., Wei, J., Yosinski, J., Novikova, J., Bosscher, J., Marsh, J., Kim, J., Taal, J., Engel, J., Alabi, J., Xu, J., Song, J., Tang, J., Waweru, J., Burden, J., Miller, J., Balis, J. U., Batchelder, J., Berant, J., Frohberg, J., Rozen, J., Hernandez-Orallo, J., Boudeman, J., Guerr, J., Jones, J., Tenenbaum, J. B., Rule, J. S., Chua, J., Kanclerz, K., Livescu, K., Krauth, K., Gopalakrishnan, K., Ignatyeva,

K., Markert, K., Dhole, K. D., Gimpel, K., Omondi, K., Mathewson, K., Chiafullo, K., Shkaruta, K., Shridhar, K., McDonell, K., Richardson, K., Reynolds, L., Gao, L., Zhang, L., Dugan, L., Qin, L., Contreras-Ochando, L., Morency, L.-P., Moschella, L., Lam, L., Noble, L., Schmidt, L., He, L., Colón, L. O., Metz, L., Şenel, L. K., Bosma, M., Sap, M., ter Hoeve, M., Farooqi, M., Faruqui, M., Mazeika, M., Baturan, M., Marelli, M., Maru, M., Quintana, M. J. R., Tolkiehn, M., Giulianelli, M., Lewis, M., Potthast, M., Leavitt, M. L., Hagen, M., Schubert, M., Baitemirova, M. O., Arnaud, M., McElrath, M., Yee, M. A., Cohen, M., Gu, M., Ivanitskiy, M., Starritt, M., Strube, M., Swędrowski, M., Bevilacqua, M., Yasunaga, M., Kale, M., Cain, M., Xu, M., Suzgun, M., Walker, M., Tiwari, M., Bansal, M., Aminnaseri, M., Geva, M., Gheini, M., T, M. V., Peng, N., Chi, N. A., Lee, N., Krakover, N. G.-A., Cameron, N., Roberts, N., Doiron, N., Martinez, N., Nangia, N., Deckers, N., Muennighoff, N., Keskar, N. S., Iyer, N. S., Constant, N., Fiedel, N., Wen, N., Zhang, O., Agha, O., Elbaghdadi, O., Levy, O., Evans, O., Casares, P. A. M., Doshi, P., Fung, P., Liang, P. P., Vicol, P., Alipoormolabashi, P., Liao, P., Liang, P., Chang, P., Eckersley, P., Htut, P. M., Hwang, P., Miłkowski, P., Patil, P., Pezeshkpour, P., Oli, P., Mei, Q., Lyu, Q., Chen, Q., Banjade, R., Rudolph, R. E., Gabriel, R., Habacker, R., Risco, R., Millière, R., Garg, R., Barnes, R., Saurous, R. A., Arakawa, R., Raymaekers, R., Frank, R., Sikand, R., Novak, R., Sitelew, R., LeBras, R., Liu, R., Jacobs, R., Zhang, R., Salakhutdinov, R., Chi, R., Lee, R., Stovall, R., Teehan, R., Yang, R., Singh, S., Mohammad, S. M., Anand, S., Dillavou, S., Shleifer, S., Wiseman, S., Gruetter, S., Bowman, S. R., Schoenholz, S. S., Han, S., Kwatra, S., Rous, S. A., Ghazarian, S., Ghosh, S., Casey, S., Bischoff, S., Gehrmann, S., Schuster, S., Sadeghi, S., Hamdan, S., Zhou, S., Srivastava, S., Shi, S., Singh, S., Asaadi, S., Gu, S. S., Pachchigar, S., Toshniwal, S., Upadhyay, S., Shyamolima, Debnath, Shakeri, S., Thormeyer, S., Melzi, S., Reddy, S., Makini, S. P., Lee, S.-H., Torene, S., Hatwar, S., Dehaene, S., Divic, S., Ermon, S., Biderman, S., Lin, S., Prasad, S., Piantadosi, S. T., Shieber, S. M., Misherghi, S., Kiritchenko, S., Mishra, S., Linzen, T., Schuster, T., Li, T., Yu, T., Ali, T., Hashimoto, T., Wu, T.-L., Desbordes, T., Rothschild, T., Phan, T., Wang, T., Nkinyili, T., Schick, T., Kornev, T., Tunduny, T., Gerstenberg, T., Chang, T., Neeraj, T., Khot, T., Shultz, T., Shaham, U., Misra, V., Demberg, V., Nyamai, V., Raunak, V., Ramasesh, V., Prabhu, V. U., Padmakumar, V., Srikumar, V., Fedus, W., Saunders, W., Zhang, W., Vossen, W., Ren, X., Tong, X., Zhao, X., Wu, X., Shen, X., Yaghoobzadeh, Y., Lakretz, Y., Song, Y., Bahri, Y., Choi, Y., Yang, Y., Hao, Y., Chen, Y., Belinkov, Y., Hou, Y., Hou, Y., Bai, Y., Seid, Z., Zhao, Z., Wang, Z., Wang, Z. J., Wang, Z., and Wu, Z. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023. URL https://arxiv.org/abs/2206.04615.

