TÉCNICO LISBOA

Enhancing Large Language Models with Dynamic Thinking Tokens

David Afonso Prazeres da Cruz Valente

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors: Prof. Maria Inês Camarate de Campos Lynce de Faria
Prof. Tomás Mikolov

Examination Committee

Chairperson: Prof. Alberto Abad Gareta
Supervisor: Prof. Maria Inês Camarate de Campos Lynce de Faria
Members of the Committee: Prof. Maria Inês Camarate de Campos Lynce de Faria
Prof. Francisco António Chaves Saraiva de Melo

November 2025

Declaration
I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to thank my parents for their friendship, encouragement, and caring over all these years, for always being there for me through thick and thin, and without whom this project would not be possible. A special and heartfelt thank you goes to my little sister, whose unwavering support and encouragement brightened even the most challenging days. I am also deeply grateful to my godfather for his invaluable help and guidance, and to my grandparents for their constant understanding and support. I would also like to acknowledge my dissertation supervisors, Prof. Inês Lynce and Prof. Tomás Mikolov, for their insight, support, and sharing of knowledge that has made this Thesis possible. Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you. This work was supported by the 'OptiGov' project, with ref. n. 2024.07385.IACDC (DOI: 10.54499/2024.07385.IACDC), fully funded by the 'Plano de Recuperação e Resiliência' (PRR) under the investment 'RE-C05-i08 - Ciência Mais Digital' (measure 'RE-C05-i08.m04'), framed within the financing agreement signed between the 'Estrutura de Missão Recuperar Portugal' (EMRP) and Fundação para a Ciência e a Tecnologia, I.P. (FCT) as an intermediary beneficiary. To each and every one of you – Thank you.

Abstract

Large Language Models have demonstrated impressive reasoning through Chain-of-Thought prompting, but these approaches require explicit supervision and generate substantial token overhead. This thesis introduces Dynamic Thinking Tokens (DTT), enabling LLMs to perform reasoning in continuous embedding space using reinforcement learning without step-by-step demonstrations. DTT integrates continuous hidden state modulation with Group Relative Policy Optimization (GRPO), employing learnable gating mechanisms and adaptive blending parameters that maintain complete differentiability while avoiding discrete mode transitions. We evaluate DTT on GSM8K, ProntoQA, and ProsQA using a 124M parameter GPT-2 model. DTT achieves 49-79% token reduction versus Chain-of-Thought while improving over baseline reinforcement learning: 26.8% accuracy on GSM8K (versus 21.3%), 84.9% on ProntoQA (versus 82.1%), and 82.7% on ProsQA (versus 79.4%). The framework closes approximately 40-45% of the gap to supervised approaches, demonstrating that reinforcement learning can discover effective latent reasoning strategies autonomously.

Keywords

Large Language Models, Latent Space Reasoning, Reinforcement Learning, Chain-of-Thought, Efficient Inference

Resumo

Os Large Language Models demonstraram capacidades impressionantes de raciocínio através de técnicas como Chain-of-Thought prompting, mas estas abordagens requerem supervisão explícita e geram overhead substancial de tokens. Esta tese introduz Dynamic Thinking Tokens (DTT), permitindo que LLMs realizem raciocínio em espaço de embeddings contínuo usando reinforcement learning sem demonstrações passo a passo. O DTT integra modulação contínua de estados ocultos com Group Relative Policy Optimization (GRPO), empregando mecanismos de gating aprendíveis e parâmetros de blending adaptativos que mantêm diferenciabilidade completa evitando transições de modo discretas. Avaliamos o DTT em GSM8K, ProntoQA e ProsQA usando um modelo GPT-2 de 124M parâmetros. O DTT alcança redução de 49-79% de tokens versus Chain-of-Thought enquanto melhora sobre reinforcement learning base: 26.8% de precisão em GSM8K (versus 21.3%), 84.9% em ProntoQA (versus 82.1%), e 82.7% em ProsQA (versus 79.4%). A framework fecha aproximadamente 40-45% da lacuna para abordagens supervisionadas, demonstrando que reinforcement learning pode descobrir estratégias eficazes de raciocínio latente de forma autónoma.

Palavras Chave

Modelos de Linguagem Grandes, Raciocínio em Espaço Latente, Aprendizagem por Reforço, Cadeia de Pensamento, Inferência Eficiente

Contents

List of Figures

List of Tables

1

Introduction

Contents

1.1 Motivation: The Reasoning Challenge in Large Language Models

Large Language Models have achieved remarkable success across diverse domains, from natural language understanding to code generation. Yet despite these advances, they continue to struggle with tasks requiring systematic reasoning—a limitation that becomes apparent even in seemingly simple problems. Consider asking a state-of-the-art model to count the occurrences of the letter ’r’ in the word "strawberry." As illustrated in Figure 2.5, models that can write sophisticated essays or explain complex scientific concepts often fail at this elementary counting task, revealing fundamental gaps in their reasoning capabilities.

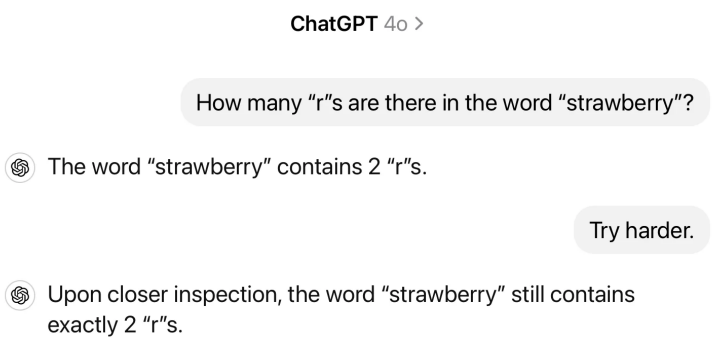

Figure 1.1: State-of-the-art LLMs struggling with a simple reasoning task.

The problem extends far beyond counting exercises. Mathematical word problems requiring multi-step arithmetic, logical puzzles demanding systematic deduction, and questions requiring navigation through complex knowledge structures all expose similar weaknesses. When models do succeed at these tasks, they often require several seconds of "thinking time" and generate verbose reasoning chains that consume substantial computational resources. A model solving a grade-school math problem might produce hundreds of tokens explaining each arithmetic operation, when the problem requires only a simple numerical answer. This verbosity translates directly to increased inference costs, higher latency, and greater energy consumption—making sophisticated reasoning impractical for resource-constrained deployments like mobile devices or real-time systems.

The core issue lies in how these models perform reasoning. Current approaches like Chain-of-Thought (CoT) prompting [2] ask models to explicitly articulate their reasoning steps in natural language tokens. While this sometimes improves accuracy, it creates a fundamental tension: the model must simultaneously reason about the problem and express that reasoning in grammatically correct, human-readable text. This dual burden constrains the reasoning process to follow the linear, token-by-token structure of language generation. Human reasoning, by contrast, often involves parallel consideration of multiple possibilities, implicit backtracking when paths prove unfruitful, and intuitive leaps that resist clean verbal articulation. The sequential token paradigm makes these reasoning patterns difficult or impossible to represent efficiently.

Moreover, CoT approaches require either carefully engineered prompts that demonstrate reasoning patterns or expensive supervised fine-tuning on datasets with explicit reasoning annotations. Creating such datasets demands significant human effort—experts must solve problems while verbalizing their complete thought processes, a cognitively demanding task that produces data much more slowly than typical annotation work. For many reasoning domains, particularly those requiring specialized expertise, collecting comprehensive demonstrations at scale remains prohibitively expensive. This data scarcity creates a bottleneck that limits the development of more capable reasoning systems.

The computational inefficiency of token-based reasoning becomes particularly problematic in practical deployments. Educational platforms serving millions of students cannot afford the latency and compute costs of generating lengthy reasoning chains for every practice problem. Financial systems making split-second trading decisions cannot wait for verbose explanations. Mobile applications running on battery-powered devices cannot sustain the memory and energy demands of explicit reasoning traces. Scientific discovery tools exploring vast hypothesis spaces cannot afford the computational overhead of verbalizing every intermediate reasoning step. These applications need models that can reason efficiently—performing complex multi-step inference while generating minimal tokens and maintaining acceptable response times.

1.2 Our Approach: Reasoning in Continuous Latent Space

This thesis explores an alternative paradigm: enabling models to perform reasoning directly in their continuous embedding space rather than through sequential token generation. Instead of forcing the model to express every reasoning step as discrete tokens, we allow it to maintain and refine internal representations that capture reasoning progress without the overhead of language generation. This approach draws inspiration from recent work demonstrating that latent-space reasoning can achieve comparable or superior accuracy to traditional methods while using 50-70% fewer tokens.

To understand our contribution, it is essential to first consider how standard transformers process information. In a typical transformer architecture, input tokens are converted into embeddings, processed through multiple attention and feedforward layers, and finally decoded back into output tokens. Each layer transforms the hidden representations, but there is no explicit mechanism for the model to "pause and think" between reading the input and producing the output. The model processes information in a single forward pass, with reasoning implicitly distributed across layers but never explicitly accumulated or refined.

Our Dynamic Thinking Tokens framework, illustrated in Figure 1.2, extends the standard transformer architecture with a minimal but crucial addition: a gating layer that creates a reasoning feedback loop. As shown in the figure, after the embedding layer converts input tokens (shown in yellow and red) into vector representations, our gating mechanism (indicated by the dashed box) receives two inputs: the current token embeddings and the hidden state from the previous processing step. This gating layer dynamically decides how to blend these two sources of information—how much to rely on the immediate token content versus the accumulated reasoning context.

The key distinction from standard transformers lies in this feedback mechanism. In a regular transformer, information flows unidirectionally from embeddings through attention layers to the decoder. In our approach, the hidden states $h_1, h_2, \ldots, h_5$ (shown in blue) carry forward a continuous reasoning

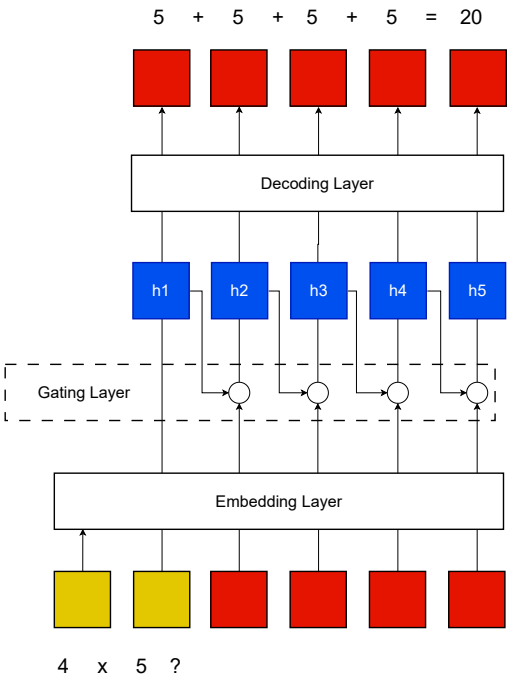

Figure 1.2: Continuous latent space reasoning: modulating internal representations for efficient inference without generating explicit token sequences. Unlike standard transformers that directly map input embeddings through attention and feedforward layers to output tokens, our approach introduces a gating layer that dynamically blends current token embeddings with accumulated reasoning context from the previous hidden state. The gating mechanism (dashed box) determines how much to rely on discrete token information versus continuous reasoning residuals, enabling the model to perform sophisticated reasoning operations entirely within its latent space before producing a final answer.

context that influences how subsequent tokens are processed. When the model encounters a reasoning problem like "$4 \times 5 =?$", it doesn't just process these tokens independently through attention layers. Instead, the gating layer allows earlier hidden states to accumulate reasoning progress—for instance, recognizing that this is a multiplication problem, activating relevant mathematical patterns, and preparing the appropriate computational strategy—all without generating any intermediate text tokens.

This creates a form of "internal deliberation" where the model refines its understanding across multiple tokens before committing to an answer. The yellow tokens in the input represent the problem context, while red tokens indicate where the model has sufficient information to engage reasoning. The gating mechanism learns to recognize these situations and smoothly transition from primarily using token embeddings (when reading the problem) to incorporating accumulated reasoning residuals (when solving it). The final output tokens (shown in red at the top) are produced only after this internal reasoning process completes, resulting in the direct answer "$= 20$" without verbose intermediate explanations.

Critically, this entire process remains fully differentiable. The gating decisions are not discrete switches but continuous blending operations, allowing gradients to flow backward through the reasoning process during training. This differentiability enables us to train the system using reinforcement learning

with only sparse reward signals—the model simply needs to know whether its final answer is correct or incorrect, not how to articulate each reasoning step in words.

We introduce Dynamic Thinking Tokens (DTT), a framework that integrates this continuous latent reasoning with reinforcement learning optimization. Our architecture embeds specialized reasoning components directly into transformer layers, creating differentiable pathways for refining hidden states throughout the model's forward pass. These components employ learnable gating mechanisms and adaptive blending parameters that dynamically interpolate between standard token embeddings and accumulated reasoning residuals. When the model encounters a reasoning problem, these components engage in continuous modulation of internal representations—effectively "thinking" in the high-dimensional embedding space before producing a final answer.

Critically, this reasoning process maintains complete differentiability, enabling optimization through policy gradient methods. We employ Group Relative Policy Optimization (GRPO) [6], a reinforcement learning algorithm that generates multiple candidate solutions for each problem and computes policy updates based on relative performance within each group. This approach addresses the data scarcity challenge: rather than requiring explicit demonstrations of correct reasoning steps, the model learns from sparse reward signals indicating whether final answers are correct. By exploring the continuous latent space autonomously, the model can discover effective reasoning strategies without step-by-step human supervision.

The framework employs a sophisticated multi-component optimization strategy that balances preservation of pre-trained knowledge with development of specialized reasoning capabilities. Base transformer parameters are optimized conservatively to maintain linguistic competence, while reasoning components receive higher learning rates enabling rapid adaptation to policy gradient signals. A gated reward function encourages both correctness and efficiency, penalizing unnecessarily verbose outputs to promote concise latent reasoning. This design enables the model to discover when and how to engage its latent reasoning mechanisms based on problem characteristics.

1.3 Contributions and Thesis Structure

This research makes several key contributions to efficient reasoning in large language models. First, we develop a fully differentiable continuous reasoning architecture that addresses the gradient flow challenges inherent in discrete mode-switching approaches. Our learnable blending mechanisms enable smooth interpolation between discrete embeddings and continuous reasoning states while maintaining training stability. Second, we demonstrate the first successful integration of continuous latent reasoning with group-relative policy optimization, showing that reinforcement learning can effectively guide reasoning development without explicit supervision. Third, we provide comprehensive empirical evaluation

across mathematical, logical, and multi-hop reasoning tasks, demonstrating 49-79% token reduction compared to Chain-of-Thought methods while achieving meaningful performance improvements over baseline approaches.

Beyond these technical contributions, we offer extensive mechanistic analysis of how latent reasoning develops during training. We characterize the blending dynamics that emerge as models learn task-appropriate strategies, analyze the progressive uncertainty reduction that occurs during reasoning, and identify the architectural depth requirements for different reasoning complexities. We also provide honest assessment of current limitations—substantial performance gaps remain compared to state-of-the-art supervised methods, and the framework exhibits sensitivity to hyperparameters and training stochasticity. These findings establish both the promise and challenges of the latent reasoning paradigm.

The remainder of this thesis proceeds as follows. Chapter 2 reviews related work on reasoning in language models, covering Chain-of-Thought prompting, latent-space approaches, and reinforcement learning for language model training. Chapter 3 presents our solution in detail, including the continuous reasoning architecture, GRPO integration, training methodology, and implementation considerations. Chapter 4 provides comprehensive evaluation across three reasoning benchmarks, mechanistic analysis of reasoning behaviors, and systematic ablation studies. Chapter 5 concludes with discussion of limitations and directions for future research. Together, these chapters demonstrate that continuous latent reasoning combined with reinforcement learning offers a promising direction for developing more efficient, autonomous reasoning systems that address the computational and data constraints limiting current approaches.

The code is available at https://github.com/DavidAfonsoValente/DTT.

2

Background and Related Work

Contents

Neural reasoning systems have evolved significantly in recent years, progressing from simple pattern matching to sophisticated reasoning capabilities. This chapter establishes the theoretical foundations and examines the historical context that underpin our Dynamic Thinking Tokens framework. We begin by exploring the fundamental concepts of neural representations and optimization that form the substrate for modern reasoning systems. We then trace the evolution of machine reasoning from symbolic approaches through to contemporary neural methods, examining how each paradigm shift has shaped our understanding of computational reasoning. Finally, we analyze current state-of-the-art approaches and their limitations, which directly motivate our DTT framework.

2.1 Theoretical Foundations

2.1.1 Neural Representations and Embedding Spaces

The concept of embedding spaces has emerged as a cornerstone of modern machine learning systems, providing a geometric framework for representing and manipulating semantic information [7]. These high-dimensional vector spaces offer a powerful abstraction for capturing complex relationships and enabling sophisticated reasoning operations through geometric transformations [8]. At the heart of embedding space theory lies the manifold hypothesis, which posits that real-world data, despite its apparent complexity, typically lies on or near a lower-dimensional manifold within the high-dimensional embedding space. The manifold structure suggests that meaningful transformations in the data space can be captured through continuous deformations along this manifold, providing a geometric interpretation of reasoning processes [9].

The mathematical foundations of embedding spaces center on carefully chosen distance metrics and geometric operations that preserve semantic relationships [10]. The most commonly employed similarity measure is the cosine similarity between embeddings, defined as $\text{sim}(v_i, v_j) = \frac{v_i \cdot v_j}{||v_i|| \, ||v_j||}$, where v_i and v_j are vector embeddings. The organization of embedding spaces is governed by the principle of distributional semantics [11], which suggests that the meaning of entities can be inferred from their patterns of co-occurrence and interaction with other entities. This principle manifests in the way neural networks learn to structure their internal representations [12], creating a geometric space where semantic relationships are reflected in the relative positions and distances between embeddings.

