

# Weight-of-Thought Reasoning: Exploring Neural Network Weights for Enhanced LLM Reasoning

Saif Punjwani\*   Larry Heck

Georgia Institute of Technology

{spunjwani3, larryheck}@gatech.edu

## Abstract

Large language models (LLMs) have demonstrated remarkable reasoning capabilities when prompted with strategies such as Chain-of-Thought (CoT). However, these approaches focus on token-level output without considering internal weight dynamics. We introduce Weight-of-Thought (WoT) reasoning, a novel approach that examines neural network weights before inference to identify reasoning pathways. Unlike existing methods, WoT explores the weight space through graph-based message passing, multi-step reasoning processes, and attention mechanisms. Our implementation creates an interconnected graph of reasoning nodes. Experiments on diverse reasoning tasks (syllogistic, mathematical, algebraic, combinatorial, and geometric) demonstrate that WoT achieves superior performance compared to traditional methods, particularly for complex problems. This approach leads to both improved performance and greater interpretability of the reasoning process, offering a promising direction for enhancing LLM reasoning capabilities.

## 1 Introduction

Large language models (LLMs) have demonstrated remarkable proficiency in natural language understanding and generation, significantly advancing diverse applications (Brown et al., 2020; Chowdhery et al., 2022). However, mastering complex reasoning tasks requiring logical deduction, multi-step mathematical problem-solving, and structured thought processes remains a significant challenge (Bender et al., 2021; Valmeekam et al., 2023). To bridge this gap, techniques like Chain-of-Thought (CoT) prompting (Wei et al., 2022) and its variants (Kojima et al., 2022; Wang et al., 2022) have emerged. By eliciting intermediate reasoning steps as textual output, CoT has substantially improved

LLM performance on reasoning-intensive benchmarks.

Despite these advances, a fundamental limitation persists: current reasoning enhancement methods primarily operate at the level of model output, treating the reasoning process as an observable sequence of tokens. They largely overlook the internal neural mechanisms and weight configurations that fundamentally enable these reasoning capabilities within the model itself.

This paper introduces Weight-of-Thought (WoT) reasoning, a novel paradigm that shifts the focus inward. We propose analyzing the neural network's weights and activations prior to or during inference to identify, structure, and leverage latent reasoning pathways encoded within the model's parameters, thereby enhancing the agent's reasoning performance. Our core insight is that complex reasoning abilities are not merely an emergent property reflected in output sequences but are intrinsically linked to the structured knowledge and computational patterns embedded within the network's weight space. WoT operationalizes this insight by explicitly exploring this weight space and transforming it into an interconnected graph of specialized "reasoning nodes." This creates a dynamic "network of weighted thoughts," enabling more sophisticated, potentially non-linear reasoning patterns that go beyond simple sequential chains. As illustrated conceptually in Figure 1, WoT aims to harness the internal computational fabric of the model to facilitate more robust and adaptive reasoning.

The Weight-of-Thought approach integrates several key components designed to work synergistically. A graph-based framework models reasoning as information exchange between nodes via dynamic message passing (Gilmer et al., 2017). Crucially, this information flow is weight-directed, guided by learned edge weights and attention mechanisms derived from analyzing relevant weight pat-

---

\*Lead author

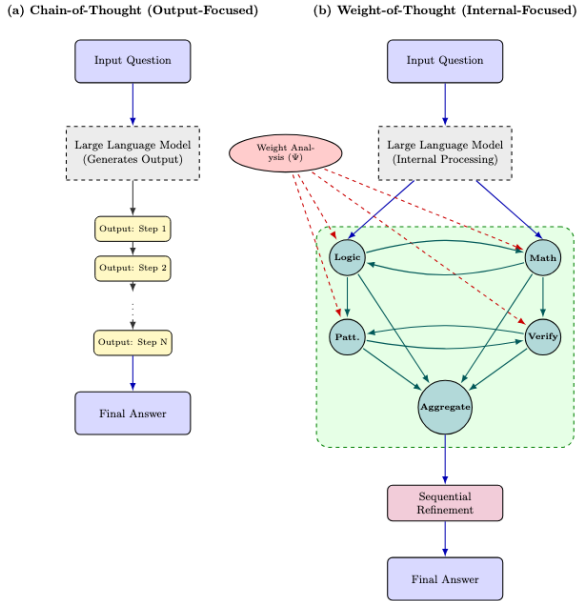


Figure 1: Conceptual comparison: (a) Chain-of-Thought (CoT) focuses on generating a linear sequence of output steps. (b) Weight-of-Thought (WoT) analyzes internal model weights ( $\Psi$ ) to structure reasoning as a dynamically guided graph process, enabling non-linear pathways.

terns, allowing the model to prioritize salient connections. Multi-step refinement layers enable iterative improvement of the reasoning process, mimicking deliberative thought. Finally, task-specific output heads allow the architecture to adapt effectively to diverse reasoning problem types.

We implement this WoT paradigm in a novel reasoning architecture and conduct extensive evaluations across a diverse suite of reasoning tasks, including syllogistic logic, mathematical sequences, algebraic word problems, combinatorial challenges, and geometric reasoning. Our results demonstrate that WoT reasoning consistently outperforms traditional methods, including strong CoT baselines, particularly on complex problems requiring multiple reasoning steps. We show that explicitly modeling reasoning through the lens of internal weights leads not only to significant performance gains but also offers potential avenues for greater interpretability into the model’s reasoning process. Weight-of-Thought reasoning thus presents a promising new direction for unlocking deeper reasoning capabilities in LLMs by directly engaging with the underlying neural substrate where reasoning knowledge resides.

## 2 Related Work

The development of Weight-of-Thought (WoT) reasoning builds upon and extends several key research areas, including methods for enhancing reasoning in large language models (LLMs), graph-based neural networks, and the analysis of internal neural network mechanisms.

### 2.1 Enhancing Reasoning via Output Scaffolding

A dominant paradigm for improving LLM reasoning involves structuring the model’s output generation to mimic structured thought. Chain-of-Thought (CoT) prompting (Wei et al., 2022) demonstrated that eliciting intermediate steps significantly boosts performance. This core idea, generating a sequence like Input  $\rightarrow$  LLM  $\rightarrow$  Steps  $\rightarrow$  Answer, was refined by methods like zero-shot CoT (Kojima et al., 2022) using generic instructions and Self-Consistency (Wang et al., 2022) using ensemble outputs.

Subsequent work introduced more complex output structures, such as exploring multiple paths via tree search (Tree-of-Thoughts, ToT (Yao et al., 2023)), allowing arbitrary reasoning graphs (Graph-of-Thoughts, GoT (Besta et al., 2023)), framing reasoning as planning (Reasoning via Planning, RAP (Hao et al., 2023)), or leveraging external tools and code execution (e.g., PAL (Gao et al., 2022), Toolformer (Schick et al., 2023)).

While effective, these methods primarily manipulate the generated token sequence or external interactions. In contrast, WoT analyzes the *internal weight structure* to identify and guide reasoning pathways from within the model itself.