Team, K., Du, A., Gao, B., Xing, B., Jiang, C., Chen, C., Li, C., Xiao, C., Du, C., Liao, C., Tang, C., Wang, C., Zhang, D., Yuan, E., Lu, E., Tang, F., Sung, F., Wei, G., Lai, G., Guo, H., Zhu, H., Ding, H., Hu, H., Yang, H., Zhang, H., Yao, H., Zhao, H., Lu, H., Li, H., Yu, H., Gao, H., Zheng, H., Yuan, H., Chen, J., Guo, J., Su, J., Wang, J., Zhao, J., Zhang, J., Liu, J., Yan, J., Wu, J., Shi, L., Ye, L., Yu, L., Dong, M., Zhang, N., Ma, N., Pan, Q., Gong, Q., Liu, S., Ma, S., Wei, S., Cao, S., Huang, S., Jiang, T., Gao, W., Xiong, W., He, W., Huang, W., Xu, W., Wu, W., He, W., Wei, X., Jia, X., Wu, X., Xu, X., Zu, X., Zhou, X., Pan, X., Charles, Y., Li, Y., Hu, Y., Liu, Y., Chen, Y., Wang, Y., Liu, Y., Qin, Y., Liu, Y., Yang, Y., Bao, Y., Du, Y., Wu, Y., Wang, Y., Zhou, Z., Wang, Z., Li, Z., Zhu, Z., Zhang, Z., Wang, Z., Yang, Z., Huang, Z., Huang, Z., Xu, Z., Yang, Z., and Lin, Z. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Wang, B., Yue, X., Su, Y., and Sun, H. Grokking of implicit reasoning in transformers: a mechanistic journey to the edge of generalization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385. URL https://dl.acm.org/doi/10.5555/3737916.3740933.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., ichter, b., Xia, F., Chi, E., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. URL https://dl.acm.org/doi/10.5555/3600270.3602070.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yue, X., Qu, X., Zhang, G., Fu, Y., Huang, W., Sun, H., Su, Y., and Chen, W. Mammoth: Building math generalist models through hybrid instruction tuning, 2023. URL https://arxiv.org/abs/2309.05653.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. Star: self-taught reasoner bootstrapping reasoning with reasoning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088. URL https://dl.acm.org/doi/10.5555/3600270.3601396.

Zelikman, E., Harik, G., Shao, Y., Jayasiri, V., Haber, N., and Goodman, N. D. Quiet-star: Language models can teach themselves to think before speaking, 2024. URL https://arxiv.org/abs/2403.09629.

Zeng, B., Song, S., Huang, S., Wang, Y., Li, H., He, Z., Wang, X., Li, Z., and Lin, Z. Pretraining language models to ponder in continuous space, 2025. URL https://arxiv.org/abs/2505.20674.

Zhu, X., Wang, J., Zhang, L., Zhang, Y., Huang, Y., Gan, R., Zhang, J., and Yang, Y. Solving math word problems via cooperative reasoning induced language models. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4471–4485, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.245. URL https://aclanthology.org/2023.acl-long.245/.

A. More Training Details

A.1. Pseudocode Implementation

Algorithm 1 provides the complete pseudocode for our primary training procedure.