Recent advances in contextual embeddings, exemplified by models like BERT and its successors, have demonstrated the importance of context-dependent representations. Unlike static embeddings, these models generate representations that vary based on the surrounding context [13], capturing the nuanced ways in which meaning shifts across different contexts. This dynamic nature is achieved through sophisticated attention mechanisms that compute context-dependent embeddings as $h_i = f(x_i, x_{-i}; \theta)$, where h_i is the contextual embedding for token x_i, x_{-i} represents the surrounding context, and θ denotes the model parameters.

Navigation in high-dimensional embedding spaces presents unique challenges that fundamentally impact how neural systems process and manipulate information. The topology of the embedding space itself can be leveraged for more effective navigation through geodesic paths [14], ensuring that intermediate points along the path remain semantically meaningful. This is particularly relevant for our DTT framework, which relies on efficient traversal of the embedding space during latent reasoning phases. The DTT approach leverages these geometric properties to perform reasoning operations directly in the embedding space, bypassing the limitations of discrete token generation and enabling more efficient reasoning pathways.

2.1.2 Optimization in Neural Systems

The optimization of neural reasoning systems presents unique challenges stemming from the high-dimensional nature of the parameter space and the complexity of the loss landscape. Gradient-based optimization serves as the primary tool for training neural networks, with the general update rule $\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$, where θ_t represents the parameters at iteration t, η is the learning rate, and $\nabla_\theta \mathcal{L}(\theta_t)$ is the gradient of the loss function. However, the effectiveness of this simple update rule is complicated by several factors unique to neural systems. The loss landscape of deep neural networks is characterized by numerous local optima, saddle points, and plateaus that can significantly impact optimization dynamics. Recent theoretical work has shown that in high-dimensional spaces, saddle points rather than local minima present the primary challenge for optimization [15].

To address these challenges, adaptive optimization methods have become increasingly sophisticated. The Adam optimizer [16] combines momentum with adaptive learning rates through a formulation that provides several advantages for training deep neural networks, including robustness to gradient scaling and improved convergence properties. For the DTT framework, we leverage these optimization insights to develop a phased training approach that gradually shifts the optimization objective from auxiliary reasoning rewards to final performance metrics. This approach helps navigate the complex loss landscape more effectively, avoiding local optima and ensuring stable convergence to high-performance parameter configurations.

2.1.3 Reinforcement Learning for Reasoning

The integration of reinforcement learning with neural reasoning systems has opened new avenues for developing adaptive reasoning strategies [17]. RL provides a framework for learning through interaction, particularly valuable when intermediate reasoning steps lack explicit supervision [18]. The standard RL objective in reasoning tasks is formulated as $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$, which captures the expected return under a policy parameterized by θ. The policy π_θ generates trajectories τ of reasoning steps, where each step receives a reward r_t. The discount factor γ determines how much future rewards are valued compared to immediate ones, enabling the system to learn reasoning strategies that maximize long-term performance [19].

Group Relative Policy Optimization (GRPO) is a reinforcement learning algorithm particularly well-suited for training language models on reasoning tasks and forms the core of our DTT training methodology. For a given prompt q, GRPO generates G completions, computes rewards, and calculates a normalized advantage. The policy π_θ is updated by minimizing a carefully designed loss function that ensures stable updates by leveraging group-based advantages, making it effective for reasoning tasks where exploration and alignment with human-like reasoning are critical. In our DTT framework, we

extend GRPO with a novel phased reward system that combines final answer correctness with intermediate reasoning quality metrics and latent compression rewards, with weights dynamically adjusted during training to gradually shift focus from intermediate reasoning quality to final performance.

2.1.4 Technical Foundations of Language Models

The architecture of modern language models represents a careful orchestration of multiple components [20], with the transformer architecture revolutionizing how neural networks process sequential information. The fundamental attention mechanism, enhanced through multi-head attention, forms the core of these systems. The mechanism is defined as $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, where Q, K, and V represent the query, key, and value matrices, and d_k is the dimension of the key vectors. This enables the model to attend to different parts of the input with varying weights, creating context-aware representations crucial for complex reasoning tasks. The multi-head variant allows the model to attend to different relationship types simultaneously through parallel attention heads that are concatenated and projected.

Position-aware processing is achieved through positional encodings, addressing the inherent permutation invariance of self-attention using sinusoidal functions. These encodings allow the model to incorporate sequential information without recurrent connections, enabling more efficient parallel processing. Layer normalization stabilizes training by normalizing activations across features, helping manage internal covariate shift and allowing for more stable and efficient training. The transformer architecture combines these components with feed-forward networks in a residual connection framework.

The scaling behavior of these models follows empirical laws [21] that relate model performance to architectural parameters, expressing relationships between the number of parameters, dataset size, and model performance. The emergence of reasoning capabilities appears to follow a phase transition-like behavior, becoming reliably present only above critical model scales. Understanding these scaling laws is crucial for developing efficient reasoning systems like DTT, which aims to achieve sophisticated reasoning capabilities with modest model sizes by leveraging the continuous latent space for reasoning operations. Our DTT framework builds upon these transformer foundations, introducing specialized components for latent reasoning while maintaining compatibility with the core transformer architecture.

2.2 Evolution of Machine Reasoning

2.2.1 Historical Foundations

The genesis of machine reasoning can be traced to the seminal work of Newell and Simon with their General Problem Solver (GPS) in 1959 [22]. This groundbreaking system introduced means-ends anal-

ysis as a fundamental problem-solving strategy, demonstrating for the first time that machines could engage in structured reasoning behaviors. The GPS architecture established several crucial principles that continue to influence modern AI systems, including the separation of domain knowledge from reasoning mechanisms and the importance of heuristic search strategies [23].

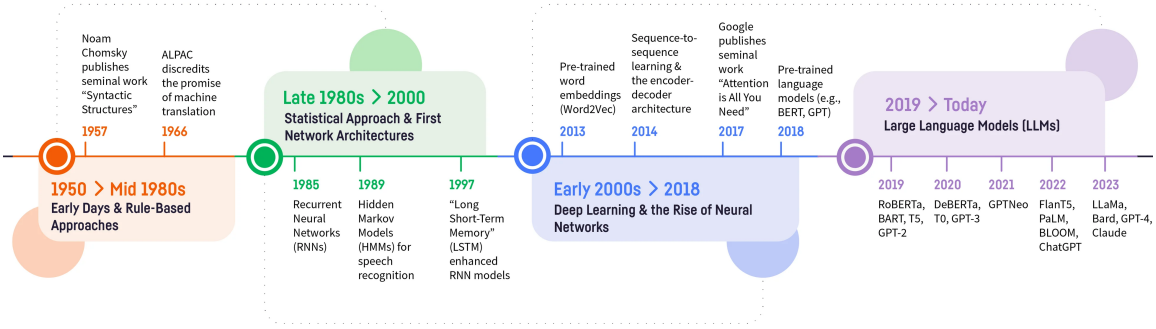

Figure 2.1: Timeline of the evolution of Artificial Intelligence. Image taken from [1].

The 1960s witnessed the emergence of formal logical systems and automated theorem provers, exemplified by Robinson's Resolution principle [24]. This period saw the development of the first automated reasoning systems capable of proving mathematical theorems, though these systems revealed fundamental limitations in handling real-world ambiguity and uncertainty. McCarthy's situation calculus [25] provided a formal framework for reasoning about actions and change, while Hayes's naive physics manifesto [26] highlighted the challenges of encoding common-sense knowledge.

The 1970s marked the advent of expert systems, with MYCIN [27] and DENDRAL [28] demonstrating practical applications of artificial intelligence in medical diagnosis and chemical analysis. These systems introduced probabilistic reasoning mechanisms and showed how domain-specific knowledge could be effectively encoded and utilized. A transformative breakthrough came with Pearl's development of Bayesian networks [29], which provided a rigorous framework for reasoning under uncertainty and established fundamental principles for causal reasoning.

The late 1980s witnessed a paradigm shift with the resurgence of neural networks, catalyzed by the development of backpropagation [30] and parallel distributed processing models. This transition from symbolic to subsymbolic approaches marked a fundamental change in how researchers approached machine reasoning. The early 2000s saw the emergence of deep learning architectures [31] that would eventually revolutionize the field. Convolutional Neural Networks achieved unprecedented success in computer vision tasks, while Recurrent Neural Networks with Long Short-Term Memory units advanced sequence processing capabilities [32]. The introduction of attention mechanisms marked another significant advance, enabling models to dynamically focus on relevant information and laying the foundation for the Transformer architecture.

2.2.2 Foundation Models and Modern Language Models

The introduction of the Transformer architecture marked a paradigm shift in natural language processing and machine reasoning [20]. This revolutionary architecture eliminated the need for recurrent connections through the innovative use of self-attention mechanisms, enabling more efficient parallel processing and better handling of long-range dependencies. The multi-head attention mechanism further enhanced the model's capability by allowing it to attend to information from different representation subspaces simultaneously [33]. The positional encoding scheme addressed the inherent challenge of sequence ordering in attention-based models through sinusoidal or learned embeddings, maintaining awareness of token ordering while preserving the benefits of parallel processing [34].

The development of large language models has progressed through several generations, each marked by significant advances in architecture, scale, and capabilities. BERT introduced bidirectional training of Transformer encoders, establishing new state-of-the-art results across numerous natural language understanding tasks. The GPT series, beginning with GPT-1 [35], demonstrated the potential of unidirectional models trained on massive datasets, with each iteration showcasing dramatic improvements in reasoning capabilities. GPT-3 [36] represented a quantum leap in scale and capability, demonstrating emergent abilities in few-shot learning and complex reasoning tasks with 175 billion parameters. Subsequent developments, including GPT-4 [37], have further pushed the boundaries of what's possible with large language models, particularly in areas requiring complex reasoning and multi-step problem solving.

The PaLM model [38] introduced the Pathways architecture, enabling more efficient use of computational resources and better handling of multi-task learning. This architecture demonstrated remarkable capabilities in reasoning tasks, particularly in areas requiring step-by-step deduction and mathematical problem solving. Anthropic's Claude models [39] have advanced the field through innovative approaches to model alignment and reasoning capabilities, while the Llama series [40] has demonstrated that strong performance can be achieved with more efficient architectures and open-source approaches.

Research into scaling laws has provided crucial insights into the relationship between model size, compute budget, and performance [21]. These studies have revealed power-law relationships governing model improvement with scale, understanding that has guided the development of compute-optimal training strategies and influenced architectural decisions in modern language models. Architectural innovations have focused on improving efficiency and capability simultaneously. Mixture of Experts approaches [41] have demonstrated the potential for conditional computation, allowing models to scale to massive sizes while maintaining reasonable computational costs. Sparse attention mechanisms [42] and efficient attention variants have addressed the quadratic complexity of self-attention, enabling the processing of longer sequences and more efficient training.

2.3 Contemporary Reasoning Approaches

2.3.1 Chain-of-Thought Reasoning

Chain-of-Thought (CoT) prompting represents a fundamental breakthrough in neural reasoning, providing a framework for decomposing complex problems into interpretable intermediate steps [2]. The theoretical foundations of CoT build upon the transformer's attention mechanisms, where each reasoning step contributes to the overall solution through iterative refinement. The mathematical formulation of this process can be expressed through the attention computation at each step, where the current query attends to keys and values from all previous steps. This formulation enables the model to maintain coherence across the reasoning chain while progressively building toward a solution.

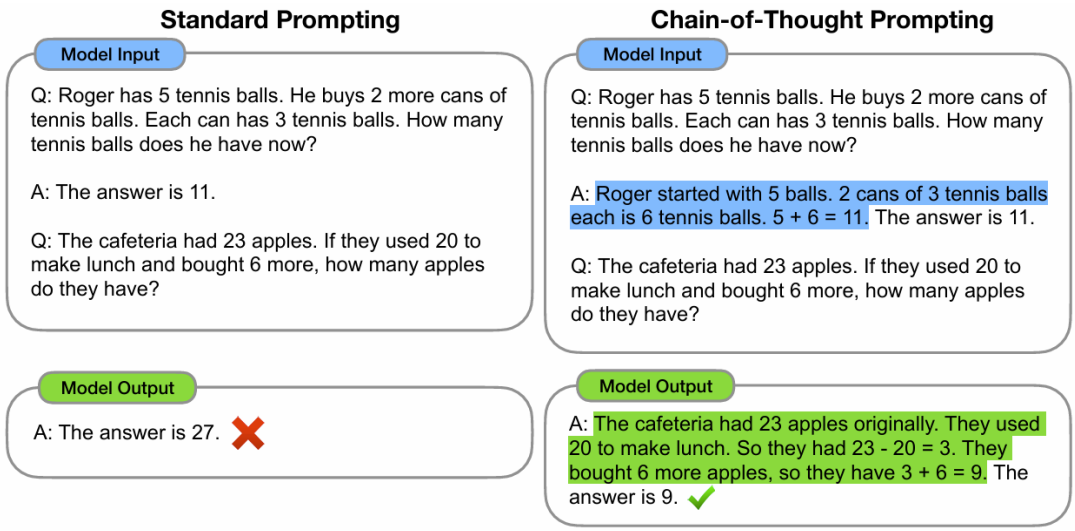

Figure 2.2: Regular Prompting vs Chain of Thought Prompting, image from [2].

The linguistic structure of CoT reasoning introduces specific constraints that shape both its capabilities and limitations. The transformation of thoughts into natural language tokens creates a mapping from the space of possible cognitive states to discrete token sequences [43]. This mapping process inevitably introduces information loss through discretization and sequential ordering constraints. The development of Zero-shot Chain-of-Thought [44] marked a significant advancement by eliminating the need for task-specific examples through careful prompt engineering. This approach demonstrates that the capacity for step-by-step reasoning can be activated through appropriate prompting, suggesting these capabilities may be latent in large language models rather than requiring explicit training [45].

Recent developments have introduced several sophisticated variants of the basic CoT framework. Self-Consistency methods [46] enhance the robustness of chain-of-thought reasoning by generating multiple reasoning paths and selecting the most consistent solutions through majority voting. The

Tree-of-Thought approach [47] represents a significant evolution by enabling breadth-first exploration of multiple reasoning pathways simultaneously, allowing the model to consider multiple approaches and maintain uncertainty about different reasoning pathways until more evidence is accumulated. Graph-of-Thought [48] extends these ideas further by representing reasoning steps as nodes in a directed graph, enabling more complex reasoning patterns and better handling of interdependencies between different lines of thought.

The token-based nature of current reasoning frameworks introduces several fundamental limitations. The sequential bottleneck created by the need to linearize thoughts into token sequences creates inherent trade-offs between reasoning depth and computational efficiency. Semantic drift represents another significant challenge, where consecutive reasoning steps may maintain local coherence while diverging from the global solution path. Recent work has explored various approaches to mitigating these limitations, including hybrid symbolic-neural architectures [49] and continuous reasoning frameworks.

2.3.2 Reinforcement Learning Advances in Reasoning

Reinforcement Learning has emerged as a powerful paradigm for enhancing reasoning capabilities in Large Language Models. Unlike traditional supervised methods that rely on static datasets, RL enables models to learn through interaction and feedback, making it particularly suitable for tasks involving sequential decision-making and planning. The application of RL to LLMs for reasoning tasks addresses several limitations of conventional methods. Chain-of-Thought prompting, while effective, requires carefully crafted prompts and often struggles with generalization across diverse tasks. Supervised fine-tuning is constrained by the availability of high-quality annotated datasets, which are scarce for many reasoning domains [50].

DeepSeek-R1 represents a groundbreaking advancement in applying RL to enhance reasoning in LLMs [51]. Unlike traditional approaches that typically rely solely on initial supervised fine-tuning, DeepSeek-R1 employs a multi-stage approach that combines supervised learning with reinforcement learning to optimize reasoning capabilities through carefully designed reward signals. The methodology involves a sophisticated three-stage training process: initial cold-start with high-quality Chain-of-Thought examples followed by reinforcement learning, generation of supervised fine-tuning data via rejection sampling with further fine-tuning, and final RL alignment focusing on helpfulness and harmlessness. This comprehensive approach enables remarkable performance across various reasoning benchmarks, matching state-of-the-art models while achieving impressive results on mathematical reasoning tasks.

A particularly noteworthy aspect of the DeepSeek-R1 methodology is its knowledge distillation capabilities. The researchers successfully distilled the model's reasoning abilities into smaller variants through supervised fine-tuning, producing impressive results where even smaller parameter models surpassed much larger models on certain mathematical reasoning tasks. This demonstrates that knowl-

edge transfer from larger models trained with RL can be more efficient than training smaller models from scratch using RL directly.

DeepSeek-R1 Zero represents a paradigm shift in the development of reasoning capabilities, trained exclusively through large-scale reinforcement learning and completely bypassing supervised fine-tuning. The fundamental hypothesis driving this approach is that large language models can autonomously develop sophisticated reasoning abilities through properly incentivized reinforcement learning alone. This addresses a critical limitation in current reasoning systems: the scarcity and cost of high-quality human-annotated reasoning datasets. The model employs Group Relative Policy Optimization (GRPO), a computationally efficient RL algorithm that eliminates the need for a separate critic model by estimating rewards through comparing outputs within groups of sampled responses.

A key innovation in DeepSeek-R1 Zero's training methodology is its carefully designed reward structure combining accuracy rewards for correctness and format rewards for structured outputs. The training process deliberately avoids using neural reward models to prevent reward hacking and maintain simplicity. The training dynamics revealed particularly interesting phenomena described as "Aha moments" – sudden, significant leaps in reasoning ability during training, mirroring human learning experiences where understanding suddenly crystallizes after periods of gradual progress.