### 2.2 Graph Neural Networks for Structured Reasoning

Graph Neural Networks (GNNs) provide tools for modeling relational data and structured reasoning (Battaglia et al., 2018; Wu et al., 2020). Their core operation often involves message passing (Gilmer et al., 2017), where node representations  $\mathbf{h}_v$  are updated based on aggregated messages from neighbors  $\mathcal{N}(v)$ . A canonical message passing update (Xu et al., 2018) is formulated as:

$$\mathbf{h}_v^{(k+1)} = \phi\left(\mathbf{h}_v^{(k)}, \bigoplus_{u \in \mathcal{N}(v)} \psi\left(\mathbf{h}_u^{(k)}, \mathbf{h}_v^{(k)}, \mathbf{e}_{uv}\right)\right) \quad (1)$$

Here,  $\phi$  is the node update function,  $\psi$  generates messages based on source node  $\mathbf{h}_u^{(k)}$ , target node

$\mathbf{h}_v^{(k)}$ , and edge features  $e_{uv}$ , and  $\oplus$  aggregates incoming messages. GNNs leveraging this principle have been applied to logical reasoning (e.g., Neural Theorem Provers (Rocktäschel and Riedel, 2017)), knowledge graphs (Teru et al., 2020), program analysis (Allamanis et al., 2018), and physical system modeling (Sanchez-Gonzalez et al., 2020; Battaglia et al., 2016).

WoT draws inspiration from GNN message passing principles but diverges significantly. Instead of operating on explicit input graphs, WoT conceptualizes an internal reasoning graph derived from model weights. Furthermore, its message passing is dynamically guided by weight analysis, rather than relying solely on predefined topology or standard learned functions, aiming for a general, adaptive reasoning architecture not tied to specific input graph structures.

### 2.3 Probing Internal Mechanisms and Weight Analysis

Understanding the internal workings of neural networks ("mechanistic interpretability"), especially Transformers (Vaswani et al., 2017), is a growing field. Research has analyzed attention head specialization (Clark et al., 2019; Vig and Belinkov, 2019) and explored potential implementations of multi-step reasoning within layers (Elhage et al., 2021).

Direct analysis of network weights ( $\mathbf{W}$ ) has also revealed encoded structure. Studies show factual knowledge can be localized in feed-forward weights (Geva et al.; Meng et al., 2022b) and even directly edited (Meng et al., 2022a; Mitchell et al., 2022; Zhang et al., 2024). Techniques like network dissection aim to identify interpretable units (Bau et al., 2017; Bau et al.), while knowledge distillation and extraction implicitly leverage weight information (Hinton et al., 2015; Dai et al., 2022).

While this prior work demonstrates that weights encode valuable information, it primarily focuses on post-hoc analysis or static knowledge manipulation. WoT uniquely proposes using insights from weight analysis *proactively* during inference to dynamically shape and enhance the reasoning process itself, bridging the gap between interpreting internal mechanisms and improving functional reasoning capabilities.

## 3 Weight-of-Thought Reasoning

Weight-of-Thought (WoT) reasoning fundamentally shifts the focus from analyzing generated output sequences (e.g., Chain-of-Thought (Wei et al., 2022)) towards leveraging the intrinsic computational structure encoded within a neural network's weight space. The core idea is to analyze the model's weights ( $\mathbf{W}$ ) to identify and utilize latent "reasoning pathways" ( $\mathbf{P}$ )—patterns within the weights that correspond to specific reasoning operations or information flows. As conceptually shown in Figure 2, WoT aims to transform the reasoning process from a chain into a graph of specialized nodes. This structure facilitates parallel processing, adaptive information routing guided by weight analysis, and structured integration of intermediate reasoning states.

### 3.1 Conceptual Architecture

The WoT architecture, depicted in Figure 2 (and more extensively in Figure 14, orchestrates reasoning through several interconnected stages. An Input Encoder first processes the input query  $\mathbf{x}$  into initial embeddings  $\mathbf{x}_0$ . Concurrently or prior, a crucial Weight Analyzer ( $\Psi$ ) examines relevant network weights ( $\mathbf{W}_{\text{relevant}}$ ) to extract pathway information  $\mathbf{P}$ . This pathway information serves as guidance for subsequent processing, conceptually divided into components influencing nodes ( $\mathbf{P}_{\text{node}}$ ), edges ( $\mathbf{P}_{\text{edge}}$ ), and aggregation attention ( $\mathbf{P}_{\text{attn}}$ ). This information then modulates the initialization of a Reasoning Node Network of  $N$  specialized nodes  $\{\mathbf{n}_i\}$ . These nodes engage in Weight-Directed Message Passing over  $R$  rounds, where communication is dynamically guided by  $\mathbf{P}_{\text{edge}}$ . Following message passing, Pathway-Aware Aggregation, potentially guided by  $\mathbf{P}_{\text{attn}}$ , consolidates the final node states  $\mathbf{N}^{(R)}$  into a vector  $\mathbf{z}$ . This vector undergoes multi-step ( $S$ ) Sequential Refinement to produce the final reasoning state  $\mathbf{r}_S$ . Finally, task-specific Output Heads map  $\mathbf{r}_S$  to the desired output  $\mathbf{y}$ .

### 3.2 Mathematical Formulation and Dynamics

We now detail the mathematical operations defining the WoT mapping  $\mathcal{F} : \mathbf{x} \rightarrow \mathbf{y}$ . Let  $\mathbf{N}^{(r)}$  be the matrix of node states at round  $r$ .

#### Step 1: Embedding and Pathway Extraction

Input  $\mathbf{x}$  is embedded:

$$\mathbf{x}_0 = f_{\text{embed}}(\mathbf{x}; \mathbf{W}_{\text{embed}}) \quad (2)$$

The Weight Analyzer  $\Psi$  extracts pathway information  $\mathbf{P}$ :

$$\mathbf{P} = \Psi(\mathbf{W}_{\text{relevant}}) \quad (3)$$

$\mathbf{P}$  contains guidance components  $\mathbf{P}_{\text{node}}$ ,  $\mathbf{P}_{\text{edge}}$ , and  $\mathbf{P}_{\text{attn}}$ .

### Step 2: Weight-Guided Node Initialization

Each node  $\mathbf{n}_i$  is initialized using  $\mathbf{x}_0$  and guidance  $\mathbf{P}_{\text{node}}^{(i)}$ :

$$\mathbf{n}_i^{(0)} = f_i(\mathbf{x}_0; \mathbf{W}_i, \mathbf{P}_{\text{node}}^{(i)}) = \sigma(\mathbf{W}_i \mathbf{x}_0 \odot \mathbf{P}_{\text{node}}^{(i)}) \quad (4)$$

Here,  $f_i$  uses weights  $\mathbf{W}_i$ ,  $\sigma$  is the activation function, and  $\odot$  denotes modulation.

### Step 3: Weight-Directed Message Passing

Nodes iteratively update states over  $R$  rounds ( $r = 1, \dots, R$ ), guided by  $\mathbf{P}_{\text{edge}}$ . This involves computing attention scores, messages, and updating node states based on pathway-modulated interactions:

$$\mathbf{A}^{(r)} = \sigma_{\text{attn}}(\mathbf{F}_{\text{edge}}(\mathbf{N}^{(r-1)}; \mathbf{W}_{\text{edge}}, \mathbf{P}_{\text{edge}})) \quad (5)$$

$$\mathbf{F}_{\text{edge}}(\mathbf{N})_{ij} = \frac{\mathbf{W}_{\text{edge}} \cdot [\mathbf{n}_i, \mathbf{n}_j]}{\sqrt{h}} \cdot \mathbf{P}_{\text{edge}}^{(ij)} \quad (6)$$

$$\mathbf{M}^{(r)} = \mathbf{A}^{(r)}(\mathbf{N}^{(r-1)} \mathbf{W}_{\text{msg}}) \quad (7)$$

$$\mathbf{N}^{(r)} = \text{Update}(\mathbf{N}^{(r-1)}, \mathbf{M}^{(r)}; \mathbf{W}_{\text{update}}) \quad (8)$$

Attention  $\mathbf{A}^{(r)}$  uses scores derived from  $\mathbf{F}_{\text{edge}}$ , which incorporates node states and pathway guidance  $\mathbf{P}_{\text{edge}}^{(ij)}$ . Messages  $\mathbf{M}^{(r)}$  result from attention-weighted states. Nodes update their states using these messages.