Algorithm 1 Forward Pass with Cross-Layer Latent Bridges

Input: Token sequence $\mathbf{X} \in \mathbb{N}^{B \times L}$, group size G, bridge set $\mathcal{B} = \{(src, dest)\}$, multiplier α
Initialize: KV cache $\mathcal{K} \leftarrow \emptyset$, logits list $\mathcal{Y} \leftarrow [\,]$
Initialize: State buffers $\mathbf{S}_{s,d} \leftarrow$ NULL for all $(s, d) \in \mathcal{B}$
for $t = 0, G, 2G, \ldots, L - G$ **do**
  // Extract current group of tokens
  $\mathbf{X}_g \leftarrow \mathbf{X}_{:,\, t:t+G}$
  $\mathbf{H}^{(0)} \leftarrow \text{EmbedTokens}(\mathbf{X}_g)$
  for $i = 0$ **to** $N - 1$ **do**
    // 1. Standard Transformer Computation
    $\mathbf{H}^{(i+1)}, \mathcal{K} \leftarrow \text{TransformerLayer}_i(\mathbf{H}^{(i)}, \mathcal{K})$
    // 2. Injection Phase: Inject latent state from previous group
    for $(s, d) \in \mathcal{B}$ **do**
      if $i = d$ **and** $\mathbf{S}_{s,d} \neq$ NULL **then**
        $\mathbf{H}^{(i+1)} \leftarrow \mathbf{H}^{(i+1)} + \alpha \cdot \text{UpProj}_{s,d}(\mathbf{S}_{s,d})$
      end if
    end for
    // 3. Extraction Phase: Save state for next group
    for $(s, d) \in \mathcal{B}$ **do**
      if $i = s$ **then**
        $\mathbf{S}_{s,d} \leftarrow \text{DownProj}_{s,d}(\mathbf{H}^{(i+1)})$
      end if
    end for
  end for
  Append $\text{LMHead}(\mathbf{H}^{(N)})$ to $\mathcal{Y}$
end for
Output: $\text{Concat}(\mathcal{Y}, \text{dim} = 1)$ *// Shape:* $[B, L, V]$

A.2. Hyperparameters

We set the learning rate for Llama 3 models to 1.92×10^{-5}. This was determined by scaling a base learning rate of 3×10^{-7} proportionally with our batch size of 64. For Qwen 3 models, the learning rate is set to 5.12×10^{-6}, which is $8 \times 10^{-8} \times 64$. We set the decoding temperature to 1.0.

A.3. Model Implementation

We implement TurboConn using a cache. During the forward pass, as the model processes each token, the cache stores the hidden states from the specified higher layers. When processing subsequent tokens, these cached states are retrieved and added to the corresponding lower-layer inputs.

The sequential nature of TurboConn necessitates the use of a key-value (KV) cache during the training forward pass, a mechanism typically reserved only for autoregressive inference. Because tokens (or groups of tokens) are processed sequentially, the attention keys and values from all previous tokens must be cached to be available for the current token's self-attention computation. To implement this efficiently, we store past key and value states in a list of tensors rather than concatenating them. This approach avoids the memory duplication that can occur when new tensors are created through repeated concatenation.

A.4. Example Training Data

Here we offer some examples for the format of data used for training and evaluation.

A.4.1. PARITY

Prompt:
Question: Output the parity of this sequence.
Input: 1 0 0 1 0 1
Answer:

Completion:
1

A.4.2. MULTI-STEP ARITHMETIC

Prompt:
Question: Evaluate this expression modulo 10.
Input: (-(3 + -(1)) * 4 * (8 + 2 + 9)) =
Answer:

Completion:
8

A.4.3. GSM8K

Prompt:
Question: Answer the math question.
Input: John cuts his grass to 2 inches. It grows .5 inches per month. When it gets to 4 inches he cuts it back down to 2 inches. It cost $100 to get his grass cut. How much does he pay per year?
Answer:

Completion:
300

A.5. Connection Configurations

The specific wiring configurations were derived from a simple heuristic. Our principle was to ensure broad layer coverage while maintaining a low computational overhead:

$$\mathcal{C} = \{(s, l) \mid s - l > k, \text{ where } s, l \in [L_{min}, L_{max}] \text{ sampled at interval } m\}$$

We did not conduct an extensive search for "optimal" wiring. The only decision based on experiment is whether to leave the bottom-most layers unchanged. They are generally understood to encode basic features rather than the high-level logic targeted by our recurrence. When training appears unstable, we increase L_{min} from 0 to a value that feels right. This turns out to be sufficient for our method to work.

The other hyperparameters of the connection formula were arbitrarily chosen at the beginning of our experiments. Our primary constraints were to ensure that the total number of trainable parameters did not exceed the standard baseline and that the connections covered the layers relatively evenly. Then these configurations were kept constant throughout all subsequent experiments. Because we conducted one or zero trials per model to determine these hyperparameters, we believe the method is robust to the specific wiring setup.

The specific wiring configurations for different models are listed below. The downward connections are defined from a source layer to a destination layer using the notation: 'source -> destination'.