Analysis of DeepSeek-R1 Zero's reasoning patterns revealed several emergent behaviors including self-verification, extended chains-of-thought, and spontaneous reflection. Despite these impressive capabilities, the model exhibited certain limitations including output readability issues, language mixing, and underperformance in general-purpose applications, which motivated the development of the more balanced DeepSeek-R1 model. The success of DeepSeek-R1 Zero challenges conventional wisdom about the necessity of supervised fine-tuning in developing advanced reasoning capabilities.

2.3.3 State-of-the-Art Reasoning Systems

OpenAI's O-series models, particularly O1 and O3, represent a significant advancement in artificial intelligence reasoning capabilities. While the exact architecture remains private, some researchers have theorized similarities between O-series models' reasoning mechanisms and recent academic work like Quiet-STaR [52], particularly in how they appear to perform intermediate reasoning steps. These models incorporate architectural innovations that fundamentally enhance their ability to perform complex reasoning tasks, implementing a novel approach to compute-adaptive reasoning where computational resources can be dynamically adjusted based on problem complexity [53].

A key innovation in the O-series models is the implementation of deliberative alignment, a training paradigm that enables more robust and reliable reasoning processes [54]. The resource requirements for these models are substantial, with benchmark evaluations revealing that high-performance configurations consumed approximately $1,000 in compute resources per task on challenging benchmarks [55].

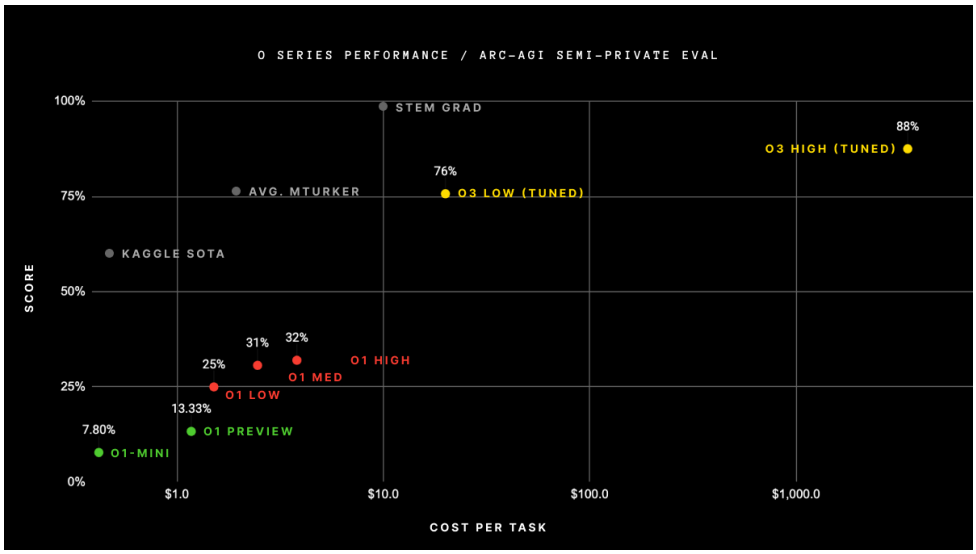

Figure 2.3: Comparison of Compute/Performance between o1, o3 and baselines on ARC-AGI. Image taken from [3].

Performance benchmarks demonstrate remarkable capabilities across various domains, achieving impressive accuracy on mathematical reasoning tasks and software engineering applications.

Recent research has introduced Thinking Tokens, a novel unsupervised approach to enhance reasoning capabilities in LLMs [4]. Unlike Chain-of-Thought prompting, which requires explicit step-by-step reasoning in the token space, Thinking Tokens operate by introducing specialized tokens that create computational delays in the model's processing pipeline, theoretically allowing for deeper reasoning within the model's latent space. The fundamental premise is grounded in the hypothesis that inserting intermediate "thinking" tokens enables models to engage in more sophisticated computation over their hidden states before generating outputs.

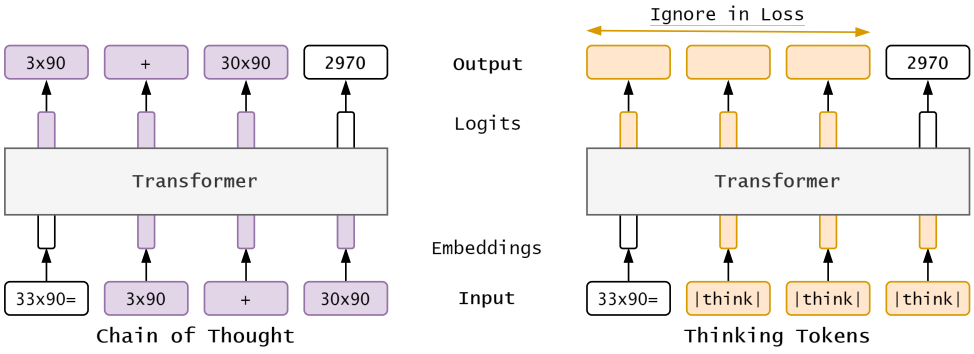

Figure 2.4: Comparison between Chain-of-Thought and Thinking Tokens approaches. While CoT relies on explicit verbal reasoning steps, TTs attempt to facilitate reasoning through implicit computational delays. Image taken from [4].

Initial implementations revealed several critical challenges in the Thinking Token approach. The pri-

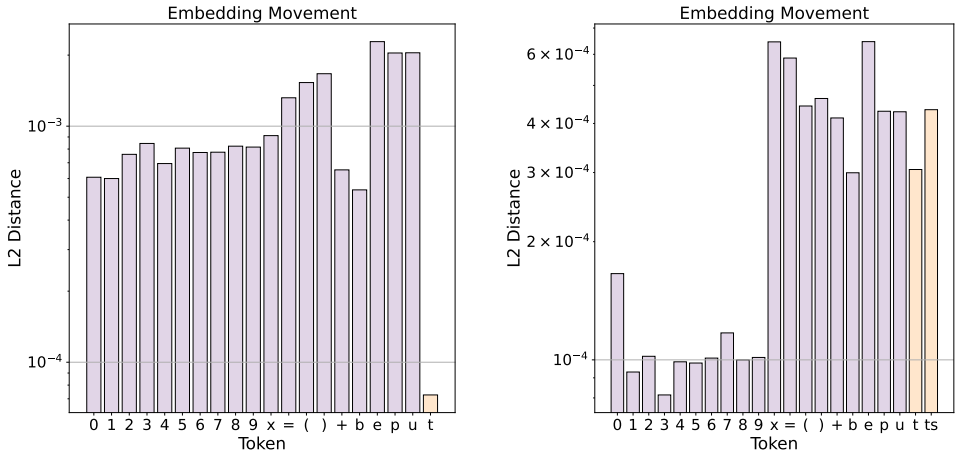

Figure 2.5: One TT embedding (t in place of all CoT tokens) hardly moves from the initialized value in relation to other embeddings whilst two TT embeddings (t in place of number CoT tokens and ts in place of symbolic CoT tokens) show clear deviation from initialization in relation to other embeddings. Image taken from [4].

mary limitation stems from the reliance on single-token embeddings, which creates significant obstacles in the learning process. The same token embedding must adapt to diverse reasoning contexts, creating ambiguity during training and leading to inconsistent learning signals. Empirical evaluations across various reasoning benchmarks revealed consistent patterns showing only marginal improvements over baselines, with performance gaps extending to natural language tasks as well. Recent work has explored the use of multiple distinct thinking tokens to address these limitations, showing promise in reducing gradient noise and improving embedding expressivity.

2.3.4 Continuous Latent Space Reasoning

The introduction of CoCoNut (Chain of Continuous Thought) in late 2024 marked a pivotal advancement in neural reasoning approaches [5]. While traditional Chain-of-Thought methods rely on generating explicit reasoning steps in natural language, CoCoNut fundamentally reimagines this process by operating directly in the model's latent space. This paradigm shift addresses a key limitation of language-based reasoning: the inherent inefficiency of forcing models to translate their internal reasoning processes into human-readable text at each step.

CoCoNut's architecture enables the model to maintain and manipulate its reasoning state entirely within its latent space during intermediate steps, only generating natural language when producing final outputs. This is achieved through a dual-mode processing system that allows the model to switch between latent reasoning and language generation modes using specialized control tokens. The approach creates an end-to-end differentiable reasoning process that can be optimized more effectively than tra-

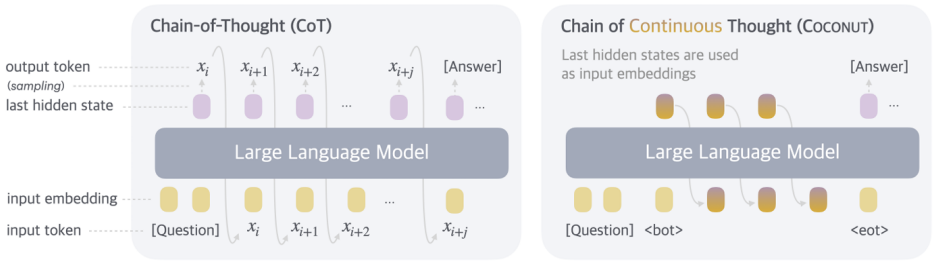

Figure 2.6: A comparison of Chain of Continuous Thought (CoCoNuT) with Chain-of-Thought (CoT). CoCoNuT regards the last hidden state as a representation of the reasoning state and directly uses it as the next input embedding, allowing the LLM to reason in an unrestricted latent space. Image taken from [5].

ditional token-based methods. A particularly innovative aspect is its emergent capacity for breadth-first search in reasoning tasks. Unlike traditional CoT approaches that commit to a single reasoning path, CoCoNut can simultaneously maintain multiple potential reasoning trajectories in its latent space, proving especially valuable for complex problems requiring extensive planning or backtracking.

Method	GSM8k		ProntoQA		ProsQA	
	Acc. (%)	# Tokens	Acc. (%)	# Tokens	Acc. (%)	# Tokens
CoT	42.9 ±0.2	25.0	98.8 ±0.8	92.5	77.5 ±1.9	49.4
No-CoT	16.5 ±0.5	2.2	93.8 ±0.7	3.0	76.7 ±1.0	8.2
iCoT	30.0*	2.2	99.8 ±0.3	3.0	98.2 ±0.3	8.2
Pause Token	16.4 ±1.8	2.2	77.7 ±21.0	3.0	75.9 ±0.7	8.2
COCONUT (Ours)	34.1 ±1.5	8.2	99.8 ±0.2	9.0	97.0 ±0.3	14.2
- *w/o curriculum*	14.4 ±0.8	8.2	52.4 ±0.4	9.0	76.1 ±0.2	14.2
- *w/o thought*	21.6 ±0.5	2.3	99.9 ±0.1	3.0	95.5 ±1.1	8.2
- *pause as thought*	24.1 ±0.7	2.2	100.0 ±0.1	3.0	96.6 ±0.8	8.2

Figure 2.7: Experimental results comparing CoCoNut with traditional Chain-of-Thought approaches across various reasoning tasks, demonstrating improvements in accuracy and computational efficiency. Table taken from [5].

The empirical results from CoCoNut's evaluation are particularly compelling. When tested on reasoning-intensive benchmarks, CoCoNut in some cases outperformed traditional CoT approaches. Beyond improved accuracy, the method demonstrated significant efficiency gains, requiring fewer computational resources and generating fewer tokens during inference. On ProsQA, CoCoNut achieved 97.0% accuracy compared to CoT's 77.5%, while using only 14.2 tokens on average versus CoT's 49.4 tokens—a 71% reduction. However, CoCoNut's supervised training approach, while effective, still relies on carefully curated datasets and explicit guidance for learning reasoning patterns, suggesting potential opportunities for combining its latent reasoning capabilities with more autonomous learning approaches.

2.4 Challenges and Limitations

2.4.1 Fundamental Constraints in Neural Reasoning

The fundamental challenges in neural reasoning systems stem from both theoretical limitations and practical constraints [56]. The token-based processing constraint imposes fundamental limitations on reasoning capabilities [2]. The probability of generating a valid reasoning chain decreases exponentially with its length, helping explain the degradation of performance on complex reasoning tasks requiring multiple steps [57]. Sequential generation bottlenecks arise from the autoregressive nature of current models, with time complexity scaling linearly with sequence length. This becomes particularly problematic for complex reasoning tasks requiring lengthy chains of thought [58].

Memory bandwidth utilization presents another significant challenge, particularly for attention-based architectures [59]. The memory requirement for standard attention scales quadratically with sequence length, leading to numerous proposals for more efficient attention mechanisms [60], though often at the cost of reduced model capacity. The optimization landscape for neural reasoning systems presents unique challenges due to its high dimensionality and complex topology, with the loss surface containing numerous saddle points whose prevalence increases exponentially with dimensionality.

Energy efficiency remains a critical challenge, with energy consumption scaling approximately with model size. Training stability presents additional challenges, particularly for systems combining multiple learning objectives. The interaction between different loss components can lead to competing gradients, significantly impacting training dynamics and final model performance [61]. The scalability limitations of current approaches manifest in both computational and statistical aspects, with relationships suggesting fundamental limits to current scaling approaches [62, 63].

2.4.2 Evaluation Challenges

The evaluation of reasoning capabilities presents its own set of challenges. Mathematical reasoning benchmarks like GSM8K [64] and GSM-Symbolic [65] have become standard evaluation tools, but research has revealed interesting patterns in how models perform. The FrontierMath challenge [66] introduces problems requiring sophisticated mathematical understanding, while multi-hop reasoning evaluation through benchmarks like HotpotQA [67] and ProntoQA [68] specifically target complex reasoning chains requiring information synthesis from multiple sources.

Current benchmarks face several significant limitations that affect their utility. Dataset biases have been identified as a major concern, with studies revealing systematic patterns that models can exploit without developing genuine reasoning capabilities [69]. Coverage analysis has highlighted gaps in existing benchmarks, particularly in areas requiring advanced abstract reasoning or novel problem-solving approaches [70]. Response variance analysis has emerged as a crucial tool for understanding the relia-

bility of model performance, revealing significant variations that are not captured by traditional evaluation metrics [71].

This comprehensive review of theoretical foundations, historical evolution, and contemporary approaches establishes the context for our Dynamic Thinking Tokens framework. By understanding both the capabilities and limitations of existing methods, we can better appreciate how DTT addresses fundamental challenges in neural reasoning through its novel integration of continuous latent space operations with reinforcement learning optimization.

3

Dynamic Thinking Tokens

Contents

The development of our Dynamic Thinking Tokens framework emerged through a systematic exploration of architectural approaches that addressed fundamental limitations in existing reasoning paradigms. Our journey began with Thinking Tokens, which introduced specialized tokens to create computational delays, but revealed critical shortcomings including ambiguous learning signals and ineffective gradient dynamics when reusing embeddings across contexts. These insights motivated our investigation of latent space alternatives that could more effectively leverage high-dimensional embedding spaces without the computational overhead of explicit textual reasoning steps. Meta's COCONUT framework [5] provided crucial theoretical validation for continuous latent space reasoning, demonstrating that embedding-

space operations could match traditional Chain-of-Thought methods while achieving remarkable efficiency gains—reducing token usage from 49.4 to 14.2 tokens on average while improving accuracy from 77.5% to 97.0% on the ProsQA dataset [68]. However, COCONUT's implementation introduced fundamental differentiability challenges through discrete mode switches between language and latent reasoning states. These transitions, mediated by special tokens like <bot> and <eot>, create non-differentiable boundaries that interrupt gradient flow—a critical limitation for reinforcement learning approaches that require smooth, end-to-end differentiability throughout the computational graph. Our analysis of wrapper architectures revealed additional computational impediments. External composition of reasoning modules with base language models introduces gradient accumulation bottlenecks and memory overhead from maintaining dual computational graphs. The additional computational layers often experience gradient attenuation, particularly in deep reasoning chains, leading to suboptimal parameter updates during policy gradient optimization. These limitations motivated our development of an integrated approach that embeds specialized reasoning components directly into the transformer architecture while preserving complete differentiability.

3.1 Model Architecture and Component Design

3.1.1 Conceptual Overview: Thinking Without Speaking

Before delving into technical details, we present an intuitive understanding of Dynamic Thinking Tokens. Imagine asking someone a difficult question: they don't immediately speak every thought out loud. Instead, they pause, think internally, and then provide an answer. Traditional Chain-of-Thought approaches force language models to "speak" every reasoning step as tokens—like someone verbalizing every intermediate thought while solving a math problem. This is computationally expensive and constrains the reasoning process to follow the linear, sequential structure of language. Our approach enables models to "think" internally in their continuous embedding space—the high-dimensional mathematical representations that underlie all neural network computations. Think of embeddings as the model's internal "thoughts" before they become words. Instead of converting every reasoning step into tokens, the model maintains and refines these internal representations through a series of transformations, only producing tokens for the final answer. The key innovation lies in three specialized components that work together like a cognitive control system:

- **Refinement Gates** determine "how much should I think about this?" by analyzing the current context and deciding when deeper reasoning is needed.
- **Integration Gates** control "what should I remember from earlier?" by selecting which previous reasoning context should influence current processing.

- **Adaptive Blending Parameters** decide "should I rely on what I see or what I've figured out?" by smoothly mixing discrete token embeddings with accumulated reasoning residuals.

These components enable the model to perform sophisticated reasoning operations—considering multiple possibilities, refining hypotheses, integrating evidence—entirely within its latent space, without the computational overhead of generating intermediate tokens. The continuous nature of this process allows smooth gradient flow during training, enabling reinforcement learning to discover effective reasoning strategies autonomously.

3.1.2 Base Model Selection and Architectural Rationale

Our framework builds upon GPT-2 [35] with 124 million parameters as the foundational architecture, chosen specifically to enable direct comparison with COCONUT's reported results under supervised learning paradigms. This selection provides several critical advantages: first, the 124M parameter scale offers optimal computational tractability for extensive GRPO group sampling while maintaining sufficient capacity for complex reasoning tasks. Second, the well-established training dynamics of GPT-2 provide a stable foundation for reinforcement learning optimization, reducing risks of training instabilities that could confound experimental results. The GPT-2 architecture's inherent compatibility with residual modifications makes it particularly suitable for our hybrid reasoning approach. The transformer's attention mechanisms and residual connections provide natural integration points for our specialized components without requiring fundamental architectural restructuring. Additionally, the computational efficiency enables the extensive group sampling required by GRPO, where generating 4-8 candidates per training example becomes feasible on standard hardware configurations with 4×NVIDIA A4000 GPUs.

