THINK DEEP, SPEAK ONCE: RELIT, A RECURSIVE LATENT IMPLICIT TRANSFORMER FRAMEWORK

Abhishek Panwar, Maheep Singh, Saksham Bansal *
Indian Institute of Technology Roorkee, India
{abhishek_p1, maheep_s, saksham_b}@me.iitr.ac.in

ABSTRACT

Chain-of-Thought (CoT) prompting has become the dominant paradigm for eliciting reasoning in Large Language Models (LLMs), yet it creates substantial computational overhead by forcing models to externalize intermediate reasoning steps as discrete tokens. Recent latent reasoning approaches attempt to internalize this process within continuous hidden states. One of the latest advancements in the field of latent reasoning, Tiny Recursive Models (TRMs) excel at symbolic reasoning but struggle to preserve semantic coherence in natural language settings. To bridge this gap, we introduce **ReLIT** (**Re**cursive **L**atent **I**mplicit **T**ransformer), a hybrid framework that grounds deep recursive reasoning within the rich semantic representations of a foundational model. ReLIT augments a frozen LLM backbone (TinyLlama-1.1B) with a lightweight, trainable recursive block that iteratively refines its latent thinking (z) before committing to a final output, structurally solving linguistic intuition from algorithmic processing and enabling "deep thinking" via gradient-isolated recurrent loops without the latency of explicit token generation. Empirically, ReLIT achieves high parameter efficiency on the GLoRE logical reasoning benchmark, matching or outperforming significantly larger models on challenging tasks such as ProofWriter and RuleTaker despite minimal supervision. These results demonstrate that reasoning capability can be scaled efficiently through recurrent depth rather than parameter width, offering a principled framework for semantically grounded implicit reasoning.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities in complex problem-solving. The dominant paradigm driving this success is **Chain-of-Thought (CoT)** reasoning (Wei et al., 2022), where models are prompted to explicitly generate intermediate reasoning steps in natural language before producing a final answer. By verbalizing the thought process, CoT decomposes complex problems into manageable sequential steps. However, this reasoning methodology comes with significant efficiency drawbacks. CoT incurs substantial computational costs due to the generation of long, verbose reasoning chains, which increase latency and memory consumption. Moreover, while autoregressive natural language generation aligns intuitively with human communication, recent research suggests it may not be the most efficient medium for reasoning. (Hao et al., 2024). Generating discrete tokens for every atomic reasoning step forces the model to collapse high-dimensional "thoughts" into low-bandwidth text, potentially losing information and wasting compute on syntactic overhead. To address this, the field is shifting toward **Latent Reasoning** (Zhu et al., 2025). This paradigm explores performing reasoning steps within continuous hidden representations rather than discrete token space. By internalizing the "chain of thought" into a recurrent or deep latent process, models can theoretically reason with greater expressiveness and efficiency. Despite this promise, a gap remains in how to effectively architect these latent reasoners for natural language. Training recursive models from scratch on language is difficult due to the requirement of large amounts of computational resources. To bridge this gap between robust language understanding and efficient recursive logic, we introduce **ReLIT**, a framework that combines deep recursive reasoning with the semantic intuition of a foundation model.

*Equal contribution Code: https://github.com/mdgspace/Think-Deep-Speak-Once-ReLIT

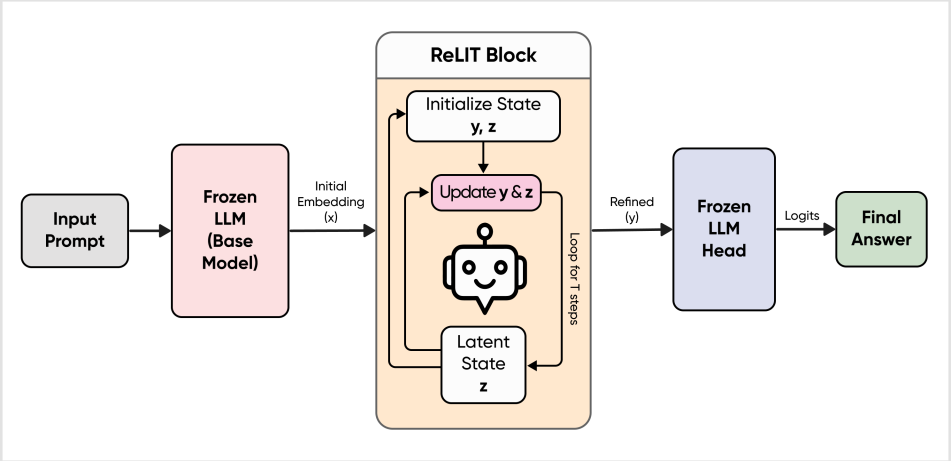

Figure 1: Inference Pipeline of the ReLIT Framework. The input prompt is processed by a frozen LLM to generate initial embeddings, which are iteratively refined within the recurrent ReLIT Block.