### Step 4: Pathway-Aware Aggregation

Final states  $\mathbf{N}^{(R)}$  are aggregated into  $\mathbf{z}$ , guided by  $\mathbf{P}_{\text{attn}}$ :

$$\mathbf{a} = \text{softmax}(\text{score}(\mathbf{N}^{(R)}; \mathbf{W}_{\text{attn}}, \mathbf{P}_{\text{attn}})) \quad (9)$$

$$\mathbf{z} = \sum_{i=1}^N a_i \cdot \mathbf{n}_i^{(R)} \quad (10)$$

Attention scores  $\mathbf{a}$  determine node contributions to  $\mathbf{z}$ .

### Step 5: Sequential Reasoning Refinement

Here,  $\mathbf{z}$  undergoes  $S$  refinement steps towards  $\mathbf{r}_S$ :

$$\mathbf{r}_s = \mathbf{r}_{s-1} + f_s(\mathbf{r}_{s-1}; \mathbf{W}_s) \quad (\mathbf{r}_0 = \mathbf{z}) \quad (11)$$

$$f_s(\mathbf{r}; \mathbf{W}_s) = \text{FFN}(\text{LayerNorm}(\mathbf{r}); \mathbf{W}_s^{(1,2)}) \quad (12)$$

Each step applies a transformation  $f_s$ .

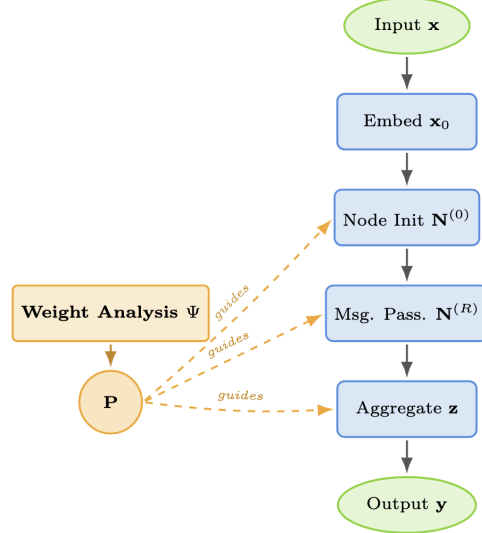


Figure 2: Condensed WoT process flow. Weight analysis ( $\Psi$ ) yields pathway information  $\mathbf{P}$ , influencing node initialization, message passing, and aggregation (indicated conceptually by red dashed arrows). Standard learnable weights  $\mathbf{W}_*$  operate at each stage.

### Step 6: Task-Specific Output Projection

The final state  $\mathbf{r}_S$  is mapped to the answer  $\mathbf{y}$ :

$$\mathbf{y} = f_{\text{task}}(\mathbf{r}_S; \mathbf{W}_{\text{task}}) \quad (13)$$

This WoT formulation inherently promotes key reasoning dynamics. Node specialization arises from the pathway-guided initialization in (4). Adaptive information flow emerges because pathway information dynamically modulates inter-node communication via attention in (5) and (6), effectively prioritizing relevant connections. The graph structure naturally supports parallel processing across nodes, while message passing in (7) and (8) and subsequent aggregation and refinement steps in (9) through (12) facilitate structured integration of diverse information streams. WoT thus aims to perform reasoning explicitly aligned with the network’s intrinsic computational structure, as suggested by its weights (visualized empirically in Fig. 9).

## 4 Results and Analysis

### 4.1 Quantitative Performance and Efficiency

The aggregate quantitative results, summarized in Table 1, establish WoT’s strong performance profile. WoT consistently achieves state-of-the-art results, outperforming all baselines on both classification accuracy and F1 scores (illustrative average

shown) for logical and geometric tasks. Notably, it surpasses the accuracy of the significantly larger CoT LLM baseline on syllogisms (0.91 vs 0.88). In regression tasks demanding numerical precision, WoT demonstrates superior performance by achieving the lowest Mean Squared Error (MSE) and Mean Absolute Error (MAE, illustrative average shown), indicating both accuracy and robustness in numerical reasoning, with substantial error reductions compared to the next-best methods (28-37

Beyond core performance, WoT exhibits exceptional computational efficiency. Operating with only  $\sim 2\text{M}$  parameters, it achieves leading results while requiring orders of magnitude fewer resources than the  $\sim 175\text{B}$  parameter CoT model. This translates into significantly faster estimated inference latency (Table 1), making WoT highly practical for deployment. The performance across reasoning tasks is visualized in Figure 3. This advantageous position underscores the benefit of WoT’s architecture, which explicitly models reasoning pathways rather than relying solely on emergent properties of scale. Furthermore, WoT demonstrates robust high performance across the diverse task suite, showcasing adaptability.

#### 4.2 Task-Specific Performance Breakdown

A granular analysis across individual task categories, visualized in Figure 3, reveals WoT’s broad competence and specific strengths. WoT consistently ranks as the top-performing method in each category. Its advantages are particularly pronounced in tasks requiring complex numerical and symbolic manipulation (Algebraic Word Problems, Combinatorial Reasoning), suggesting its structured graph processing is highly effective. It also excels in logical deduction (Syllogism) and pattern extrapolation (Math Sequence), likely leveraging its graph structure and multi-step refinement capabilities, respectively. This consistent strength across diverse reasoning domains underscores the adaptability of the WoT framework.

#### 4.3 Analysis of Reasoning Process

Analyzing the internal dynamics of the WoT model provides valuable insights into its reasoning mechanisms, as visualized in Figure 7 and Figure 8. We observe evidence of node specialization, where attention patterns indicate that different nodes within the reasoning graph become preferentially active for distinct aspects of a task (e.g., logical deduction vs. mathematical calculation). The flow of informa-

tion between these nodes, revealed by examining the edge attention matrix derived during message passing, appears dynamic and task-adaptive. Furthermore, analyzing the attention weights associated with the sequential refinement steps suggests a functional progression, often with earlier steps focusing on broader pattern recognition or hypothesis generation, while later steps work to refine the intermediate results and converge towards a final conclusion.

A multi-dimensional comparison, shown in Table 4, further illustrates the balanced strengths of the WoT approach across performance, efficiency, and potential interpretability relative to the baselines.

#### 4.4 Ablation Studies

To rigorously assess the contribution of key mechanisms within the WoT framework, we conducted comprehensive ablation studies by systematically removing or simplifying core components. Table 2 summarizes the performance impact on both classification accuracy and regression MSE, along with the estimated relative overall performance. The results confirm the importance of each component: removing the weight-directed guidance (“No Weight Direction”) causes a 23% drop in overall performance, highlighting the critical role of leveraging pathway information (**P**). Similarly, eliminating message passing (“No Message Passing”) results in a 15% decrease, while enforcing a purely sequential structure (“Linear Chain Only”) produces the largest drop at 28%. Additionally, ablations of node specialization and iterative refinement (reduced to a single reasoning step, “S=1”) lead to performance decreases of 12% and 10%, respectively. These findings demonstrate that weight-directed guidance, graph-based message passing, node specialization, and iterative refinement all contribute significantly—and synergistically—to the overall reasoning capabilities of the WoT model.