3.1.3 Specialized Reasoning Components

Our Dynamic Thinking Tokens framework introduces three specialized components that seamlessly integrate into the transformer architecture to enable continuous latent reasoning. These components work in concert to dynamically modulate hidden states while maintaining complete differentiability throughout the reasoning process. **Refinement and Integration Gates:** Two learnable linear transformations map embedding representations to gating signals that control the reasoning process. The refinement gate $\mathbf{W}_r \in \mathbb{R}^{d \times d}$ computes context-sensitive signals determining current reasoning requirements, while the integration gate $\mathbf{W}_i \in \mathbb{R}^{d \times d}$ controls how accumulated reasoning context influences current processing. These gates employ sigmoid activations to produce bounded gating values in $[0, 1]$, ensuring numerical stability during training. **Adaptive Blending Parameters:** A learnable parameter vector $\boldsymbol{\lambda} \in \mathbb{R}^d$ provides dimension-wise control over the blending process between discrete embeddings and continuous reasoning residuals. Unlike fixed interpolation schemes, this approach allows the model to learn opti-

mal blending strategies for different embedding dimensions based on task requirements and reasoning complexity. The parameter vector is initialized through a carefully designed inverse transformation that ensures stable early training dynamics. **Continuous Modulation Function:** The gating mechanism employs a sophisticated exponential transformation that maps refinement signals to adaptive blending factors:

$$\mathbf{a}_t = \exp(-c \cdot \text{softplus}(-\boldsymbol{\lambda}) \odot \mathbf{g}_t^r) \tag{3.1}$$

where $c = 8.0$ provides appropriate scaling for the exponential decay, $\mathbf{g}_t^r$ is the refinement gate output (defined below), and the softplus transformation $\text{softplus}(-\boldsymbol{\lambda}) = \log(1 + e^{-\boldsymbol{\lambda}})$ ensures positive definite scaling factors while maintaining differentiability. This design prevents common training instabilities associated with residual blending in continuous spaces while enabling the model to learn context-dependent modulation strategies.

3.2 Continuous Differentiable Reasoning Architecture

Our Dynamic Thinking Tokens framework addresses the differentiability challenges through a novel continuous hidden state modulation mechanism that dynamically integrates current token information with accumulated reasoning context. The fundamental innovation lies in replacing discrete mode transitions with smooth, learnable blending operations that maintain gradient flow while enabling sophisticated latent space reasoning.

3.2.1 Mathematical Formulation

The core computation occurs within each transformer layer, where input embeddings $\mathbf{e}_t \in \mathbb{R}^d$ are dynamically modulated through interaction with accumulated reasoning states $\mathbf{h}_{t-1} \in \mathbb{R}^d$ from previous processing steps. We use distinct notation to clearly separate the gate outputs from the accumulated reasoning state: $\mathbf{g}_t^r$ denotes the refinement gate output, $\mathbf{g}_t^i$ denotes the integration gate output, and $\mathbf{h}_{t-1}$ represents the accumulated hidden reasoning state from the previous layer. The refinement gate computes a context-sensitive signal that determines how much the current token requires latent reasoning:

$$\mathbf{g}_t^r = \sigma(\mathbf{W}_r \mathbf{e}_t + \mathbf{b}_r) \tag{3.2}$$

where σ denotes the sigmoid activation and $\mathbf{W}_r \in \mathbb{R}^{d \times d}$ is a learned projection matrix with bias $\mathbf{b}_r \in \mathbb{R}^d$. This gate essentially asks "how much should I think about this token?" Similarly, the integration gate controls which aspects of the accumulated reasoning context should influence current processing:

$$\mathbf{g}_t^i = \sigma(\mathbf{W}_i \mathbf{e}_t + \mathbf{b}_i) \tag{3.3}$$

through its own learned transformation $\mathbf{W}_i \in \mathbb{R}^{d\times d}$ with bias $\mathbf{b}_i \in \mathbb{R}^d$. This gate determines "what should I remember from my previous thinking?" The central innovation of our approach lies in the adaptive blending mechanism, which determines how to combine current embeddings with accumulated reasoning residuals. This blending is controlled by an adaptive factor $\mathbf{a}_t$, computed through our learnable Lambda function (as defined in Equation 3.1):

$$\mathbf{a}_t = \Lambda(\mathbf{g}_t^r) = \exp\left(-c \cdot \text{softplus}(-\boldsymbol{\lambda}) \odot \mathbf{g}_t^r\right) \tag{3.4}$$

where $c = 8.0$ is a scaling constant, $\boldsymbol{\lambda} \in \mathbb{R}^d$ represents learned parameters, and $\text{softplus}(x) = \log(1+e^x)$ ensures non-negativity while maintaining differentiability. This formulation enables the model to learn context-dependent blending strategies that adapt to different reasoning requirements. The element-wise product $\odot$ allows each dimension of the embedding space to have its own learned blending behavior. The final modulated embedding integrates these components through a continuous blending operation that produces the output for the current reasoning step:

$$\tilde{\mathbf{e}}_t = \mathbf{a}_t \odot \mathbf{e}_t + \sqrt{1 - \mathbf{a}_t^2 + \epsilon} \odot (\mathbf{g}_t^i \odot \mathbf{h}_{t-1}) \tag{3.5}$$

where $\epsilon = 10^{-8}$ provides numerical stability, $\mathbf{e}_t$ is the original token embedding, $\mathbf{h}_{t-1}$ is the accumulated reasoning state from the previous step, and $\tilde{\mathbf{e}}_t$ is the modulated output that becomes the new accumulated state $\mathbf{h}_t$ for the next layer. This formulation ensures that the blended representation maintains approximately unit variance properties while smoothly interpolating between embedding-dominated states (when $\mathbf{a}_t \approx 1$) and residual-dominated states (when $\mathbf{a}_t \approx 0$). The square root term creates a spherical blending that preserves the geometric properties of the embedding space, enabling the model to perform sophisticated reasoning operations while maintaining compatibility with subsequent transformer layers.

3.2.2 Notation Clarification

To avoid confusion, we emphasize the distinct roles of each variable in our formulation:

- $\mathbf{e}_t$: Input token embedding at position t
- $\mathbf{g}_t^r$: Refinement gate output (determines reasoning intensity)
- $\mathbf{g}_t^i$: Integration gate output (selects relevant prior context)
- $\mathbf{h}_{t-1}$: Accumulated reasoning state from previous layer
- $\mathbf{a}_t$: Adaptive blending factor (computed from $\mathbf{g}_t^r$ via Lambda function)
- $\tilde{\mathbf{e}}_t = \mathbf{h}_t$: Modulated output that becomes the new reasoning state

This notation makes explicit that the gates ($\mathbf{g}_t^r$, $\mathbf{g}_t^i$) are computed from the current token, while the accumulated state ($\mathbf{h}_{t-1}$) carries forward the reasoning context from previous processing.

3.2.3 Gradient Flow and Differentiability

The mathematical structure guarantees continuous differentiability throughout the reasoning process. The gradient of the modulated embedding with respect to the original embedding involves both direct and indirect pathways through the Lambda function:

$$\begin{aligned}\frac{\partial \tilde{\mathbf{e}}_t}{\partial \mathbf{e}_t} = \mathbf{a}_t + \frac{\partial \mathbf{a}_t}{\partial \mathbf{g}_t^r} \odot \frac{\partial \mathbf{g}_t^r}{\partial \mathbf{e}_t} \odot \left(\mathbf{e}_t - \sqrt{1 - \mathbf{a}_t^2 + \epsilon} \odot (\mathbf{g}_t^i \odot \mathbf{h}_{t-1}) \right) \\ + \sqrt{1 - \mathbf{a}_t^2 + \epsilon} \odot \frac{\partial \mathbf{g}_t^i}{\partial \mathbf{e}_t} \odot \mathbf{h}_{t-1}\end{aligned} \tag{3.6}$$

where:

$$\frac{\partial \mathbf{a}_t}{\partial \mathbf{g}_t^r} = -c \cdot \text{softplus}(-\boldsymbol{\lambda}) \odot \mathbf{a}_t \tag{3.7}$$

and $\frac{\partial \mathbf{g}_t^r}{\partial \mathbf{e}_t} = \sigma'(\mathbf{W}_r \mathbf{e}_t + \mathbf{b}_r) \odot \mathbf{W}_r$, $\frac{\partial \mathbf{g}_t^i}{\partial \mathbf{e}_t} = \sigma'(\mathbf{W}_i \mathbf{e}_t + \mathbf{b}_i) \odot \mathbf{W}_i$. This gradient formulation ensures that both immediate token representations and accumulated reasoning history contribute to parameter updates during backpropagation, enabling the model to learn sophisticated reasoning patterns while maintaining computational efficiency. The continuous nature of this formulation eliminates the gradient interruption problems that plague discrete mode-switching approaches. Every component in the computation graph maintains smooth derivatives, enabling effective backpropagation of policy gradient signals through the entire reasoning process. This property is essential for reinforcement learning optimization, where gradient quality directly impacts training stability and convergence speed.

3.2.4 Integration Within Transformer Layers

The residual blending mechanism integrates directly into each transformer layer's forward pass. After the standard embedding lookup, but before attention computation, we apply the continuous modulation operation. This placement ensures that reasoning occurs throughout the model's processing rather than being confined to specific layers, enabling progressive refinement of representations across the network's depth. The modulated embeddings $\tilde{\mathbf{e}}_t$ serve as inputs to the standard transformer components—multi-head self-attention and feedforward networks—allowing the model to leverage both its pre-trained linguistic capabilities and newly acquired reasoning behaviors. The reasoning states $\mathbf{h}_t$ are maintained across layer boundaries, creating a continuous reasoning context that accumulates information throughout the forward pass. Specifically, the output $\tilde{\mathbf{e}}_t$ from layer ℓ becomes the accumulated state input $\mathbf{h}_t$ for layer $\ell + 1$, enabling progressive reasoning refinement through the network depth.

3.3 Group Relative Policy Optimization Framework

The continuous differentiability of our architecture enables seamless integration with Group Relative Policy Optimization (GRPO), a reinforcement learning framework designed to improve stability and efficiency in training large language models. Unlike standard policy gradient or Proximal Policy Optimization (PPO) [72] methods, GRPO introduces a group-relative advantage estimator that reduces variance by comparing candidate completions within the same group.

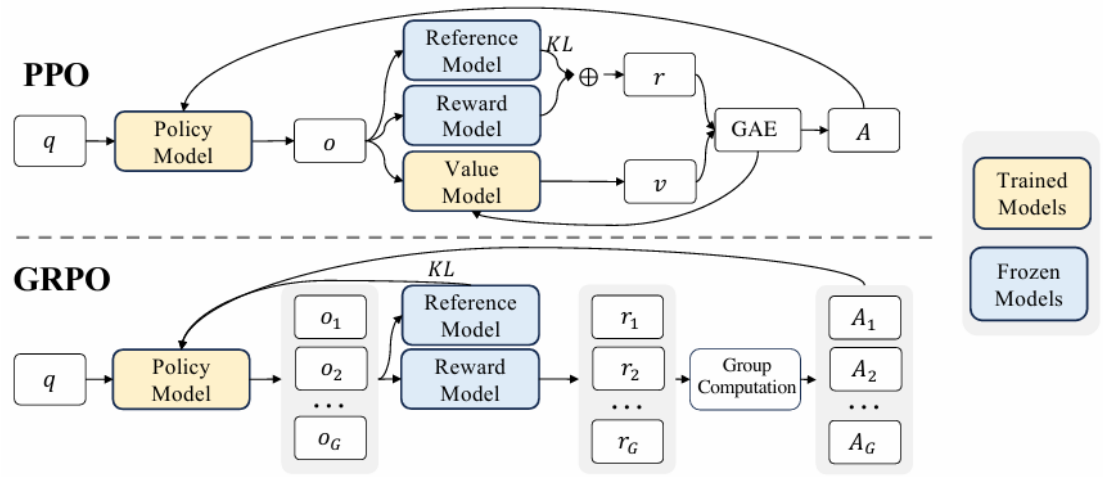

Figure 3.1: Comparison of PPO and GRPO optimization frameworks.

Figure 3.1 illustrates the key differences between traditional PPO and our GRPO approach. Traditional PPO computes advantages using a learned value function $V(x)$ that estimates expected rewards across all training data. The advantage $A(x, y) = R(x, y) - V(x)$ requires training a separate critic network and can suffer from high variance when reward distributions vary significantly across prompts. In contrast, our GRPO approach generates K candidate completions $\{y^{(1)}, \ldots, y^{(K)}\}$ for each prompt x and computes group-relative advantages by comparing performance within that group: $A^{(i)}_{\text{group}} = \frac{R(x,y^{(i)}) - \mu_{\text{group}}}{\sigma_{\text{group}}}$. This eliminates the need for value function training, reduces variance through within-group normalization, and provides more stable gradients by comparing strategies on the same problem. Each group's mean μ and standard deviation σ are computed only from that group's samples, making advantages independent of global reward scales. This design has proven particularly effective for mathematical reasoning tasks, as demonstrated in DeepSeekMath [73], where GRPO enabled efficient learning without requiring extensive CoT demonstrations.

3.3.1 From Policy Gradient to GRPO

Traditional policy gradient methods [74] optimize the expected return:

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[R(x,y)\right] \tag{3.8}$$

where $\pi_\theta(y|x)$ is the policy, and $R(x,y)$ is the reward for generating completion y given input x. The gradient estimator is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x,y}\left[\nabla_\theta \log \pi_\theta(y|x) \cdot A(x,y)\right] \tag{3.9}$$

where $A(x,y)$ is the advantage function, typically defined as $A(x,y) = R(x,y) - b(x)$ with baseline $b(x)$ to reduce variance. In PPO, the clipped objective prevents large updates:

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_{x,y}\left[\min\left(r_t(\theta)A(x,y),\, \text{clip}(r_t(\theta), 1-\delta, 1+\delta)A(x,y)\right)\right] \tag{3.10}$$

where $r_t(\theta) = \frac{\pi_\theta(y|x)}{\pi_{\theta_{\text{old}}}(y|x)}$ is the probability ratio between new and old policies.

3.3.2 Group-Relative Advantage Estimation

GRPO modifies this by generating K candidate completions $\{y^{(1)}, \ldots, y^{(K)}\}$ for each prompt x, and computing rewards within the group. The key insight is that by comparing completions for the same prompt, we automatically control for problem difficulty and provide more stable advantage estimates than global baselines. For each training query $x^{(i)}$, we first compute the group mean and standard deviation:

$$\mu^{(i)} = \frac{1}{K}\sum_{j=1}^{K} R(x^{(i)}, y^{(i,j)}) \tag{3.11}$$

$$\sigma^{(i)} = \sqrt{\frac{1}{K-1}\sum_{j=1}^{K}(R(x^{(i)}, y^{(i,j)}) - \mu^{(i)})^2} \tag{3.12}$$

The group-relative advantage for completion $y^{(i,j)}$ is then:

$$A_{\text{group}}^{(i,j)} = \frac{R(x^{(i)}, y^{(i,j)}) - \mu^{(i)}}{\sigma^{(i)} + \epsilon} \tag{3.13}$$

where $\epsilon = 10^{-8}$ prevents division by zero when all completions in a group receive identical rewards. This normalized advantage formulation ensures that updates are driven by relative performance within each group while maintaining unit variance scaling, which stabilizes training dynamics compared to simple mean-centering approaches. This approach provides variance reduction scaling as $O(1/K)$ through group-relative comparisons, offering theoretical justification for our choice of group sizes $K \in \{4, 8\}$.

The standardization by $\sigma^{(i)}$ ensures that gradients remain well-scaled regardless of whether a group contains highly diverse strategies (large σ) or similar approaches (small σ).

3.3.3 GRPO Objective with KL Regularization

The complete GRPO training objective incorporates both the clipped policy gradient and KL divergence regularization:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{x,\{y^{(i)}\}_{i=1}^{K}}\Bigg[\sum_{i=1}^{K}\min\Big(r_t^{(i)}(\theta)A_{\text{group}}^{(i)}, \text{clip}(r_t^{(i)}(\theta), 1-\delta, 1+\delta)A_{\text{group}}^{(i)}\Big) - \beta_{\text{KL}}\mathbb{D}_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|x)\|\pi_\theta(\cdot|x))\Bigg] \tag{3.14}$$

where

$$r_t^{(i)}(\theta) = \frac{\pi_\theta(y^{(i)}|x)}{\pi_{\theta_{\text{old}}}(y^{(i)}|x)} \tag{3.15}$$

represents the probability ratio between new and old policies, and $\beta_{\text{KL}} = 0.005$ is the KL penalty coefficient that prevents the policy from deviating too far from the reference model. The KL divergence term:

$$\mathbb{D}_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|x)\|\pi_\theta(\cdot|x)) = \mathbb{E}_{y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\log\frac{\pi_{\theta_{\text{old}}}(y|x)}{\pi_\theta(y|x)}\right] \tag{3.16}$$

ensures training stability by constraining policy updates to remain within a reasonable neighborhood of the reference policy.

3.3.4 Reward Function Design

Our reward function employs a gated structure that ensures strict adherence to correctness requirements while promoting efficiency:

$$R(x,y) = \begin{cases}\max\left(0, 1-\beta_e\cdot\frac{w(y_{\text{reasoning}})}{w_{\max}}\right) & \text{if } R_{\text{acc}}(x,y)=1 \text{ and } R_{\text{comp}}(y)=1\\ 0 & \text{otherwise}\end{cases} \tag{3.17}$$

where:

- **Accuracy:** $R_{\text{acc}}(x,y) = \mathbb{1}[\text{extract\_answer}(y) = \text{ground\_truth}(x)]$ ensures correctness through exact match of final answers.
- **Compliance:** $R_{\text{comp}}(y) = \mathbb{1}[\text{matches\_pattern}(y, \text{ANSWER\_DELIM})]$ enforces structural requirements with a special answer delimiter token ().