A.5.1. LLAMA 3.2 1B (16 TOTAL LAYERS)

A total of 15 connections were used, as specified below:

```
6 -> 0      8 -> 0      8 -> 2      10 -> 0     10 -> 2
```

```
10 -> 4    12 -> 0    12 -> 2    12 -> 4    12 -> 6
14 -> 0    14 -> 2    14 -> 4    14 -> 6    14 -> 8
```

A.5.2. LLAMA 3.1 8B (32 TOTAL LAYERS)

A total of 45 connections were used, as specified below:

```
14 -> 7    16 -> 7    16 -> 9    18 -> 7    18 -> 9
18 -> 11   20 -> 7    20 -> 9    20 -> 11   20 -> 13
22 -> 7    22 -> 9    22 -> 11   22 -> 13   22 -> 15
24 -> 7    24 -> 9    24 -> 11   24 -> 13   24 -> 15
24 -> 17   26 -> 7    26 -> 9    26 -> 11   26 -> 13
26 -> 15   26 -> 17   26 -> 19   28 -> 7    28 -> 9
28 -> 11   28 -> 13   28 -> 15   28 -> 17   28 -> 19
28 -> 21   30 -> 7    30 -> 9    30 -> 11   30 -> 13
30 -> 15   30 -> 17   30 -> 19   30 -> 21   30 -> 23
```

A.5.3. QWEN 3 1.7B (28 TOTAL LAYERS)

A total of 21 connections were used, as specified below:

```
12 -> 4    15 -> 4    15 -> 7    18 -> 4    18 -> 7
18 -> 10   21 -> 4    21 -> 7    21 -> 10   21 -> 13
24 -> 4    24 -> 7    24 -> 10   24 -> 13   24 -> 16
27 -> 4    27 -> 7    27 -> 10   27 -> 13   27 -> 16
27 -> 19
```

A.6. LoRA Setting

For both 1B and 8B models, we use $r = 120$ for TurboConn and $r = 140$ for the original model. In this section, we show that with our setup, the number of trainable parameters available to TurboConn is less than or equal to that of the standard Transformer baseline.

In both setups, LoRA was applied to several primary parameters within each Transformer block.

There are the following key dimensions:

- d_{hidden}: The hidden size of the model.
- d_{kv}: The dimension of the key/value heads.
- d_{inter}: The intermediate size of the MLP blocks.

Table 6. Dimensions of the layers adapted with LoRA.

Module	d_{in}	d_{out}
Attention Block		
q_proj	d_{hidden}	d_{hidden}
k_proj	d_{hidden}	d_{kv}
v_proj	d_{hidden}	d_{kv}
o_proj	d_{hidden}	d_{hidden}
MLP Block		
gate_proj	d_{hidden}	d_{inter}
up_proj	d_{hidden}	d_{inter}
down_proj	d_{inter}	d_{hidden}

It is noted that all of these targeted modules in the base architecture are configured without bias terms.

The matrices for those parameters are unbiased. Given a LoRA rank r, the total number of trainable parameters per Transformer block (P_{block}) is calculated as follows:

$$P_{block} = \underbrace{4rd_{hidden}}_{\text{for q\_proj, o\_proj}} + \underbrace{2r(d_{hidden} + d_{kv})}_{\text{for k\_proj, v\_proj}} + \underbrace{3r(d_{hidden} + d_{inter})}_{\text{for gate, up, down\_proj}}$$

The total number of trainable parameters is then derived from this block-level calculation. For our method, we add the parameters from the N_{conn} downward connections. Each connection is a LoRA-adapted linear layer ($d_{hidden} \to d_{hidden}$) with an added bias. The parameter calculation is:

$$P_{connection} = \underbrace{2rd_{hidden}}_{\text{LoRA matrices}} + \underbrace{r}_{\text{bias}} + \underbrace{d_{hidden}}_{\text{bias}}$$

A.6.1. LLAMA 3.2 1B

For the Llama 3.2 1B model, the key dimensions are:

- $d_{hidden} = 2048$
- $d_{kv} = 512$
- $d_{inter} = 8192$

The number of trainable LoRA parameters per Transformer block (P_{block}), given a rank r, is calculated as:

$$\begin{aligned} P_{block} &= 4r(2048) + 2r(2048 + 512) + 3r(2048 + 8192) \\ &= 44032r \end{aligned}$$

Baseline Model ($r = 140$):

$$\begin{aligned} P_{\text{total}} &= 16 \times (44032 \times 140) \\ &= \mathbf{98{,}631{,}680} \end{aligned}$$