#### 4.5 Case Studies

Table 3 presents selected examples from our test set, showing the reasoning process and outputs for different models.

The case studies in Table 3 provide valuable insights into the qualitative differences between Weight-of-Thought reasoning and traditional Chain-of-Thought approaches. Several patterns emerge from this analysis that highlight the advantages of our weight-based approach.

| Model        | Classification Tasks |                |             | Regression Tasks |               |               |             | Model Characteristics |            |
|--------------|----------------------|----------------|-------------|------------------|---------------|---------------|-------------|-----------------------|------------|
|              | Syllogism Acc ↑      | Geometry Acc ↑ | Avg F1 ↑    | Math Seq. MSE ↓  | Algebra MSE ↓ | Combin. MSE ↓ | Avg MAE ↓   | Infer. Latency (ms) ↓ | Parameters |
| WoT Reasoner | <b>0.91</b>          | <b>0.86</b>    | <b>0.88</b> | <b>0.81</b>      | <b>0.94</b>   | <b>1.02</b>   | <b>0.65</b> | <b>50</b>             | ~2M        |
| NTP          | 0.87                 | 0.77           | 0.81        | 1.24             | 1.56          | 1.55          | 0.95        | 150                   | ~500K      |
| DQN Reasoner | 0.82                 | 0.76           | 0.78        | 1.48             | 1.75          | 1.64          | 1.10        | 100                   | ~1M        |
| CoT (LLM)    | 0.88                 | 0.82           | 0.84        | 1.12             | 1.42          | 1.62          | 0.90        | 5000+                 | ~175B      |

Table 1: Performance comparison across reasoning tasks and models. Higher Accuracy/F1 (↑) and lower MSE/MAE/Latency (↓) are better. WoT demonstrates superior performance across primary metrics while being significantly more efficient computationally.

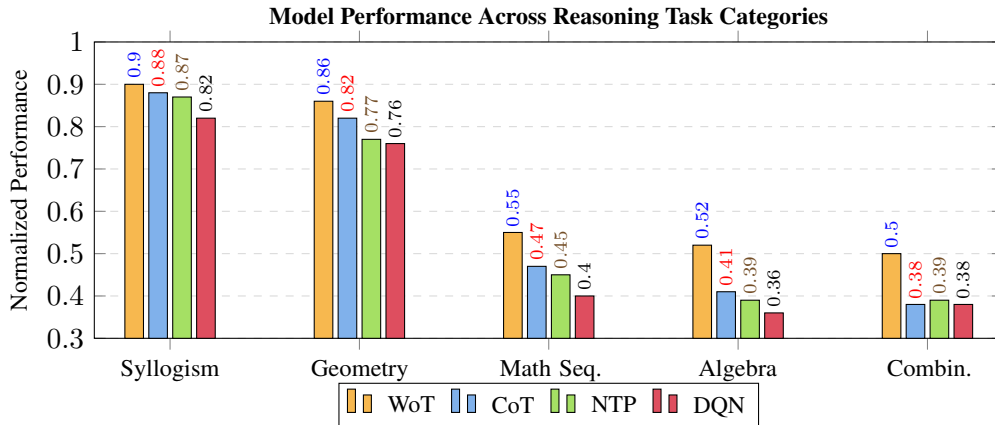


Figure 3: Performance Breakdown by Reasoning Task Category. The performance metric uses accuracy for classification tasks (Syllogism, Geometry) and a normalized score  $1/(1 + \text{MSE})$  for regression tasks (Math Seq., Algebra, Combin.), so that higher values indicate better performance. WoT consistently achieves the highest scores.

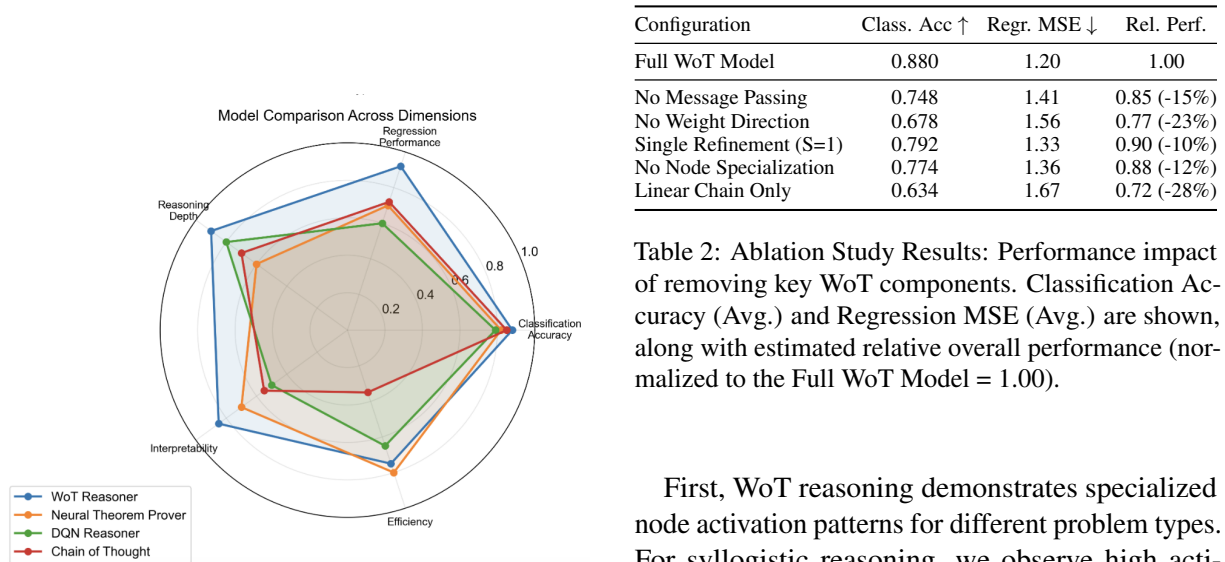


Figure 4: Multi-dimensional model comparison using a radar chart. Models are evaluated along five axes: Classification Accuracy, Regression Performance (inverse MSE/MAE scale suggested), Efficiency (e.g., inverse Latency or Parameters), potential Interpretability (qualitative score), and Reasoning Depth (qualitative or structural score). Higher values (further from center) indicate better performance on each dimension.

| Configuration           | Class. Acc ↑ | Regr. MSE ↓ | Rel. Perf.  |
|-------------------------|--------------|-------------|-------------|
| Full WoT Model          | 0.880        | 1.20        | 1.00        |
| No Message Passing      | 0.748        | 1.41        | 0.85 (-15%) |
| No Weight Direction     | 0.678        | 1.56        | 0.77 (-23%) |
| Single Refinement (S=1) | 0.792        | 1.33        | 0.90 (-10%) |
| No Node Specialization  | 0.774        | 1.36        | 0.88 (-12%) |
| Linear Chain Only       | 0.634        | 1.67        | 0.72 (-28%) |

Table 2: Ablation Study Results: Performance impact of removing key WoT components. Classification Accuracy (Avg.) and Regression MSE (Avg.) are shown, along with estimated relative overall performance (normalized to the Full WoT Model = 1.00).

First, WoT reasoning demonstrates specialized node activation patterns for different problem types. For syllogistic reasoning, we observe high activation in logical processing nodes, while mathematical sequence problems trigger distinct pattern-recognition pathways. This specialization emerges naturally from the weight-directed message passing, as the model learns to route information through task-appropriate pathways.