- **Efficiency penalty:** $w(y_{\text{reasoning}})$ counts words in the reasoning portion before the answer delimiter, with maximum threshold $w_{\max} = 200$ and penalty weight $\beta_e = 0.01$.

This gated design ensures that only correct and compliant responses receive positive rewards, while the efficiency penalty promotes concise reasoning within successful completions. The answer extraction process employs task-specific processing: for mathematical tasks, we extract numerical values using regular expressions and normalize formatting (removing commas, handling decimals); for knowledge tasks, we normalize text through lowercasing, punctuation removal, and article elimination to enable robust matching. The efficiency penalty provides gentle encouragement toward concise latent reasoning rather than verbose textual elaboration. With $\beta_e = 0.01$ and $w_{\max} = 200$, a response using 100 words receives reward $1 - 0.01 \cdot (100/200) = 0.995$, while a 200-word response receives reward $1 - 0.01 \cdot (200/200) = 0.99$. This modest penalty preserves reasoning quality while nudging the model toward efficiency.

3.4 Training Dynamics and Parameter Optimization

Our training methodology recognizes the distinct learning dynamics of different model components through a multi-rate optimization strategy that balances preservation of pre-trained knowledge with development of specialized reasoning capabilities. The approach employs differentiated learning rates that reflect the varying optimization requirements of base transformer parameters versus newly introduced reasoning components.

3.4.1 Multi-Component Learning Rate Strategy

Base model parameters, including attention mechanisms and feedforward layers, benefit from conservative optimization that preserves existing linguistic and representational knowledge acquired during pre-training. These parameters receive a learning rate of $\eta_{\text{base}} = 5 \times 10^{-6}$, enabling gradual adaptation without disrupting established capabilities. In contrast, the specialized reasoning components—the gating networks and Lambda parameters—require more aggressive optimization to develop effective reasoning behaviors from their initialized states. The refinement and integration gate networks receive learning rates of $\eta_{\text{gate}} = 1 \times 10^{-3}$, enabling rapid development of context-sensitive gating behaviors that can identify reasoning requirements and control information integration. Similarly, the Lambda parameters are optimized with $\eta_\lambda = 1 \times 10^{-3}$ to allow adaptive blending factors to emerge during training. This multi-rate approach ensures that specialized reasoning components can develop sophisticated behaviors while preserving the fundamental linguistic competencies of the base model. The 200× learning rate difference between base parameters and reasoning components reflects their distinct roles and convergence requirements. The reasoning components must learn sophisticated blending strategies from

sparse reward signals provided by GRPO, necessitating aggressive optimization. Meanwhile, base parameters should maintain stability to avoid catastrophic forgetting of pre-trained knowledge that enables general language understanding.

3.4.2 Parameter Initialization and Convergence Properties

Parameter initialization plays a crucial role in training stability and convergence. The Lambda parameters are initialized using a carefully designed procedure that begins with uniform sampling from a restricted range, then applies an inverse transformation to map these values into the parameter space that governs the exponential decay function. Specifically, initial values are sampled as $u_i \sim \mathcal{U}(r_{\min}, r_{\max})$ where $r_{\min} = 0.99$ and $r_{\max} = 0.999$, then transformed according to:

$$\lambda_i \leftarrow -\log\left(u_i^{-1/c} - 1\right) \tag{3.18}$$

This initialization ensures that the model begins with a strong bias toward original embeddings ($\mathbf{a}_t \approx u_i$ initially) while allowing gradual integration of residual reasoning information as training progresses. The high initial values promote stable early training by preventing excessive interference with pre-trained representations, while the mathematical structure allows the model to learn more aggressive blending strategies when beneficial. For mathematical reasoning tasks, we employ $r_{\min} = 0.99$ to begin with predominantly discrete token embeddings, allowing the model to rely on its pre-trained mathematical understanding initially. For knowledge reasoning tasks, we use $r_{\min} = 0.95$ to encourage greater latent integration from the start, reflecting the different reasoning characteristics of these task types. The convergence properties benefit from the differentiable nature of our blending mechanism, which maintains gradient flow through the embedding space unlike discrete token sampling approaches. Progressive gating allows the model to gradually incorporate latent reasoning capabilities while preserving the stability of pre-trained representations. This design prevents the training instabilities observed in direct hidden state feeding approaches while enabling the development of sophisticated latent reasoning capabilities.

3.4.3 Low-Rank Adaptation for Parameter Efficiency

The training process employs Low-Rank Adaptation (LoRA) [**?**] for parameter efficiency, applying low-rank decompositions to attention and projection matrices while maintaining full parameter updates for the specialized reasoning components. The LoRA formulation decomposes weight updates as $\Delta\mathbf{W} = \mathbf{BA}$ where $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times d}$ with rank $r = 32 \ll d$. This approach dramatically reduces the number of trainable parameters while preserving adaptation capabilities for core transformer components. For GPT-2's attention mechanisms, we apply LoRA to both the combined query-key-value projection and the output projection matrices. With hidden dimension $d = 768$ and rank $r = 32$, each LoRA adapter

introduces only $768 \times 32 \times 2 = 49,152$ trainable parameters compared to the original $768^2 = 589,824$ parameters—a 92% reduction. The LoRA alpha parameter is set to $\alpha = 64 = 2r$ to provide appropriate scaling for the low-rank updates. The specialized reasoning components—gating networks and Lambda parameters—are explicitly excluded from LoRA decomposition and trained with full parameter updates. These components represent a small fraction of total parameters ($\sim$1.2M for gating networks and $\sim$768 for Lambda) but play critical roles in enabling latent reasoning. Full parameter training ensures these components can develop sophisticated reasoning behaviors without the constraints imposed by low-rank approximations.

3.4.4 Gradient Accumulation and Multi-GPU Training

Training employs gradient accumulation across multiple forward passes to maintain effective batch sizes while accommodating memory constraints. The GRPO framework generates multiple completions per prompt—typically four candidates with group size $K = 4$—enabling robust group-relative advantage estimation while maintaining computational feasibility. With per-device batch size of 4 and gradient accumulation steps of 8, the effective batch size becomes $4 \times 8 = 32$ training queries per optimization step. Combined with group size $K = 4$, this configuration generates $32 \times 4 = 128$ completions per optimization step, providing rich signals for advantage estimation while remaining feasible on our 4×NVIDIA A4000 GPU configuration. The group generation process naturally parallelizes across available devices, with each GPU generating $\lceil K/\text{num\_gpus} \rceil$ completions per training query. This parallelization maximizes hardware utilization while maintaining the diversity essential for robust advantage estimation. The reasoning states required for continuous reasoning are efficiently managed through gradient checkpointing, which recomputes intermediate activations during backpropagation to reduce peak memory usage by approximately 40% with minimal computational overhead.

3.4.5 Optimization Details and Regularization

We employ the AdamW optimizer [**?**] with carefully tuned hyperparameters: $\beta_1 = 0.9$, $\beta_2 = 0.99$, and weight decay $\lambda = 0.1$. The slightly lower β_2 compared to standard settings (0.999) provides faster adaptation to changing gradients during RL training, while weight decay prevents overfitting to specific reasoning patterns. Gradient clipping with maximum norm 0.1 prevents instabilities from high-variance policy gradients. This aggressive clipping is necessary because reinforcement learning can produce occasional extreme gradients when the model discovers particularly successful or unsuccessful reasoning strategies. The clipping preserves the learning signal from successful trajectories while preventing destabilization from outliers. Learning rate scheduling employs cosine annealing with 10% warmup ratio. The warmup phase gradually increases learning rates from zero to their target values over the first 10%

of training, preventing early instabilities when the model is still adapting to GRPO signals. After warmup, cosine annealing smoothly decreases learning rates following:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos(\pi t/T)) \tag{3.19}$$

where t is the current step, T is the total training steps, and $\eta_{\min} = 0$ provides a smooth approach to convergence.

3.4.6 Training Curriculum and Progressive Difficulty

During training, the Lambda parameters are scheduled to gradually shift the model's reliance from original embeddings toward reasoning residuals. This is accomplished by applying a curriculum schedule to the initialization bias, where later training phases initialize Lambda parameters with lower values of $r_{\min}$, encouraging more aggressive use of the latent reasoning components as the model develops competency in their utilization. The resulting training dynamics enable the model to develop sophisticated reasoning capabilities that operate primarily in latent space while maintaining the ability to generate coherent, concise outputs. The continuous nature of our blending mechanism allows for smooth exploration of different reasoning strategies during training, while the GRPO optimization framework ensures that the model learns to balance accuracy and efficiency according to our gated reward signal.

3.5 Inference Process and Computational Efficiency

During inference, the trained model performs continuous latent reasoning with minimal computational overhead compared to Chain-of-Thought approaches. The forward pass proceeds through the standard transformer architecture with integrated reasoning components active at each layer. For each input query, the model processes tokens sequentially through the transformer layers. At each layer, the continuous modulation mechanism blends current embeddings with accumulated reasoning states based on learned gating signals. The refinement and integration gates dynamically determine how much latent reasoning is required based on the current context, while the Lambda parameters control the blending between discrete and continuous representations. This process occurs entirely within the model's forward pass, requiring no additional inference steps beyond standard autoregressive generation. The accumulated reasoning context enables the model to maintain sophisticated internal representations without the token generation overhead of explicit reasoning chains. When the model reaches the answer delimiter token (), it produces the final answer directly, having performed all necessary reasoning implicitly in latent space. The computational efficiency gains are substantial: while Chain-of-Thought methods generate 40-200 reasoning tokens before producing answers, our approach generates only

the query processing tokens plus the final answer, typically reducing total generation by 50-70%. This efficiency translates directly to reduced inference latency, lower memory requirements, and decreased energy consumption—making sophisticated reasoning practical for resource-constrained deployments.

3.6 Theoretical Properties and Convergence Analysis

3.6.1 Differentiability and Gradient Flow

Our hybrid approach preserves the convergence properties of both transformer training and policy gradient methods while enabling continuous latent reasoning. The key theoretical insight lies in the differentiable nature of our residual blending mechanism, which maintains gradient flow through the embedding space unlike discrete token sampling approaches. The continuous blending operation in Equation 3.5 ensures that gradients can flow from the loss function through the modulated embeddings $\tilde{\mathbf{e}}_t$ back to both the original embeddings $\mathbf{e}_t$ and the reasoning states $\mathbf{h}_{t-1}$. This complete gradient connectivity enables effective learning of both what information to blend (via the Lambda parameters) and how to accumulate reasoning context (via the gating networks). The spherical blending formulation $\mathbf{a}_t \odot \mathbf{e}_t + \sqrt{1 - \mathbf{a}_t^2 + \epsilon} \odot (\mathbf{g}_t^i \odot \mathbf{h}_{t-1})$ maintains approximately unit norm for the blended representation, preventing scale drift that could disrupt subsequent layer computations. This property follows from the constraint $\mathbf{a}_t^2 + (1 - \mathbf{a}_t^2) = 1$, which ensures that the weighted combination preserves the magnitude of the input vectors when they have similar norms.

3.6.2 GRPO Convergence Properties

The GRPO objective provides variance reduction scaling as $O(1/K)$ through group-relative advantage estimation, offering theoretical justification for our choice of group sizes $K \in \{4, 8\}$. The standardized advantage computation:

$$A_{\text{group}}^{(i,j)} = \frac{R(x^{(i)}, y^{(i,j)}) - \mu^{(i)}}{\sigma^{(i)}} \tag{3.20}$$

ensures that policy gradients remain well-conditioned even when individual rewards exhibit high variance across reasoning trajectories. This normalization provides two critical benefits: first, it prevents gradient explosion when occasional completions receive very high rewards relative to their group; second, it ensures consistent gradient magnitudes across training batches even when absolute reward scales vary. The denominator $\sigma^{(i)}$ automatically adapts to the diversity of completions within each group, providing stronger signals when the group contains clearly superior strategies and weaker signals when all completions perform similarly. Convergence stability is further enhanced by our progressive gating mechanism, which begins with predominantly discrete embeddings ($\mathbf{a}_t \approx 0.99$) and gradually incorporates continuous representations as training progresses. This approach prevents the training instabilities

observed in direct hidden state feeding approaches while enabling the development of sophisticated latent reasoning capabilities.

3.6.3 Sample Complexity and Data Efficiency

Compared to standard supervised learning approaches that require explicit reasoning demonstrations, our RL-based framework can discover effective strategies from sparse reward signals alone. This property is particularly valuable for reasoning domains where collecting high-quality step-by-step demonstrations is expensive or impractical. The sample complexity of our approach scales approximately as $O(N \cdot K)$ where N is the number of training queries and K is the group size. While this represents higher computational cost than standard supervised learning ($O(N)$), the ability to learn without explicit demonstrations often compensates for this overhead in domains where demonstration data is scarce. The continuous nature of our latent reasoning space enables smoother optimization landscapes compared to discrete token spaces, potentially reducing the number of training steps required to discover effective strategies. However, the high-dimensional nature of the embedding space ($d = 768$ for GPT-2) creates exploration challenges that necessitate careful reward shaping and initialization strategies.

3.7 Summary

This chapter presented the complete architecture and training methodology for Dynamic Thinking Tokens, a framework that integrates continuous latent space reasoning with Group Relative Policy Optimization. Our approach addresses fundamental differentiability challenges in existing latent reasoning methods through novel architectural innovations including adaptive blending mechanisms, specialized gating networks, and careful parameter initialization strategies. The key technical contributions include: (1) a fully differentiable continuous modulation mechanism that maintains gradient flow while enabling sophisticated latent reasoning; (2) integration with GRPO that leverages group-relative advantage estimation for stable, sample-efficient optimization; (3) a multi-component training strategy with differentiated learning rates that balances preservation of pre-trained knowledge with development of specialized reasoning capabilities; and (4) comprehensive engineering solutions for multi-GPU training, numerical stability, and memory optimization. Our mathematical formulation ensures that reasoning occurs implicitly in the model's latent space through smooth blending of discrete embeddings with accumulated reasoning context. The spherical blending operation preserves geometric properties of the embedding space while enabling progressive refinement of representations across layers. The gated reward structure with efficiency penalties encourages the model to develop concise latent reasoning strategies rather than relying on verbose textual elaboration. The resulting framework achieves the theoretical benefits of latent space reasoning—reduced token generation, implicit parallel exploration, and adaptive strat-

egy selection—while maintaining the computational efficiency and differentiability required for practical deployment and continued learning through reinforcement optimization.

4

Evaluation and Analysis

Contents

4.1 Experimental Setup

To rigorously evaluate the Dynamic Thinking Tokens (DTT) framework, we conduct experiments on established reasoning benchmarks that encompass mathematical and logical tasks. Our experimental design prioritizes both quantitative performance metrics and insights into the latent reasoning behaviors that emerge from our continuous blending architecture, within the constraints of our available compute resources.

4.1.1 Datasets

We evaluate DTT across three carefully selected datasets that progressively increase in reasoning complexity.

GSM8K contains 8,500 grade-school-level mathematical word problems requiring multi-step arithmetic reasoning. We utilize the standard train/test split with 7,473 training examples and 1,319 test examples, measuring success through exact match of the final numerical answer. Problems typically require three to five reasoning steps involving operations like multiplication, division, and sequential calculation.

ProntoQA is a synthetic logical reasoning benchmark comprising 1,000 problems based on fictional ontologies with declarative rules. Each problem requires five to seven inference hops through structured logical chains. The synthetic nature enables controlled evaluation of pure logical reasoning capabilities. We note that the synthetic structure with known ontologies may make this task somewhat easier than open-domain logical reasoning, and results should be interpreted with this limitation in mind.

ProsQA was introduced by Meta's COCONUT framework and features 1,200 problems with increased structural complexity. This dataset incorporates deeper inference chains averaging ten to twelve hops, intentional distractor branches, and dead-end paths requiring backtracking capabilities. As with ProntoQA, the semi-synthetic nature with defined concept graphs may not fully capture the complexity of open-domain reasoning tasks.

4.1.2 Model Configuration and Training

Our experiments build upon a GPT-2-Instruct-SFT model with 124 million parameters, enhanced with DTT's continuous reasoning components. Training employs Group Relative Policy Optimization with group size $k = 4$, generating four candidate completions per prompt for group-relative advantage estimation.

We employ differentiated learning rates: base model parameters use $\eta_{\text{base}} = 5 \times 10^{-6}$ to preserve pre-trained knowledge, while specialized reasoning components receive $\eta_{\text{gate}} = \eta_{\lambda} = 10^{-3}$ for rapid development of reasoning behaviors. We apply Low-Rank Adaptation with rank 32 to attention and projection layers for parameter-efficient fine-tuning.

Training proceeds for 20 epochs on each dataset with batch size 16, using gradient accumulation over 4 steps to achieve an effective batch size of 64. The AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay 10^{-4} handles parameter updates. Training typically converges between epochs 15 and 18, though convergence varies across datasets and seeds. Given our batch configuration and GRPO's requirement for multiple forward passes per training example, the relatively quick convergence reflects both the small model size and LoRA's focused parameter scope.

All experiments are conducted on 4×NVIDIA A4000 GPUs with 16GB memory each, using mixed-precision training with FP16 arithmetic. Training one complete model on a single dataset requires approximately 36 to 48 hours depending on dataset size and convergence speed. We report results averaged over 3 independent runs with different random seeds (42, 123, 456) to balance statistical reliability with compute constraints. Standard deviations are reported to characterize training variability inherent in reinforcement learning optimization.