Our Method ($r = 120$):

$$\begin{aligned} P_{\text{conn}} &= N_{\text{conn}} \times (2rd_{\text{hidden}} + r + d_{\text{hidden}}) \\ &= 15 \times (2 \cdot 120 \cdot 2048 + 120 + 2048) \\ &= 7,405,320 \end{aligned}$$

The total number of trainable parameters is therefore:

$$\begin{aligned} P_{\text{ours}} &= (16 \times P_{\text{block}}) + P_{\text{conn}} \\ &= (16 \times 44032 \times 120) + 7,405,320 \\ &= \mathbf{91{,}946{,}760} \end{aligned}$$

A.6.2. LLAMA 3.1 8B

For the Llama 3.1 8B model, the key dimensions are:

- $d_{hidden} = 4096$
- $d_{kv} = 1024$
- $d_{inter} = 14336$

The number of trainable LoRA parameters per Transformer block (P_{block}), given a rank r, is calculated as:

$$\begin{aligned} P_{block} &= 4r(4096) + 2r(4096 + 1024) + 3r(4096 + 14336) \\ &= 81920r \end{aligned}$$

Baseline Model ($r = 140$):

$$\begin{aligned} P_{\text{total}} &= 32 \times (81920 \times 140) \\ &= \mathbf{367{,}001{,}600} \end{aligned}$$

Our Method ($r = 120$):

$$\begin{aligned} P_{\text{conn}} &= N_{\text{conn}} \times (2rd_{\text{hidden}} + r + d_{\text{hidden}}) \\ &= 45 \times (2 \cdot 120 \cdot 4096 + 120 + 4096) \\ &= 44,426,520 \end{aligned}$$

The total number of trainable parameters is therefore:

$$\begin{aligned} P_{\text{ours}} &= (32 \times P_{\text{block}}) + P_{\text{conn}} \\ &= (32 \times 81920 \times 120) + 44,426,520 \\ &= \mathbf{358{,}999{,}320} \end{aligned}$$

A.6.3. QWEN 3 1.7B

For the Qwen 3 1.7B model, the key dimensions are:

- $d_{hidden} = 2048$
- $d_{kv} = 1024$
- $d_{inter} = 6144$

The number of trainable LoRA parameters per Transformer block (P_{block}), given a rank r, is calculated as:

$$\begin{aligned} P_{block} &= 4r(2048) + 2r(2048 + 1024) + 3r(2048 + 6144) \\ &= 38912r \end{aligned}$$

Baseline Model ($r = 140$):

$$\begin{aligned} P_{\text{total}} &= 28 \times (38912 \times 140) \\ &= \mathbf{152{,}535{,}040} \end{aligned}$$

Our Method ($r = 120$):

$$\begin{aligned} P_{\text{conn}} &= N_{\text{conn}} \times (2rd_{\text{hidden}} + r + d_{\text{hidden}}) \\ &= 21 \times (2 \cdot 120 \cdot 2048 + 120 + 2048) \\ &= 10,367,448 \end{aligned}$$

The total number of trainable parameters is therefore:

$$\begin{aligned} P_{\text{ours}} &= (28 \times P_{\text{block}}) + P_{\text{conn}} \\ &= (28 \times 38912 \times 120) + 10,367,448 \\ &= \mathbf{141{,}111{,}768} \end{aligned}$$

A.7. More Training Details for TurboConn + CoT Experiments

Since the original NuminaMath-CoT dataset contains a test set of only 100 problems, we perform a custom re-split of the combined training and test data to ensure more statistically stable evaluations. We randomly partitioned the dataset into $384,000$ training, $1,000$ validation, and $1,000$ testing examples. The fixed CoT was generated by a Llama 3.2-1B model fine-tuned for 3 epochs on the original training sets, using a learning rate of 6.4×10^{-4}.

For GSM8K, we maintain the same experimental setup as for NuminaMath-CoT. We use the augmented GSM8K dataset described in Deng et al. (2023). However, unlike NuminaMath-CoT, GSM8K exhibits more significant overfitting to the training data; therefore, we use only 1/6 of the data to train the CoT generation model for a single epoch.

The training setup for the final prediction models remains identical to all previous experiments: we set the learning rate for Llama 3.2-1B models to 1.92×10^{-5}, use a batch size of 64, and train for 3 epochs using all training data.

In this particular experiment, we utilized a temperature of 0.01 for evaluation to sharpen the probability distribution. This encourages the model to be more 'certain' in its predictions, often yielding higher expected accuracy by amplifying the likelihood of the dominant choice.