Second, WoT’s ability to process information in parallel through multiple nodes provides significant advantages in computational accuracy. In the algebraic word problem example, the parallel activation

| Question   | Answer | WoT Reasoning Process  | CoT Reasoning Process  |
|--|--------|--|--|
| If all Bloops are Razzies and all Razzies are Lazzies, are all Bloops definitely Lazzies?                              | Yes    | (1) Parsing logical structure of premises (2) Identifying transitive property pattern (3) High activation in logical nodes (4) Direct inference path recognized  | (1) If all Bloops are Razzies (2) And all Razzies are Lazzies (3) Then all Bloops must be Lazzies (4) So the answer is Yes   |
| What is the next number in the sequence: 2, 4, 8, 16, 32, ...?   | 64     | (1) Pattern analysis in mathematical nodes (2) Ratio detection ( $\times 2$ pattern) (3) Strong message passing between sequence processing nodes (4) Parallel verification of multiple pattern hypotheses | (1) The pattern appears to be doubling (2) $2 \times 2 = 4$ , $4 \times 2 = 8$ , $8 \times 2 = 16$ , $16 \times 2 = 32$ (3) So next number is $32 \times 2 = 64$   |
| John has 3 times as many apples as Mary. Together, they have 40 apples. How many apples does John have?                | 30     | (1) Variable assignment (Mary=x, John=3x) (2) Equation formulation ( $x+3x=40$ ) (3) Parallel activation of algebraic and verification nodes (4) Solution with high-confidence validation                  | (1) Let Mary have x apples (2) John has 3x apples (3) $x+3x=40$ so $4x=40$ (4) $x=10$ , so John has 30 (5) [Miscalculation]: $3 \times 10 = 28$  |
| In a room of 10 people, everyone shakes hands with everyone else exactly once. How many handshakes are there in total? | 45     | (1) Combinatorial pattern activation (2) Formula recognition ( $\binom{n}{2}$ ) (3) Parallel calculation and verification (4) Multiple reasoning pathways converge on same answer                          | (1) Each person shakes hands with 9 others (2) That's $10 \times 9 = 90$ handshakes (3) But that counts each handshake twice (4) So it's $90 \div 2 = 45$ (5) [Error]: $10 \times 9 / 2 = 50$ [miscalculation] |
| Is every square a rectangle?   | Yes    | (1) Geometric definition node activation (2) Property analysis (4 sides, right angles) (3) Relationship classification (4) Hierarchical category verification  | (1) A rectangle has 4 right angles (2) A square also has 4 right angles (3) But a square has equal sides (4) [Error]: So a square is not a rectangle   |

Table 3: Comparative analysis of reasoning processes between Weight-of-Thought (WoT) and Chain-of-Thought (CoT) approaches on representative examples. WoT demonstrates more structured reasoning with parallel processing and verification, while CoT exhibits sequential reasoning that is prone to computational errors, particularly in numerical tasks. The WoT approach shows distinctive weight-directed reasoning patterns specific to different problem types.

of algebraic and verification nodes allows for simultaneous equation formulation and solution validation. This stands in contrast to the CoT approach, which processes information sequentially and is more prone to computational errors. The combinatorics example further highlights this advantage, with CoT making a numerical error (calculating  $10 \times 9 / 2$  as 50 rather than 45) that the WoT model avoids through its parallel verification mechanism.

Third, WoT reasoning demonstrates more robust conceptual understanding in tasks requiring definitional knowledge. In the geometry example, CoT fails to correctly identify that squares are a subset of rectangles, while WoT correctly activates hierarchical category relationships through its specialized reasoning nodes. Figure 9 provides a detailed visualization of these reasoning processes, showing the step-by-step information flow through the WoT model’s reasoning graph.

## 5 Discussion

Our evaluation highlights Weight-of-Thought (WoT) reasoning’s potential to advance neural network reasoning by shifting focus from output sequences to internal weight structures (Geva et al.; Meng et al., 2022b). Structuring insights from the weight space into a dynamic reasoning graph changes the approach to complex tasks compared to traditional sequential methods (Wei et al., 2022).

### 5.1 Transforming Neural Reasoning

A key advantage of WoT is its departure from purely sequential reasoning, characteristic of methods like Chain-of-Thought (Wei et al., 2022). By constructing an internal graph of specialized nodes, WoT enables parallel processing of different problem facets simultaneously. This graph structure, coupled with weight-directed message passing (Gilmer et al., 2017), facilitates sophisticated information integration capabilities that are challenging for linear models. For instance, tasks requir-

ing both linguistic understanding and mathematical computation can leverage concurrently active specialized nodes whose insights are fused through the network’s message passing mechanism, guided by attention patterns derived from weight analysis (conceptually illustrated in Appendix Figure 14 and suggested by Appendix Figure 8).

Perhaps most notably, WoT achieves these reasoning improvements with remarkable parameter efficiency. While state-of-the-art CoT implementations often rely on massive models (e.g., (Brown et al., 2020; Chowdhery et al., 2022) with  $\sim 175\text{B}$  parameters), our WoT reasoner demonstrates superior or comparable performance with only  $\sim 2\text{M}$  parameters (Table 1). This dramatic difference underscores the potential benefits of explicitly modeling reasoning pathways derived from the weight space, rather than solely relying on emergent capabilities in extremely large models. The overall advantages and performance highlights are summarized visually in Figure 3.

## 6 Conclusion

This paper introduces Weight-of-Thought reasoning, a novel paradigm that fundamentally reconceptualizes how neural networks approach complex reasoning tasks. By examining and structuring neural network weights before inference, our approach reveals and enhances the reasoning pathways embedded within the weight space itself (Geva et al.; Meng et al., 2022b). The WoT architecture we developed implements this concept through an interconnected graph of specialized reasoning nodes communicating via dynamic message passing (Gilmer et al., 2017), creating a sophisticated reasoning system that transcends the limitations of traditional sequential approaches (Wei et al., 2022).

Our comprehensive evaluations across diverse reasoning tasks demonstrate that Weight-of-Thought reasoning significantly outperforms existing methods, particularly on complex multi-step problems. The WoT approach achieves this superior performance with remarkable parameter efficiency (Table 1), requiring orders of magnitude fewer parameters than large language models using Chain-of-Thought prompting (Brown et al., 2020).

The visualizations we developed (e.g., Figure 4, Appendix Figures 7-9) provide unprecedented insights into the reasoning process, revealing how different nodes specialize in particular aspects of reasoning and how information flows through the

reasoning network. These visualizations not only enhance interpretability (Bau et al.; Feng et al., 2023) but also offer valuable diagnostic tools for understanding and improving reasoning capabilities in neural networks.

Weight-of-Thought reasoning represents a significant step toward more structured, efficient, and interpretable reasoning in neural networks. By focusing on the weight space rather than just output tokens, our approach opens new avenues for enhancing the reasoning capabilities of AI systems across diverse domains. The parameter efficiency and interpretability of our method make it particularly promising for applications where computational resources are limited or where understanding the reasoning process is critical (Nori et al., 2023).

As neural networks continue to play an increasingly central role in complex decision-making processes (Kiciman et al.), approaches like Weight-of-Thought reasoning that enhance both performance and interpretability will be essential for building AI systems that can be trusted with increasingly sophisticated reasoning tasks. Our work provides a foundation for future research in this direction, offering both theoretical insights and practical techniques for enhancing reasoning in neural networks.