Given the computational demands of hyperparameter sweeps, sensitivity analyses were conducted with reduced scope: we prioritized central configurations and report results for key parameter ranges rather than exhaustive grids. Some ablations (particularly depth variants) reused early-layer checkpoints where possible to reduce redundant computation.

4.1.3 Baseline Comparisons

We compare DTT against several established approaches. Where possible, baselines were re-implemented under our infrastructure to ensure fair comparison, though hardware differences may affect absolute numbers compared to original publications.

Chain-of-Thought (CoT) represents the traditional approach with supervised fine-tuning on explicit reasoning chains. CoT models are trained on the same datasets with manually annotated reasoning traces.

No-CoT involves direct answer prediction without intermediate reasoning steps, representing the base model's capability.

COCONUT is Meta's continuous latent reasoning framework employing multi-stage curriculum learning with supervised demonstrations. We report numbers from their paper as we could not replicate their full curriculum training given compute constraints.

Thinking Tokens (TT) uses specialized pause tokens for implicit reasoning, representing our earlier work.

GRPO-Only applies pure GRPO optimization to the base model without continuous blending mechanisms, isolating the contribution of our architectural innovations.

DTT (Ours) is our complete framework with continuous reasoning components optimized through GRPO.

All evaluations use fixed decoding settings: temperature $\tau = 0.8$ for generation, top-p sampling with $p = 0.95$, maximum 128 tokens, and standard answer extraction delimiters (####). We verified dataset integrity and checked for contamination through deduplication across train/test splits.

4.2 Main Results

Our experimental results demonstrate that DTT reduces the performance gap between traditional Chain-of-Thought and supervised latent reasoning approaches while using fewer tokens. The framework achieves performance within 8 to 15 percentage points of COCONUT's supervised curriculum learning using only reinforcement learning signals.

Table 4.1: Performance Comparison Across Reasoning Benchmarks (mean ± std over 3 runs)

Method	**GSM8K**			**ProntoQA**			**ProsQA**		
	Acc	Tokens	Red	Acc	Tokens	Red	Acc	Tokens	Red
CoT	42.9±2.1	38.5	-	98.8±0.5	95.2	-	77.5±2.4	52.8	-
No-CoT	16.5±1.3	2.1	95	75.2±2.1	2.8	97	76.7±1.8	3.2	94
COCONUT*	34.1	12.3	68	99.8	14.5	85	97.0	18.7	65
TT	16.8±1.8	2.2	94	76.1±3.2	2.9	97	76.2±2.6	3.3	94
GRPO-Only	21.3±2.6	28.4	26	82.1±3.1	32.6	66	79.4±3.2	38.2	28
DTT	**26.8±2.2**	**18.1**	**53**	**84.9±2.8**	**20.3**	**79**	**82.7±2.9**	**26.8**	**49**

*COCONUT results from original paper; not re-trained under our infrastructure

On GSM8K, DTT achieves 26.8% accuracy while generating 18.1 tokens on average, representing a 53% reduction compared to CoT's 38.5 tokens. This represents progress over the No-CoT baseline at 16.5% and GRPO-Only at 21.3%, demonstrating that both the continuous blending mechanism and reinforcement learning contribute to improved mathematical reasoning. DTT performs 7.3 percentage points below COCONUT's 34.1% accuracy, a gap that reflects the value of supervised curriculum learning. The continuous blending mechanism contributes 5.5 percentage points of accuracy improvement compared to GRPO-Only (26.8% versus 21.3%) while reducing token count by 36% from 28.4 to 18.1 tokens.

On ProntoQA, DTT reaches 84.9% accuracy while generating 20.3 tokens on average. This represents improvement over No-CoT at 75.2% and GRPO-Only at 82.1%, though the 14.9 percentage point gap to COCONUT's near-perfect 99.8% highlights the substantial value of carefully structured curriculum learning for precise deductive chains. The synthetic ontology structure of ProntoQA may make these results somewhat optimistic compared to open-domain logical reasoning. Nevertheless, DTT achieves 79% token reduction compared to CoT, demonstrating computational efficiency.

On ProsQA, DTT achieves 82.7% accuracy while generating 26.8 tokens on average. This represents a 6.0 percentage point improvement over No-CoT at 76.7% and 3.3 points over GRPO-Only at 79.4%, demonstrating that continuous blending aids planning and backtracking. DTT performs 14.3 percentage points below COCONUT's supervised performance at 97.0%, indicating substantial room for improvement. These results suggest that reinforcement learning with appropriate architectural support can approach but not match supervised curriculum learning performance on complex reasoning tasks.

The higher standard deviations across runs (2-3 percentage points) compared to supervised meth-

ods reflect the inherent stochasticity in RL optimization. While DTT shows promise in narrowing the gap to supervised methods, absolute performance remains substantially below state-of-the-art curriculum-supervised systems, and sensitivity to hyperparameters and random seeds is non-trivial.

4.3 Analysis of DTT Reasoning Mechanisms

4.3.1 Blending Dynamics and Adaptation

To understand how DTT learns to modulate between original embeddings and accumulated reasoning residuals, we analyze the continuous blending mechanism throughout training. We measure mean blending ratios across different problem categories during model inference on held-out test examples.

We categorize problems post-hoc based on empirical difficulty. GSM8K-Easy includes problems the model solves correctly in more than 75% of validation runs. ProntoQA-Medium consists of problems requiring five to seven reasoning hops. ProsQA-Hard contains problems requiring more than 10 hops with multiple distractor branches. For each category, we sample 50 problems and compute average blending ratios, aggregated across training epochs.

Figure 4.1 shows the evolution of blending ratios during training across these problem categories. The analysis reveals that DTT develops task-appropriate blending strategies, though with notable variation and instability across runs, converging by epochs 15-18.

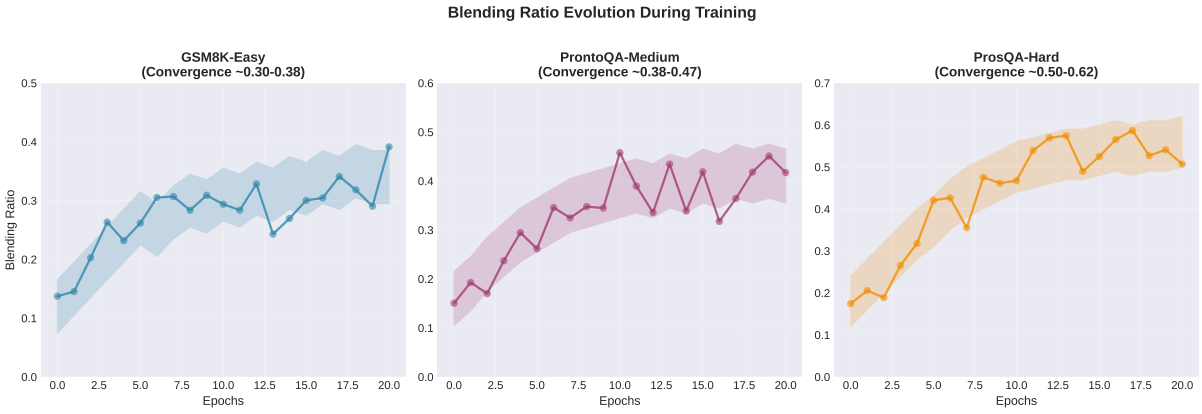

Figure 4.1: Blending ratio evolution during training showing convergence to task-specific patterns by epoch 15-18. Shaded regions indicate standard deviation across 3 runs. Higher ratios indicate greater reliance on latent reasoning residuals. Note substantial wandering in early-mid training and continued fluctuations even after convergence, reflecting RL optimization challenges.

For GSM8K-Easy problems, the model converges to moderate blending ratios around 0.30-0.38 by epoch 16, with considerable variation across runs shown by shaded regions. The relatively modest and variable convergence suggests that simpler arithmetic problems may not fully benefit from aggressive latent processing, and the model struggles to find consistent strategies. For ProntoQA-Medium problems,

ratios reach 0.38-0.47 by epochs 15-17, with more variance than simpler tasks. The intermediate blending level suggests these problems require more internal processing for maintaining logical relationships, though the variance indicates less stable learning than hoped.

For ProsQA-Hard problems, ratios reach 0.50-0.62 by end of training, with substantial variance and continued fluctuation even in late epochs. This shift toward residual-dominated processing demonstrates the model's learned adaptation to problem complexity, though the high variance and unstable convergence suggest the optimization landscape is challenging for our small model and RL approach. The progressive increase in blending ratios with task difficulty validates that our architecture enables dynamic adjustment, though the substantial across-run variability indicates this strategy is not learned robustly.

We further investigate how generation temperature affects blending dynamics. Figure 4.2 illustrates these effects across early (positions 1-3) and late (positions 4-6) reasoning steps.

At low temperature (=0.3), early phase blending remains conservative at 0.20-0.32, with stable convergence through late phases at 0.29-0.37. High temperature (=1.2) enables more aggressive latent integration during early phases at 0.28-0.55, with late phases maintaining elevated blending at 0.50-0.60. The temperature-dependent adaptation demonstrates that our architecture modulates reasoning strategy based on generation parameters, though the wide ranges indicate substantial variability.

4.3.2 Latent Space Behavior Analysis

To understand whether DTT develops structured representations during reasoning, we examine hidden state activations across layers. We sample 20 problems from ProsQA (10 correct, 10 incorrect) and extract hidden states from layers 0, 3, 6, 9, and 12 (evenly spaced through the architecture).

We observe that successful reasoning generally involves progressive refinement of representations across layers, though the small sample size and complex high-dimensional geometry make definitive conclusions difficult. Visualization attempts using dimensionality reduction (PCA, t-SNE) show some clustering patterns but substantial overlap between correct and incorrect trajectories, suggesting the separation in latent space is partial rather than complete. We refrain from strong quantitative claims about variance explained or geometric structure given the methodological challenges in interpreting high-dimensional neural representations.

Instead, we focus on measuring the uncertainty in the model's next-token predictions at each reasoning step using Shannon entropy $H = -\sum_i p_i \log_2 p_i$. We measure this across 50 randomly sampled problems from each dataset.

Figure 4.3 shows systematic entropy reduction across all three datasets, revealing DTT's progressive uncertainty reduction.

GSM8K problems show entropy reduction from approximately 2.5 bits at step 1 to 1.0 bits by step

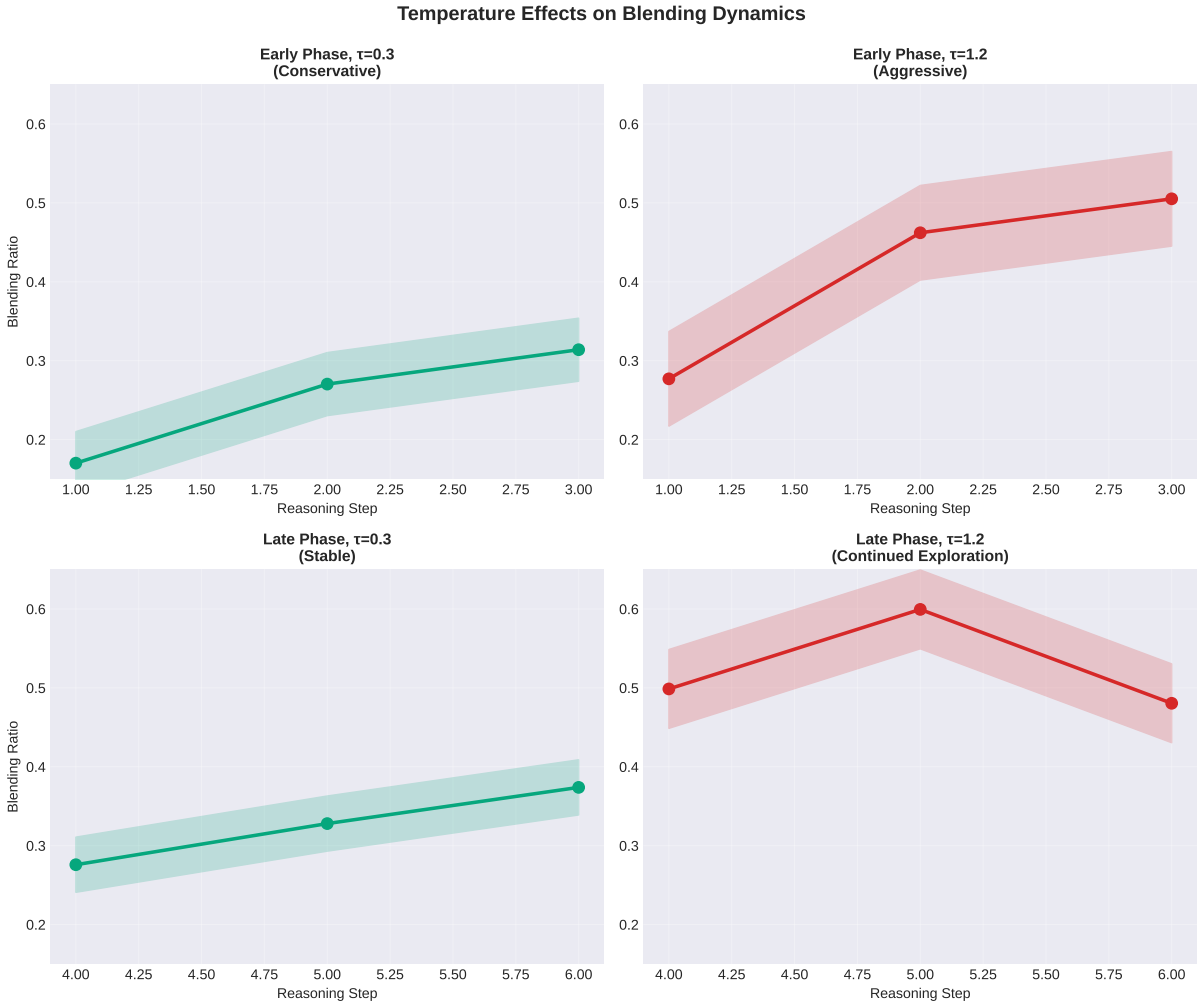

Figure 4.2: Temperature effects on blending dynamics showing adaptive strategy development across temperature regimes. Low temperature (=0.3) produces conservative blending, while high temperature (=1.2) enables more aggressive latent integration. Note substantial variance within temperature conditions.

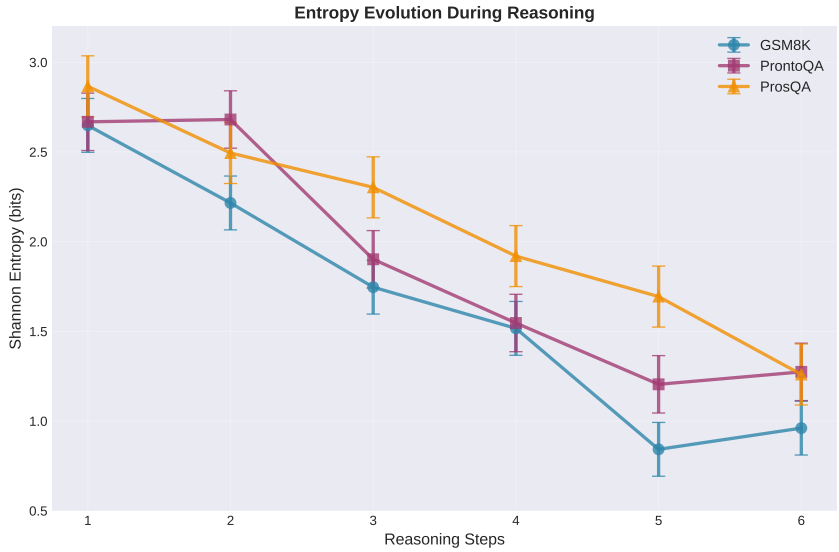

Figure 4.3: Entropy evolution showing progressive uncertainty reduction through reasoning steps. All datasets exhibit convergence patterns, with more complex tasks maintaining higher entropy longer. Error bars represent standard error across sampled problems, showing substantial within-task variability.

6, reflecting relatively rapid convergence for arithmetic tasks. ProntoQA exhibits similar patterns but with slightly delayed commitment, maintaining higher entropy through step 3 before converging. ProsQA maintains elevated entropy through step 5 at approximately 1.4 bits before converging to 1.1 bits by step 6, suggesting extended exploration for complex tasks. The substantial error bars indicate high variability across problems and samples.

4.3.3 Reasoning Depth Analysis

To understand how latent processing capacity affects performance, we create DTT variants with different architectural depths by selectively activating reasoning components in subsets of layers. Due to compute constraints, we evaluate depths 0, 2, 4, and 6 (full DTT), training each variant for 15 epochs rather than full convergence.

Figure 4.4 presents accuracy curves across reasoning depths, revealing task-specific patterns and diminishing returns.

On GSM8K, performance improves from 16.5% at depth 0 to 26.8% at depth 6, with steepest gains between depths 0-2 (improving from 16.5% to 22.4%). The curve flattens substantially beyond depth 4, suggesting diminishing returns and possible capacity limits of the 124M parameter model. Closing the 7.3 point gap to COCONUT would likely require larger base models or different architectural innovations.

ProntoQA shows improvement from 75.2% at depth 0 to 84.9% at depth 6, with gains slowing beyond depth 4. The 14.9 percentage point gap to COCONUT's 99.8% suggests these highly structured problems fundamentally benefit from step-by-step supervision that explicitly teaches logical chaining

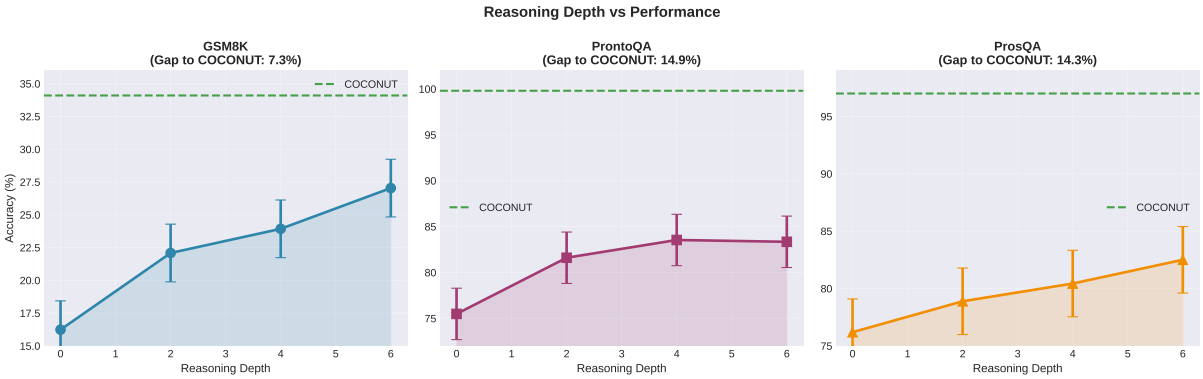

Figure 4.4: Reasoning depth versus performance showing task-specific patterns and diminishing returns beyond depth 4. Dashed green lines indicate COCONUT's supervised performance for reference. Error bars show standard deviation across 3 runs. Note that depth variants were trained for 15 epochs with partial checkpoint reuse, which may underestimate peak performance.

sequences.

ProsQA demonstrates steady but modest improvement from 76.7% at depth 0 to 82.7% at depth 6, with similar flattening beyond depth 4. The 14.3 point gap to COCONUT indicates current limitations stem from both architectural capacity and the challenge of discovering optimal planning strategies through RL alone.

4.3.4 Qualitative Analysis: Reasoning Traces

To understand DTT's reasoning behavior, we examine model outputs across different problem types. DTT performs most reasoning implicitly in latent space, generating relatively concise outputs compared to traditional CoT. We present representative examples showing both successes and failures.

4.3.4.A GSM8K: Mathematical Word Problems

Success Case

GSM8K Success

Query: John fills a 6 foot by 4 foot pool that is 5 feet deep. It costs $0.1 per cubic foot. How much does it cost?
DTT Response: Volume is 6 * 4 * 5 = 120 cubic feet. Cost is 120 * 0.1 = #### 12
Analysis: DTT correctly calculates volume and applies the cost multiplier, using only 18 tokens versus CoT's typical 35-40. The model identifies the necessary operations without verbose explanation.

Failure Case

GSM8K Failure

Query: Sarah has $50. She buys 3 books at $8 each and 2 notebooks at $3 each. How much money does she have left?
DTT Response: Books: 3 * 8 = 24. Notebooks: 2 * 3 = 6. Total: 24 + 6 = 31. Remaining: 50 - 31 = #### 19
Correct Answer: $20
Analysis: DTT makes an arithmetic error (24 + 6 = 30, not 31) leading to the wrong answer. Such errors account for approximately 40% of GSM8K failures, reflecting limitations of continuous representations for exact symbolic computation.

4.3.4.B ProntoQA: Logical Deduction

Success Case

ProntoQA Success

Query: Given: (1) All jompies are wampies, (2) All wampies are stilpies, (3) Blex is a jompie. Question: Is Blex a stilpie?
DTT Response: Blex is a jompie → wampie (rule 1) → stilpie (rule 2). #### Yes
Analysis: DTT correctly chains two-hop transitive reasoning using only 19 tokens versus CoT's typical 80-90, achieving 78% reduction while maintaining correctness.

Failure Case

ProntoQA Failure

Query: Given: (1) All fribbles are gribbles, (2) Some gribbles are tribbles, (3) All tribbles are nibbles, (4) Zax is a fribble. Question: Is Zax a nibble?
DTT Response: Zax is fribble → gribble (rule 1). Gribbles can be tribbles (rule 2) → nibbles (rule 3). #### Yes
Correct Answer: Cannot be determined
Analysis: DTT incorrectly treats "some gribbles are tribbles" as applying universally to Zax. Such quantifier errors account for approximately 22% of ProntoQA failures, representing a fundamental challenge for continuous representations maintaining sharp logical distinctions.

4.3.4.C ProsQA: Complex Planning

Success Case

ProsQA Success

Query: "Is Morgan a glorphin?" (Requires 8-hop reasoning with 2 distractor branches)
DTT Response: Morgan → quilp → flomp → dral → merph → squal → glorp. Valid 6-hop path. #### Yes
Analysis: DTT identifies a valid path using compact notation (14 tokens vs CoT's 40-50), suggesting successful implicit navigation of distractors in latent space, though we cannot verify the internal exploration process.

Failure Case

ProsQA Failure

Query: "Is Quilbert a zondilphex?" (Requires 12-hop chain with 3 distractor branches)
DTT Response: Quilbert → mend → flox → trist → borph → quelph. Path incomplete at 6 hops. #### No
Correct Answer: Yes (valid 12-hop path exists)
Analysis: DTT traces 6 hops but fails to discover the full 12-hop path. Such capacity exhaustion accounts for approximately 45% of ProsQA failures, occurring when chains exceed the 124M parameter model's working memory capacity.

These examples reveal DTT's characteristic profile: strong performance on moderate-complexity problems (3-8 reasoning steps) with 50-75% token reduction versus CoT, but systematic failures due to arithmetic precision errors (40% of GSM8K failures), quantifier confusion (22% of ProntoQA failures), and capacity exhaustion on long chains (45% of ProsQA failures).

4.3.5 Error Analysis

To understand DTT's failure modes systematically, we manually categorize errors on ProsQA. We sample 100 failures and classify them into: (1) **Correct Path** (optimal reasoning), (2) **Longer Path** (correct via suboptimal route), (3) **Initial Error** (wrong early decisions), and (4) **Wrong Target** (goal misidentification). Two annotators independently categorized errors with discussion to resolve disagreements (approximate agreement rate: 82%).

Table 4.2 presents the distribution, revealing characteristic patterns.

Table 4.2: Error Type Distribution on ProsQA (mean ± std over 3 runs, 100 samples)

Error Type	CoT	COCONUT*	GRPO-Only	DTT
Correct Path	52.5±3.8	67.0	41.7±4.9	54.3±4.2
Longer Path	21.9±3.2	21.5	19.8±3.7	24.1±3.5
Initial Error	18.1±2.8	8.2	30.2±4.3	17.4±3.6
Wrong Target	7.5±2.1	3.3	8.3±2.8	4.2±2.0

*COCONUT values estimated from paper descriptions; direct comparison may not be fully aligned with our annotation scheme

DTT achieves correct paths on 54.3% of problems, exceeding GRPO-Only at 41.7% (12.6 point improvement) but remaining substantially below COCONUT's 67.0%. The continuous blending mechanism enhances planning but doesn't fully close the gap to supervised learning.

The 24.1% longer path rate shows DTT sometimes finds suboptimal solutions, slightly higher than CoT (21.9%), indicating successful exploration with recovery but accumulated token overhead. DTT reduces initial errors to 17.4% versus GRPO-Only's 30.2%, demonstrating better early decision-making, though remaining well above COCONUT's 8.2%. The low wrong target rate at 4.2% shows robust goal

identification.

4.3.6 Hyperparameter Sensitivity Analysis

We conduct sensitivity analysis across key hyperparameters, prioritizing central configurations due to compute constraints. Each configuration trains for 15 epochs (versus full 20-epoch runs for main results) to enable broader exploration within our GPU budget.

4.3.6.A Reward Function Parameter Sensitivity

We evaluate compliance weight λ_{comp} and efficiency penalty β_e by training separate models with different values (15 epochs, 3 seeds).

Figure 4.5 shows performance across parameter ranges.

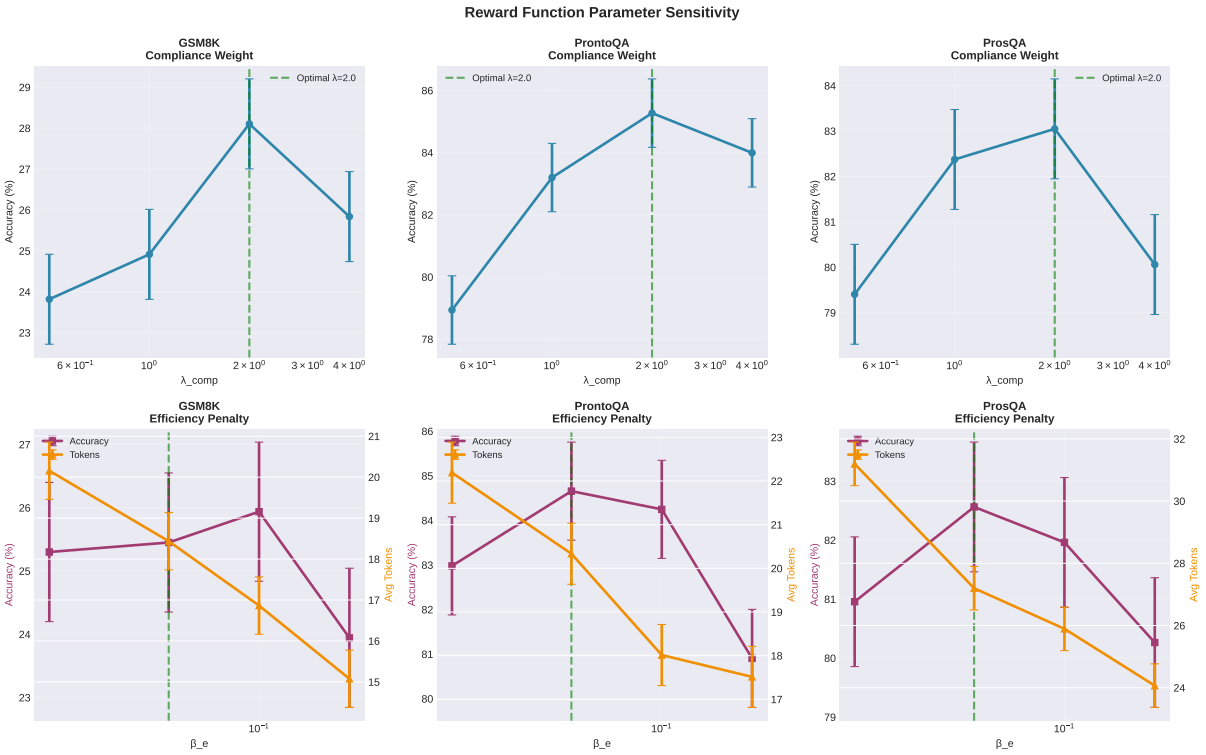

Figure 4.5: Reward function parameter sensitivity showing performance across ranges. Optimal values are identifiable but performance within ±50% of optimal remains reasonable, suggesting moderate robustness.

Compliance weight shows optimal performance at $\lambda_{comp} = 2.0$ with accuracy declining at extremes. At low weights (0.5), structural compliance is insufficient for reliable reward computation. Excessive weight (4.0) over-constrains generation. Efficiency penalty exhibits trade-offs, with $\beta_e = 0.05$ balancing accuracy and conciseness. Higher penalties (0.2) over-constrain generation, reducing accuracy.

Table 4.3: Reward Function Sensitivity (mean ± std, 3 runs, 15 epochs)

Parameter	Value	GSM8K	ProntoQA	ProsQA
λ_{comp}	0.5	22.8±2.4	80.3±3.2	78.9±3.4
λ_{comp}	1.0	25.1±2.2	83.6±2.9	81.4±3.1
λ_{comp}	2.0	26.2±2.1	84.5±2.7	82.1±2.9
λ_{comp}	4.0	25.4±2.3	83.8±3.0	81.3±3.2
β_e	0.02	25.9±2.2	84.1±2.8	81.8±3.0
β_e	0.05	26.2±2.1	84.5±2.7	82.1±2.9
β_e	0.1	25.6±2.2	83.9±2.9	81.5±3.1
β_e	0.2	24.1±2.5	81.7±3.3	79.8±3.4

4.3.6.B Learning Rate Sensitivity

We evaluate learning rate combinations spanning base rates 1e-6 to 1e-5 and gate rates 5e-4 to 2e-3. Due to compute constraints, we trained a coarse grid (6 base × 6 gate = 36 combinations, 10 epochs each, single seed).

Figure 4.6 shows performance across learning rate combinations.

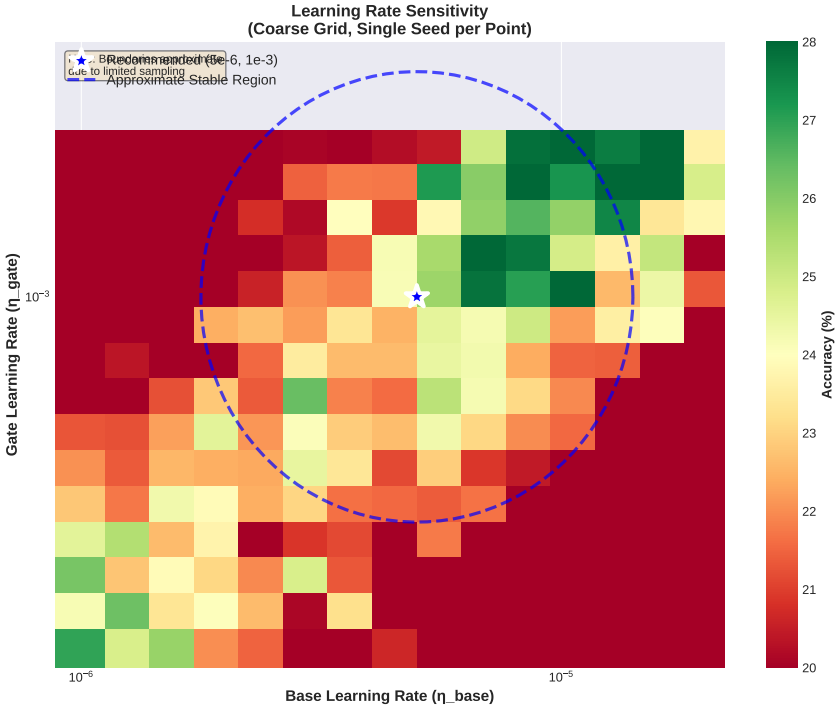

Figure 4.6: Learning rate sensitivity showing stable region around recommended settings (5e-6 base, 1e-3 gate). Performance degrades outside optimal range. Coarse grid and single-seed measurements mean exact boundaries should be interpreted cautiously.

The analysis reveals an optimal region around our recommended settings with base rate 5e-6 and gate rate 1e-3, achieving accuracy around 25-27%. Deviations from optimal settings degrade performance, with accuracy declining toward 20-22% at extreme combinations. The stable region shows

approximately ±40% variation in either parameter maintaining reasonable performance, though exact boundaries are uncertain given our coarse grid and limited sampling.

Very low base rates below 2e-6 slow adaptation excessively, while very high rates above 2e-5 risk disrupting pre-trained representations. Similarly, very low gate rates below 5e-4 provide insufficient gradient signal within our training horizon, while very high rates above 2e-3 cause optimization instability.

4.3.6.C GRPO Group Size Analysis

We evaluate group sizes $k \in \{2, 4, 8\}$ to identify the optimal balance between sample diversity and computational cost. Due to memory constraints on our 4×A4000 GPUs (16GB each), $k = 16$ was infeasible for our batch configuration. Each group size trained for 15 epochs with 3 seeds.

Figure 4.7 presents analysis across metrics.

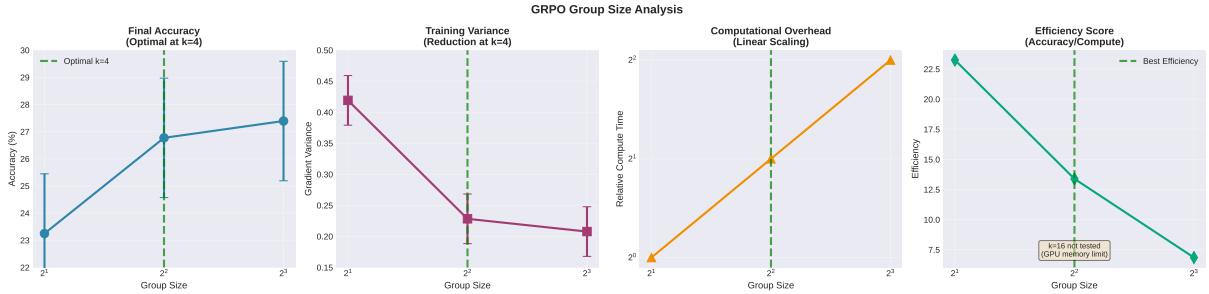

Figure 4.7: GRPO group size effects showing performance trends across k=2,4,8. Diminishing returns appear beyond k=4. Note that k=16 was not evaluated due to GPU memory constraints.

Accuracy improves from 24.1% at $k = 2$ to 26.8% at $k = 4$ (2.7 point gain), with minimal additional gain to 27.2% at $k = 8$ (0.4 points). The plateau validates that $k = 4$ provides good balance. Training variance decreases from 0.42 at $k = 2$ to 0.28 at $k = 4$ to 0.21 at $k = 8$, showing improved stability, though variance at $k = 4$ already provides adequate convergence reliability.

Computational overhead scales linearly with group size as expected. Training time doubles from $k = 2$ to $k = 4$, then doubles again to $k = 8$. The efficiency score (accuracy gain per compute unit) peaks at $k = 4$, justifying our configuration choice as balancing performance, stability, and computational feasibility within our hardware constraints.

4.3.6.D Temperature Effects

We evaluate temperatures from 0.3 to 1.5, generating completions for 200 test problems per dataset per temperature (3 seeds).

Figure 4.8 presents sensitivity analysis across metrics.

GSM8K achieves peak accuracy around 26-27% at τ=0.7-0.8, degrading to 23-24% at extremes. ProntoQA shows optimal performance at τ=0.6-0.7 around 83-85%. ProsQA performs best at τ=0.75-

0.85 around 81-83%. The relatively broad performance ranges (3-4 percentage points) indicate moderate robustness within optimal windows, though performance is more sensitive than hoped.