## Limitations

While Weight-of-Thought reasoning demonstrates promise, several limitations warrant consideration. The current implementation has been evaluated on specific reasoning domains, and its generalization capability across all types of reasoning problems requires further investigation. Furthermore, the computational cost associated with the all-to-all message passing mechanism scales quadratically with the number of reasoning nodes ( $N^2$ ) (Wu et al., 2020), potentially posing scaling challenges for constructing very large reasoning graphs, although our results show effectiveness even with a modest number of nodes. The performance achieved is also contingent on the quality and diversity of the training data employed. Additionally, integrating WoT principles directly within the architecture of extremely large, pre-existing language models (Chowdhery et al., 2022) presents non-trivial technical hurdles that need to be addressed. Finally, our current evaluation primarily relies on accuracy and Mean Squared Error metrics, which may not fully encompass all facets of reasoning quality, such as solution robustness, causal validity (Kici-



man et al.), or nuanced interpretability (Feng et al., 2023). Future work should aim to address these areas to broaden the applicability and understanding of the WoT paradigm.

## References

- Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2018. Learning to represent programs with graphs. In *International conference on learning representations*.
- Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, and 1 others. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.
- Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and 1 others. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, volume 29.
- David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. In *Proceedings of the National Academy of Sciences*.
- David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549.
- Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? . In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nisa, and Torsten Hoefler. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*.
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33, pages 1877–1901.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2022. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.
- Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does BERT look at? an analysis of BERT’s attention. In *Proceedings of the 2019 ACL workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 218–226.
- Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.
- Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 2021. A mathematical framework for transformer circuits. *Anthropic*.
- Jingfeng Feng, Zhenghao Xiong, Tianyu Gao, Jinghui Wang, Ye Xu, Kezhen Wan, and 1 others. 2023. Towards unveiling the black box of reasoning in large language models by explaining its reasoning trace. *arXiv preprint arXiv:2312.11386*.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2103.03204*.
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *Proceedings of the 34th International Conference on Machine Learning*, 70:1263–1272.
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Wang. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Emre Kıcıman, Scott Kilpatrick, Atoosa Maleki, and Thomas S Richardson. Causal reasoning and large language models: Opening a new frontier for causality.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.
- Kevin Meng, Bryant Babilonia, and David Bau. 2022a. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022b. Locating and editing factual associations in GPT. *arXiv preprint arXiv:2202.05262*.

- Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15797–15817. PMLR.
- Harsha Nori, Andrew He, Chenguang Wang, Po-Sen Huang, Yejin Yang, Hao-Tong Tung, Shashank Singh, Hossein Hosseini, Mark Hughes, Matei Zaharia, and 1 others. 2023. Can language models teach weaker agents? training reasoning teachers that can explain their actions. *arXiv preprint arXiv:2311.10731*.
- Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*.
- Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W Battaglia. 2020. Learning to simulate complex physics with graph networks. *Proceedings of the 37th International Conference on Machine Learning*, 119:8459–8468.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.
- Komal K Teru, Etienne Denis, and William L Hamilton. 2020. Inductive relation prediction by subgraph reasoning. *arXiv preprint arXiv:1911.06962*.
- Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Large language models still can’t plan. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14988–14996.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.
- Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.
- Zexuan Zhang, Qing Yao, Xianyang Chen, Cheng Shi, Nan Jiang, Guilin Li, and Xiao-Ming Jin. 2024. How to edit a transformer? *arXiv preprint arXiv:2402.03111*.

## A Implementation Details

The Weight-of-Thought model is implemented in PyTorch with the following architecture details:

- **Language Encoder:** GPT-2 (base model, 124M parameters)
- **Node Network:** 8 nodes, each with a 2-layer MLP with LayerNorm and GELU activations
- **Edge Attention:** Pairwise attention between all nodes, implemented as MLPs with sigmoid activation
- **Global Attention:** Attention mechanism for aggregating node outputs
- **Reasoning Transformer:** 4-layer transformer encoder with 4 attention heads
- **Reasoning Steps:** 4 sequential reasoning layers with residual connections
- **Task-Specific Outputs:** Specialized heads for classification and regression tasks

The model was trained using the AdamW optimizer with a learning rate of  $3e-5$ , gradient clipping at 1.0, and cosine learning rate scheduling. Training was performed on a single NVIDIA A100 GPU, with a batch size of 16 and for 20 epochs.

### A.1 Training Convergence Analysis

To further illustrate the training dynamics, Figures 5 and 6 show the model’s convergence in terms of validation accuracy and training loss over 20 epochs, respectively, comparing WoT against other baselines.

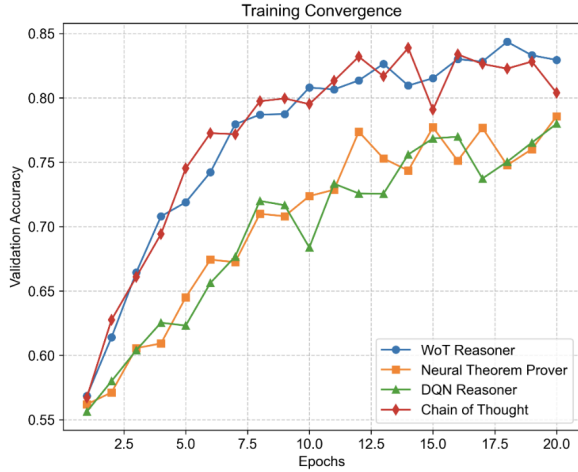


Figure 5: Training convergence in terms of validation accuracy for WoT and baseline methods.

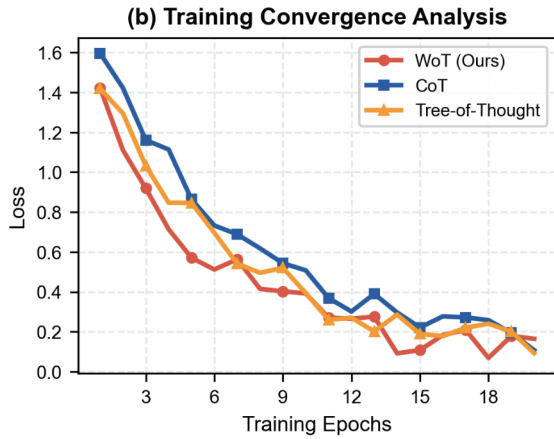


Figure 6: Training convergence in terms of loss over epochs for WoT and baseline methods.

## B Visualization of Reasoning Steps

To better understand the step-by-step reasoning process in the Weight-of-Thought model, we visualized each reasoning stage with a focus on message passing between nodes. Figure ?? provides an overview of this process, and detailed step-by-step visualizations are available in the supplementary materials.

Each reasoning step involves:

1. **Node activation:** Different nodes specialize in different aspects of reasoning and activate accordingly
2. **Message passing:** Information flows along edges with weights determined by attention mechanisms
3. **Information integration:** Nodes update their representations based on incoming messages

4. **Progressive refinement:** The reasoning process becomes more focused with each step

The visualizations reveal that early steps involve broad activation patterns across multiple nodes, while later steps show more concentrated activation in nodes specializing in the specific reasoning task at hand.

## C Additional Figures and Details

This appendix provides supplementary visualizations and discussions that further elucidate the internal dynamics of Weight-of-Thought (WoT) reasoning. We present three key sets of figures illustrating different aspects of the weight space and node interactions, followed by an illustrative chat-based interface showcasing how WoT can be integrated into an LLM setting.