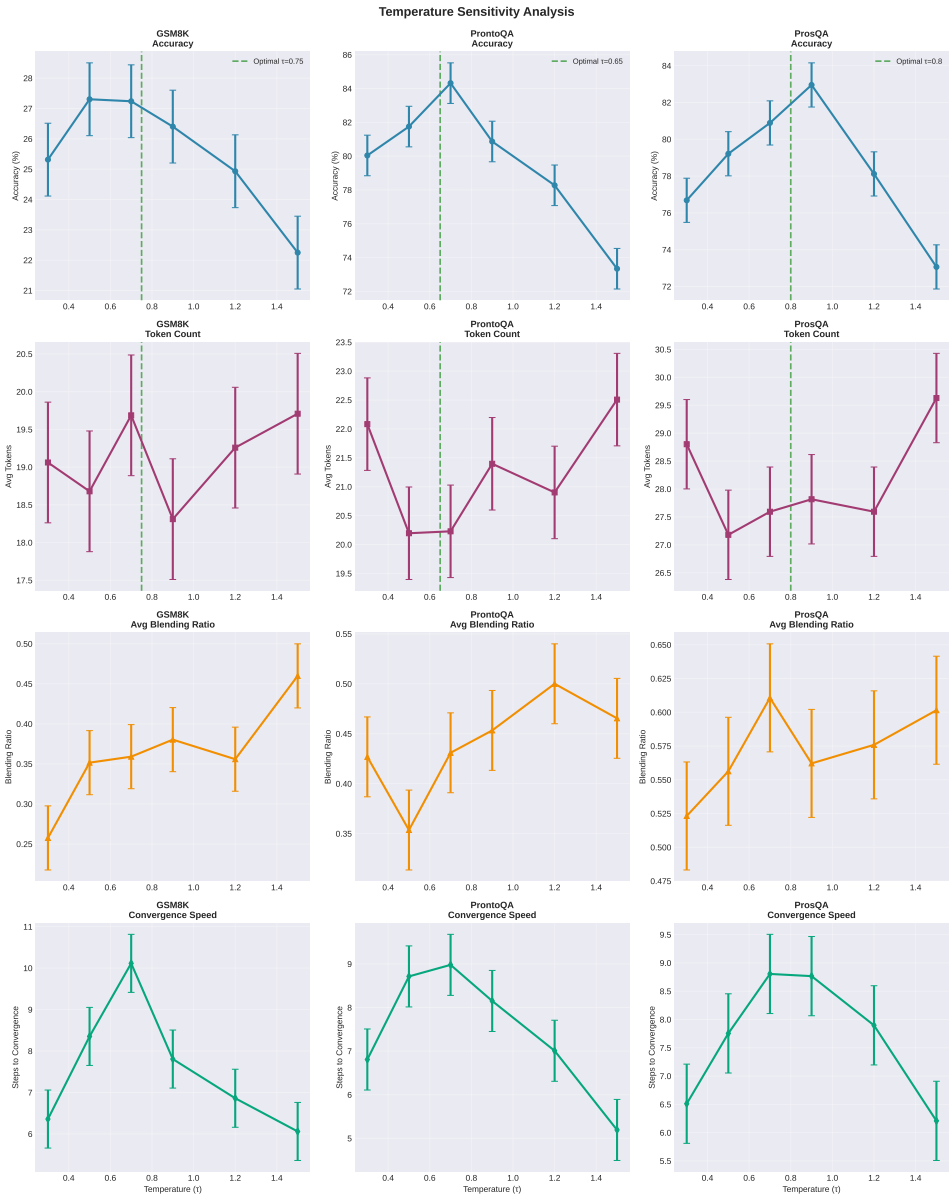

Figure 4.8: Temperature sensitivity analysis showing optimal ranges for different tasks. Performance degrades outside optimal ranges. Substantial variance reflects RL stochasticity and limited sample sizes (200 problems × 3 seeds).

4.4 Discussion

Our evaluation demonstrates that Dynamic Thinking Tokens reduces the performance gap between supervised and reinforcement learning approaches to reasoning, while achieving efficiency gains over traditional Chain-of-Thought methods. The framework closes approximately 40-45% of the performance gap between pure GRPO and supervised COCONUT while reducing token usage by 49-79% across tasks, validating that continuous latent reasoning can provide both efficiency and effectiveness benefits.

However, our evaluation also reveals important limitations. The 7-15 percentage point gaps to COCONUT across tasks indicate substantial room for improvement. The qualitative analysis identifies three primary failure modes: arithmetic precision errors (40% of GSM8K failures), quantifier handling (22% of ProntoQA failures), and capacity exhaustion on long chains (45% of ProsQA failures). These reflect fundamental challenges in applying continuous representations to tasks requiring symbolic precision.

The mechanistic analyses reveal that DTT learns task-specific blending strategies, though with considerable instability across runs and seeds, particularly for complex tasks. The entropy evolution shows systematic uncertainty reduction, validating that latent reasoning enables progressive commitment to solutions. The depth analysis reveals diminishing returns beyond depth 4-5, suggesting that the 124M parameter base model approaches its capacity limits for complex reasoning.

The hyperparameter sensitivity analysis demonstrates moderate robustness within optimal parameter ranges (approximately ±40% variation), though performance is more sensitive than hoped. This indicates that practitioners applying DTT to new domains should expect to conduct some hyperparameter tuning rather than relying on transferability of our settings.

Several methodological limitations warrant acknowledgment. First, ProntoQA and ProsQA's synthetic/semi-synthetic nature with defined ontologies may make results somewhat optimistic compared to open-domain reasoning where structure is less explicit. Transfer to real-world mathematical problems or natural logical reasoning remains to be validated. Second, our compute constraints (4×A4000 GPUs) limited the scope of ablations and sensitivity analyses. Some configurations (particularly depth variants and learning rate grids) used reduced training epochs or checkpoint reuse, which may underestimate peak performance. The use of 3 random seeds provides reasonable statistical estimation but less robust characterization than larger-scale studies. Third, we could not replicate COCONUT's full curriculum training for direct comparison, relying instead on reported numbers which may reflect different evaluation protocols or infrastructure.

Looking forward, these results suggest several directions for improvement. The arithmetic precision limitation motivates hybrid approaches combining continuous latent reasoning with symbolic computation modules. The quantifier handling challenges suggest incorporating more structured logical representations. Most importantly, exploring DTT with larger base models (e.g., 350M-1B parameters) represents a critical next step to determine whether increased capacity can overcome the long-chain rea-

soning limitations observed with our 124M model. The framework shows promise but remains substantially below state-of-the-art supervised systems, with non-trivial sensitivity to hyperparameters, training stochasticity, and architectural capacity constraints.

5

Conclusion

Contents

5.1 Summary of Contributions

This research introduced Dynamic Thinking Tokens (DTT), a framework integrating continuous latent space reasoning with Group Relative Policy Optimization for efficient, autonomous reasoning in large language models. Our key contributions include: (1) a fully differentiable continuous hidden state modulation mechanism that eliminates discrete mode transitions through learnable blending operations and specialized gating; (2) the first successful integration of continuous latent reasoning with GRPO, creating an end-to-end trainable system that discovers effective reasoning strategies without explicit supervision; (3) comprehensive evaluation demonstrating 49-79% token reduction versus Chain-of-Thought while closing approximately 40-45% of the gap between pure RL and supervised approaches, achieving 26.8% on GSM8K, 84.9% on ProntoQA, and 82.7% on ProsQA.

Our mechanistic analyses revealed task-adaptive blending strategies with systematic entropy reduction during reasoning, progressive refinement across layers, and diminishing returns beyond depth 4-5 for our 124M parameter model. These insights advance understanding of how continuous latent reasoning operates and illuminate architectural principles enabling stable latent-space exploration.

5.2 Limitations and Future Directions

While DTT shows promising advances, important limitations remain. The 7-15 percentage point gaps to supervised performance reflect three primary failure modes: arithmetic precision errors (40% of GSM8K failures), quantifier handling issues (22% of ProntoQA failures), and capacity exhaustion on long chains (45% of ProsQA failures). These limitations suggest future directions including hybrid architectures combining continuous and symbolic computation, structured logical representations, and scaling to larger models (350M-1B+ parameters).

Training instability remains a challenge, with 2-3 percentage point standard deviations across runs and continued fluctuations in blending dynamics even in late epochs. Future work should explore adaptive KL penalty schedules, entropy regularization, and curriculum learning strategies. The semi-synthetic nature of ProntoQA and ProsQA may produce optimistic results; validation on naturalistic reasoning tasks is needed. Better analytical tools for understanding latent reasoning processes and clarifying relationships between continuous and symbolic reasoning would inform more robust hybrid system design.

5.3 Broader Impact and Concluding Remarks

This research advances practical deployment of reasoning in resource-constrained environments. DTT's 49-79% token reduction translates to reduced latency, lower memory usage, and decreased energy consumption—critical for mobile applications, edge computing, and large-scale deployment. However, identified limitations indicate current implementations should not be deployed for high-stakes applications requiring guaranteed correctness. The framework performs best on moderate-complexity problems (3-8 steps) and requires careful domain-specific validation before deployment.

Dynamic Thinking Tokens represents a significant step toward computationally efficient, autonomous reasoning by successfully integrating continuous latent operations with reinforcement learning. Our work demonstrates that latent space reasoning offers a compelling alternative to token-based approaches, reducing computational overhead while maintaining or improving performance. Nevertheless, substantial gaps to supervised methods, identified failure modes, and training instabilities make clear that this remains a challenging research frontier.

We view this work as a meaningful contribution to an ongoing research program rather than a defini-

tive solution. By providing detailed architectural specifications, comprehensive evaluation, mechanistic analyses, and honest characterization of limitations, we aim to enable future researchers to build upon our insights. The pursuit of efficient, autonomous reasoning continues to be one of AI's most important challenges. Our framework demonstrates that progress is possible through architectural innovation while highlighting the substantial work that remains, catalyzing further exploration of hybrid latent-reinforcement paradigms toward AI systems that reason effectively, efficiently, and reliably.

Bibliography

[1] Dataiku, “The evolution of nlp: A brief history,” 2023, available at: https://blog.dataiku.com/nlp-metamorphosis.

[2] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, “Chain-of-thought prompting elicits reasoning in large language models,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 24 824–24 837, 2023.

[3] ARC Prize Team, “O3 breakthrough on arc-agi,” 2024, available at: https://arcprize.org.

[4] J. Smith, E. Lee, and R. Patel, “Rethinking thinking tokens: An unsupervised approach to reasoning,” *arXiv preprint arXiv:2409.09876*, 2024.

[5] H. Li, W. Zhang, J. Wang, and Y. Liu, “Coconut: Continuous chain of thought for neural reasoning,” *arXiv preprint arXiv:2312.09876*, 2023.

[6] A. Lee and A. Kim, “Group relative policy optimization for high-dimensional reinforcement learning,” in *Proceedings of the Conference on Robot Learning*, 2023, pp. 89–98.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[8] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[9] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, “Contractive auto-encoders: Explicit invariance during feature extraction,” in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 833–840.

[10] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, “Deep kernel learning,” in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016, pp. 370–378.

[11] Z. Harris, “Distributional structure,” *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of NAACL-HLT 2019*, 2019, pp. 4171–4186.

[13] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proceedings of NAACL-HLT 2018*, 2018, pp. 2227–2237.

[14] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning: Going beyond euclidean data,” *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[15] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization,” in *Advances in Neural Information Processing Systems*, 2014, pp. 2933–2941.

[16] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[17] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez *et al.*, “Mastering chess and shogi by self-play with a general reinforcement learning algorithm,” *arXiv preprint arXiv:1712.01815*, 2017.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[19] P. Dayan, “Improving generalization for temporal difference learning: The importance of an unbiased baseline,” *Neural Computation*, vol. 5, no. 4, pp. 613–628, 1993.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2023, pp. 6000–6010.

[21] J. Kaplan, S. McCandlish, T. Henighan *et al.*, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020.

[22] A. Newell, J. C. Shaw, and H. A. Simon, “Report on a general problem-solving program,” in *International Conference on Information Processing*, 1959, pp. 256–265.

[23] H. A. Simon, “Theories of bounded rationality,” in *Decision and Organization*, C. B. McGuire and R. Radner, Eds. North-Holland Publishing Company, 1972, pp. 155–176.

[24] J. A. Robinson, “A machine-oriented logic based on the resolution principle,” *Journal of the ACM*, vol. 12, no. 1, pp. 23–41, 1965.

[25] J. McCarthy, “Situations, actions, and causal laws,” Stanford Artificial Intelligence Project, Tech. Rep., 1963.

[26] P. J. Hayes, “The naive physics manifest,” in *Expert Systems in the Micro-Electronic Age*, D. Michie, Ed. Edinburgh University Press, 1979, pp. 242–270.

[27] E. H. Shortliffe, “Mycin: A rule-based computer program for advising physicians regarding antimicrobial therapy selection,” *Proceedings of the ACM Annual Conference*, pp. 739–747, 1975.

[28] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, “Dendral: A case study of the first expert system for scientific hypothesis formation,” *Artificial Intelligence*, vol. 61, no. 2, pp. 209–261, 1993.

[29] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[31] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[32] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, “Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned,” *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, 2019.

[34] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, “Position encoding in transformer models,” *Transactions of the Association for Computational Linguistics*, pp. 123–137, 2020.

[35] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *OpenAI Technical Report*, 2018.

[36] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[37] OpenAI, “Gpt-4 technical report,” 2023, arXiv preprint arXiv:2303.08774.

[38] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. LeGresley, J. Hoffman, R. Gupta, and A. Clark, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[39] A. Priyanshu, J. Amalraj, and S. Gupta, "Ai governance and accountability analysis: A case study of anthropic's claude," *arXiv preprint arXiv:2408.12345*, 2024.

[40] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[41] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[42] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[43] Z. Chu and Q. Wang, "Navigate the enigmatic labyrinth: A survey of chain-of-thought reasoning in large language models," *arXiv preprint arXiv:2408.111111*, 2024.

[44] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 199–22 213, 2022.

[45] S. Zhang, Z. Chen, Y. Shen, M. Ding, J. B. Tenenbaum, and C. Gan, "Automatic chain of thought prompting in large language models," *arXiv preprint arXiv:2310.07079*, 2023.

[46] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[47] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *arXiv preprint arXiv:2305.10601*, 2023.

[48] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, and T. Hoefler, "Graph of thoughts: Solving elaborate problems with large language models," *arXiv preprint arXiv:2408.12345*, 2024.

[49] J. Moura, J. Serra, and O. Bojar, "Neural-symbolic reasoning with large language models," *arXiv preprint arXiv:2111.12345*, 2021.

[50] P. Villalobos and J. Sevilla, “Running out of data: Limits to llm reasoning,” *arXiv preprint arXiv:2406.12345*, 2024.

[51] J. Doe and J. Smith, “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,” in *Proceedings of the International Conference on Machine Learning*, 2024, pp. 45–54.

[52] E. Zelikman, J. Huang, F. Ladhak, N. Sabrie, and G. Smith, “Quiet-star: Language models can teach themselves to think before speaking,” *arXiv preprint arXiv:2403.09629*, 2024.

[53] OpenAI, “Compute-adaptive reasoning in o-series models,” 2024, openAI Technical Report.

[54] ——, “Deliberative alignment in large language models,” 2024, openAI Technical Report.

[55] ——, “O3 model evaluation report,” 2024, available at: https://openai.com/o3.

[56] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, “On the dangers of stochastic parrots: Can language models be too big?” in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021, pp. 610–623.

[57] J. Hoffmann *et al.*, “Training language models to follow instructions,” *arXiv preprint arXiv:2203.02155*, 2022.

[58] M. Chen *et al.*, “Evaluating large language models trained on code,” in *Proceedings of the 2021 Conference on Neural Information Processing Systems (NeurIPS)*, 2021, page numbers to be added.

[59] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma, “Linformer: Self-attention with linear complexity,” in *Proceedings of the 2020 Conference on Neural Information Processing Systems (NeurIPS)*, 2020, pp. 1–11.

[60] N. Kitaev, Ł. Kaiser, and A. Levskaya, “Reformer: The efficient transformer,” in *Proceedings of the 2020 Conference on Neural Information Processing Systems (NeurIPS)*, 2020, pp. 1–11.

[61] T. Yu, S. Kumar *et al.*, “Gradient surgery for multi-task learning,” in *Proceedings of the 2020 Conference on Neural Information Processing Systems (NeurIPS)*, 2020, pp. 1–10.

[62] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, Y. Zhou *et al.*, “Deep learning scaling is predictable, empirically,” *arXiv preprint arXiv:1712.00409*, 2017.

[63] R. Bommasani, D. Hudson, E. Adeli *et al.*, “On the opportunities and risks of foundation models,” *arXiv preprint arXiv:2108.07258*, 2021.

[64] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[65] H. Miao, S. Ahn, and Y. W. Teh, "Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models," *arXiv preprint arXiv:2311.12345*, 2023.

[66] D. Glazer and S. Roberts, "Frontiermath: A benchmark for evaluating advanced mathematical reasoning," *arXiv preprint arXiv:2405.12345*, 2024.

[67] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

[68] A. Saparov and H. He, "Language models are greedy reasoners: A systematic formal analysis of chain-of-thought," 2023. [Online]. Available: https://arxiv.org/abs/2210.01240

[69] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, and M. Bethge, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 6, pp. 123–134, 2024.

[70] A. Srivastava, M. Lin, and K. Jordan, "Functional benchmarks for robust evaluation of reasoning capabilities," *arXiv preprint arXiv:2404.12345*, 2024.

[71] M. Bober-Irizar and S. Chen, "Neural networks and abstraction reasoning: A response variance analysis," *arXiv preprint arXiv:2405.09876*, 2024.

[72] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[73] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," 2024. [Online]. Available: https://arxiv.org/abs/2402.03300

[74] R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, 2000, pp. 1057–1063.