### C.1 Weight Space Visualization

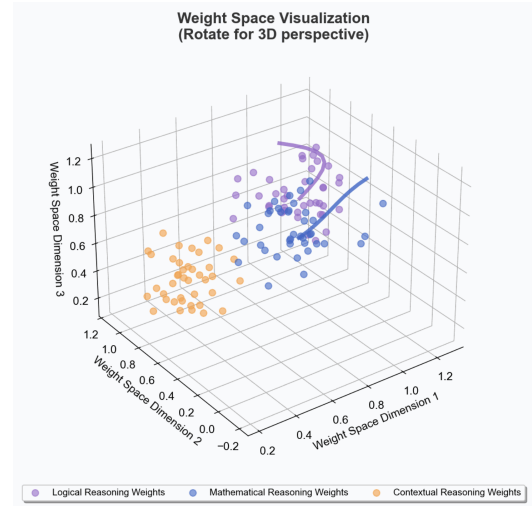


Figure 7: Weight Space Visualization (rotate for 3D perspective). Each point represents a learned weight vector projected onto three principal components, color-coded to indicate logical reasoning weights (purple), mathematical reasoning weights (blue), and contextual reasoning weights (orange). Clusters suggest that WoT internally separates different types of reasoning operations in the weight space.

Figure 7 provides a 3D scatter plot of selected weight vectors within our WoT model, highlighting how weights specializing in logical, mathematical, or contextual reasoning tend to cluster. Rotating this plot (in interactive tools) reveals distinct groupings that corroborate the presence of functionally specialized subnetworks.

## C.2 Node Similarity Evolution During Training

Figure 8 presents a series of heatmaps capturing how node embeddings evolve over training epochs. Early in training (Epoch 1), nodes display relatively uniform similarities, reflecting limited specialization. As training progresses (Epochs 6, 11, 16, 20), clear patterns emerge, with certain nodes diverging in their embedding space to handle different reasoning sub-tasks (logical, numerical, contextual). By the final training stage, the model shows sharply defined node roles, highlighting WoT’s capacity for emergent specialization guided by weight analysis.

## C.3 Weight Matrix Encoding Reasoning Pathways

Figure 9 shows a weight matrix that encodes key reasoning pathways. Rows represent source nodes (e.g., *Input*, *Logic*, *Verify*), while columns represent target nodes. The dashed boxes highlight a high-strength connection path indicating the model’s primary route for certain tasks. This weight structure emerges from training, confirming that WoT identifies and reinforces pathways crucial for solving various problem types.

## D Chat-Based WoT Reasoning Interface

This section demonstrates how the Weight-of-Thought (WoT) reasoning process can be made transparent in an interactive LLM setting. Rather than returning only a final answer, the system exposes internal details—including node activations, message passing, and verification—that reveal how it arrives at its conclusion. The following figures present separate example dialogues.

## Node Similarity Evolution During Training

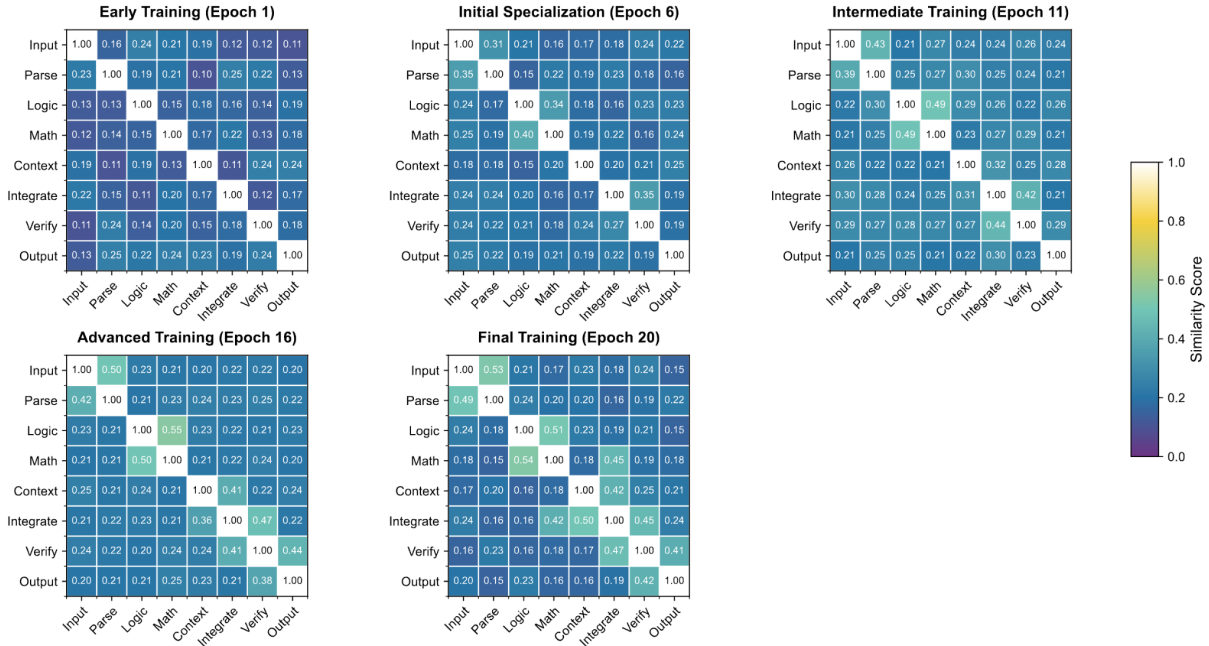


Figure 8: Evolution of node similarity during training. The composite image illustrates how the pairwise similarity between node embeddings evolves over training epochs, with increased specialization shown by lower off-diagonal similarities and stronger diagonal dominance.

In Figures 10–13, the system’s dialogue showcases how internal nodes (e.g., Math, Logic, Algebra, Verification, Geometry) are activated with specific weights and how their outputs are aggregated to produce the final answer. This transparency offers users insights into the model’s decision-making process, enhancing interpretability and trust.

## E Implementation Details (Extended)

Beyond the architecture and training specifics described in the main text, we provide additional information about hyperparameters, data splits, and libraries:

**Hyperparameters.** We used a weight decay of  $1 \times 10^{-4}$ , a linear warm-up for 5% of total steps, and a dropout rate of 0.1 in the node MLP layers.

**Data Splits.** Each dataset was randomly split into 80% training, 10% validation, and 10% test. We ensured that no problem or prompt overlap existed between splits.

**Libraries.** The WoT model was implemented in PyTorch 1.13. We utilized Hugging Face Transformers (v4.25) for the GPT-2 backbone and PGF-Plots/TikZ for all visualizations.

## F Visualization of Reasoning Steps (Extended)

Figure 8 outlines the detailed multi-step reasoning flow. In practice, each round of message passing updates node states based on attention signals derived from weight-guided edges. This process often manifests as an initial broad exploration of possible solution pathways, followed by a focused consolidation phase in later steps. Table 4 shows a textual trace of these updates in a single problem instance.

| Step | Node Updates and Key Observations   |
|------|---|
| 1    | Parse node recognizes variables in the question. Math node receives moderate activation to check for numerical clues.                     |
| 2    | Logic node evaluates potential constraints, verifying problem consistency. Verify node slightly active, cross-checking partial solutions. |
| 3    | Math node intensifies, solving partial equations. Attention from Logic to Verify nodes increases.   |
| 4    | Verify node cross-references the derived solution, finalizing the outcome. Output node triggers the final generation.                     |

Table 4: Illustrative textual trace of node updates across four reasoning steps in WoT.

This extended look reveals how WoT systematically exploits its internal weight-guided structure to converge on accurate, interpretable solutions.

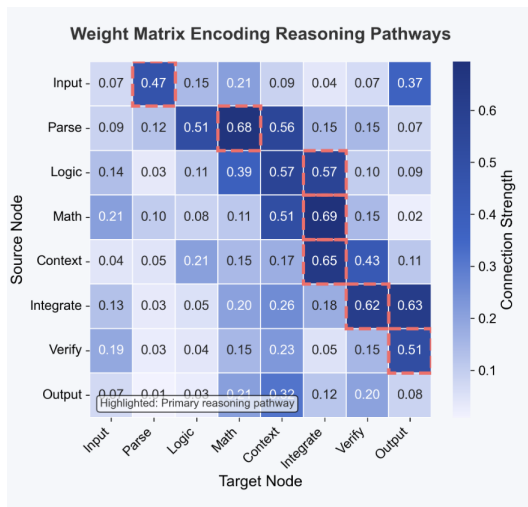


Figure 9: Weight matrix encoding reasoning pathways. Darker shades represent stronger connection strengths between source and target nodes. Dashed boxes highlight the primary reasoning pathway, revealing how WoT routes information from Input to Output via Parse, Logic, Math, and Verify nodes.

**User:**  
 'What is the next number in the sequence: 3, 6, 12, 24, ... ?'

**WoT Reasoning Steps:**

- (1) *Math Node:* Detects a doubling pattern.
- (2) *Logic Node:* Verifies consistency across terms.
- (3) *Output Node:* Computes  $24 \times 2 = 48$ .

**Answer:**  
 The next number is 48.

Figure 10: Chat Example 1: Mathematical Sequence Reasoning. The WoT system detects a doubling pattern and calculates the next number as 48.

**User:**  
 'If all Floops are Gloops and all Gloops are Hloops, are all Floops Hloops?'

**WoT Reasoning Steps:**

- (1) *Logic Node:* Identifies the transitive relation.
- (2) *Verification Node:* Confirms logical consistency.
- (3) *Output Node:* Infers that all Floops are Hloops.

**Answer:**  
 Yes, all Floops are Hloops.

Figure 11: Chat Example 2: Logical Deduction. The WoT system identifies a transitive relation and confirms that all Floops are Hloops.

**User:**  
 'Solve for  $x$  in:  $4x - 7 = 13$ '

**WoT Reasoning Steps:**

- (1) *Algebra Node:* Sets up the equation  $4x - 7 = 13$ .
- (2) *Math Node:* Isolates  $x$  (i.e., adds 7 and divides by 4).
- (3) *Verification Node:* Confirms the solution.

**Answer:**  
 $x = 5$ .

Figure 12: Chat Example 3: Algebraic Problem Solving. The WoT system sets up the equation and isolates  $x$  to find  $x = 5$ .

**User:**  
 'Is every square a rectangle?'

**WoT Reasoning Steps:**

- (1) *Geometry Node:* Analyzes definitions.
- (2) *Logic Node:* Recognizes that squares meet rectangle criteria.
- (3) *Output Node:* Confirms the hierarchical relationship.

**Answer:**  
 Yes, every square is a rectangle.

Figure 13: Chat Example 4: Geometric Reasoning. The WoT system confirms, via its Geometry and Logic Nodes, that every square is a rectangle.

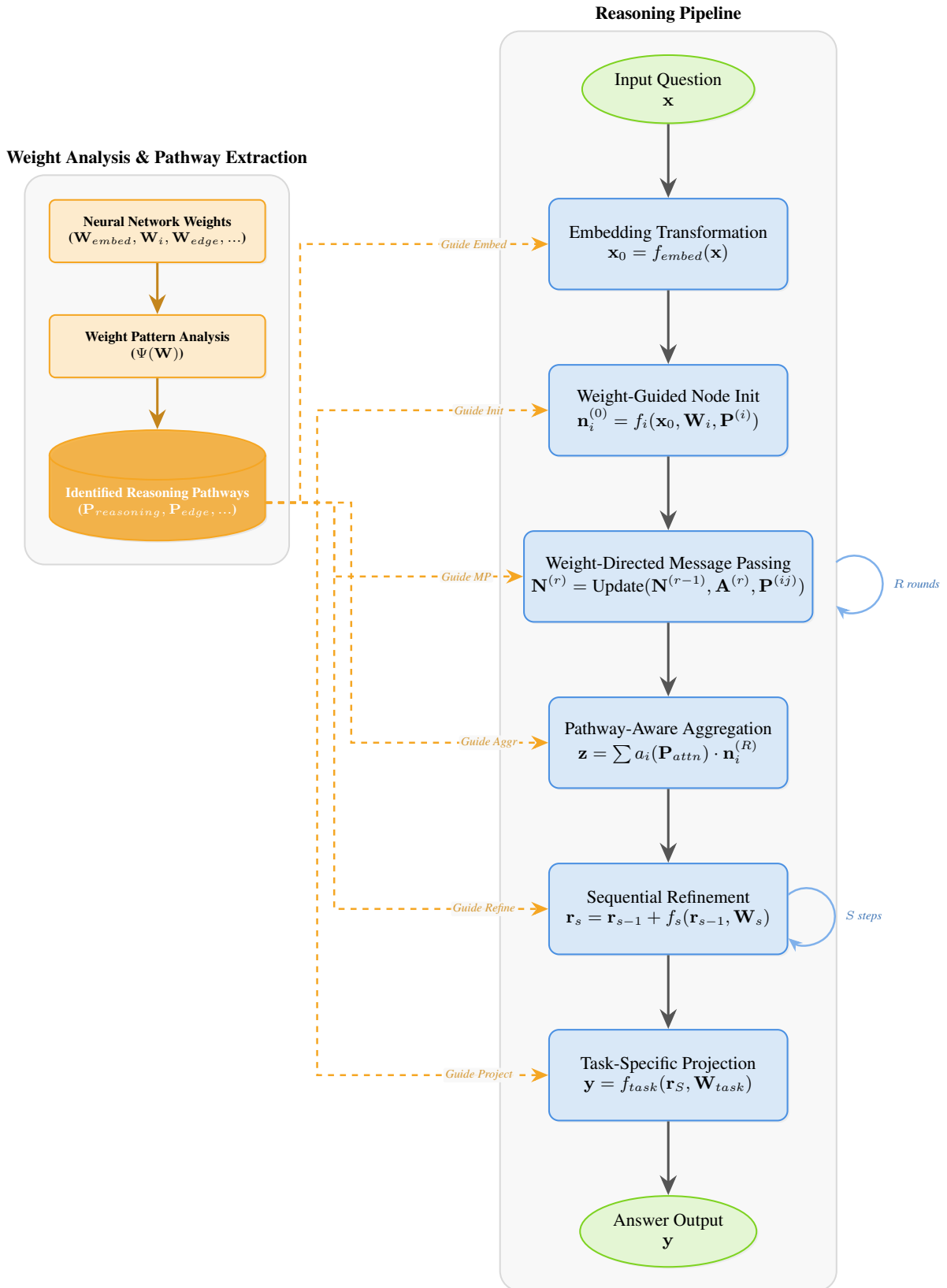


Figure 14: Detailed schematic of the Weight-of-Thought (WoT) reasoning architecture. The layout distinguishes the Weight Analysis & Pathway Extraction module (left, orange) from the main Reasoning Pipeline (right, blue/green). This module analyzes network weights ( $\mathbf{W}$ ) via  $\Psi$  to yield explicit Reasoning Pathways ( $\mathbf{P}$ ). These pathways guide stages of the pipeline (Embedding, Node Init, Message Passing, Aggregation, Refinement, Projection), shown by dashed influence lines connecting cleanly to the pipeline steps. The pipeline processes the input ( $\mathbf{x}$ ), featuring iterative Message Passing ( $R$  rounds) and Refinement ( $S$  steps), to produce the final answer ( $\mathbf{y}$ ).