1.1 Related Work

Chain-of-Thought prompting for reasoning (Wei et al., 2022) is not only expensive but also single missteps in the intermediary "logic" can break the entire reasoning chain. The community has thus come up with many approaches to both increase and make more efficient the reasoning capabilities of an LLM. Such works include Implicit CoT (Deng et al., 2023) where intermediate steps are removed gradually and the model is finetuned so as to internalize the CoT process. Further works on the concept of internalization and Latent reasoning include COCONUT (Chain of Continuous Thought) (Hao et al., 2024) which challenges the inefficiency of human language as a reasoning medium and uses a latent space to reason by feeding the last hidden state as the input embedding for the next token; Dualformer (Su et al., 2024) which integrates fast and slow reasoning modes via training on randomized reasoning traces. Parallel works have developed on the idea that recurrent depth is the key to algorithmic reasoning. Work on Universal Transformer (Dehghani et al., 2019) increased theoretical capabilities via parallel-in-time recurrent self-attentive sequence models. AlgoFormer (Gao et al., 2024) increased the algorithmic capabilities of transformers through the combination of a pre-transformer, a looped transformer and a post transformer. Geiping et al. (2025) validated empirically the improved reasoning performance and efficiency that recurrent depth provides over standard Chain-of-Thought reasoning.

Tiny Recursive Models, Hierarchical Reasoning Model and the Semantic Gap

Hierarchical Reasoning Model (Wang et al., 2025) and Tiny Recursive Model (Jolicoeur-Martineau, 2025) demonstrate the ability of tiny networks on reasoning tasks that LLMs find difficult. Moreover the TRM architecture displays exemplary performance through the use of recursive reasoning. However, these models lack the ability to understand natural language and hence their domain is limited. Our work utilizes ReLIT which draws inspiration from TRM but resolves the semantic limitation with the use of a frozen pre-trained foundational model.

1.2 The ReLIT Framework: An Introduction

ReLIT (Recursive Latent Implicit Thinking) is a hybrid architecture that augments a frozen foundation model with a trainable Recursive Reasoning Block.

Instead of treating reasoning as a single monolithic process, ReLIT structurally decouples it into three distinct, interacting vector states that evolve over time:

- x **(The Semantic Anchor):** A static, high-dimensional intuition extracted from the frozen LLM's last layer. It ensures the model never loses the semantic context of the prompt, no matter how deep the reasoning goes.

- y **(The Answer State):** A dynamic vector representing the model's current hypothesis. It starts as a rough guess and is refined at every step, converging toward the solution.
- z **(The Latent Scratchpad):** A dedicated "hidden thought" vector. Unlike the answer state, z is free to explore abstract, high-dimensional logic paths without needing to map to a valid output, acting as a purely internal working memory.

By formalizing reasoning as the iterative interaction of these three vectors,where x grounds the thought, z processes the logic, and y captures the result; ReLIT enables deep, multi-step deduction on natural language tasks without the computational overhead of explicit token generation.

2 OUR METHODOLOGY

We propose **ReLIT Framework** as shown in fig.2 , a tripartite architecture designed to decouple linguistic intuition from logical reasoning. The model follows a "sandwich" structure, effectively situating a high-frequency *(Recursive Latent Implicit Thinking)* (ReLIT) block between the frozen layers of a pre-trained Large Language Model (LLM).

2.1 THE BACKBONE: SEMANTIC GROUNDING

The foundation of our architecture utilizes a frozen **TinyLlama-1.1B** Zhang et al. (2024) backbone.The primary role of the backbone is to provide **Semantic Grounding** Lyre (2024). By passing the input through the pre-trained transformer once, we extract the initial latent representation $x \in \mathbb{R}^{B\times L\times D}$, where D is the hidden dimension. This ensures the ReLIT Block inherits a rich, high-dimensional understanding of language, allowing it to focus exclusively on logic and state-refinement.

2.2 RELIT BLOCK

ReLIT architecture is inspired from Tiny Recursive Models(TRMs) with novel ammendments. The internal architecture of ReLIT is built upon a high-performance transformer-inspired block, utilizing **RMSNorm** Zhang & Sennrich (2019) for pre-normalization and a **SwiGLU** Shazeer (2020) based feed-forward network to ensure the cell possesses the expressive capacity of much larger models while remaining computationally lean. Rather than following the TRM approach of direct state replacement, our implementation introduces a **Residual State Refinement** strategy as a core structural novelty. In this framework, the model does not attempt to regenerate the thought vector $\mathbf{z}$ and answer state $\mathbf{y}$ from scratch at each step. Instead, it computes a logical *delta*, or refinement, which is added back to the existing states:

$$z_{n+1} = \text{RMSNorm}(z_n + \text{ReLIT}(\text{Linear}([x, y_n, z_n]))) \tag{1}$$

By framing the update as $\mathbf{z} + \Delta\mathbf{z}$, the ReLIT acts as a differential reasoning engine. This allows the model to incrementally "sculpt" the final answer over N_{latent} iterations and T deep recursions, preserving the foundational linguistic context x provided by the backbone while iteratively stripping away logical inconsistencies. This residual dependency ensures smoother gradient flow during the deep supervision process and prevents the vanishing of semantic information during extended "thinking" cycles.

2.3 THE DECODER: FROM LATENTS TO TOKENS

The final component is the Projection Head, which decrypts the evolved latent state y into the vocabulary space using a strategy tailored to the dataset's reasoning requirements. By employing a Frozen Base Head, we enforce Semantic Consistency, anchoring the ReLIT's reasoning to the original LLM's semantic map to prevent latent drift. Alternatively, a Trainable Decoder allows for Task-Specific Specialization, granting the model the flexibility to re-map its internal logic to specialized domain tokens however, it requires higher compute and tokens than frozen one. This modular design provides a choice between the regularizing effect of a frozen head for general language and the adaptive precision of a trained head for complex, logic-heavy tasks.

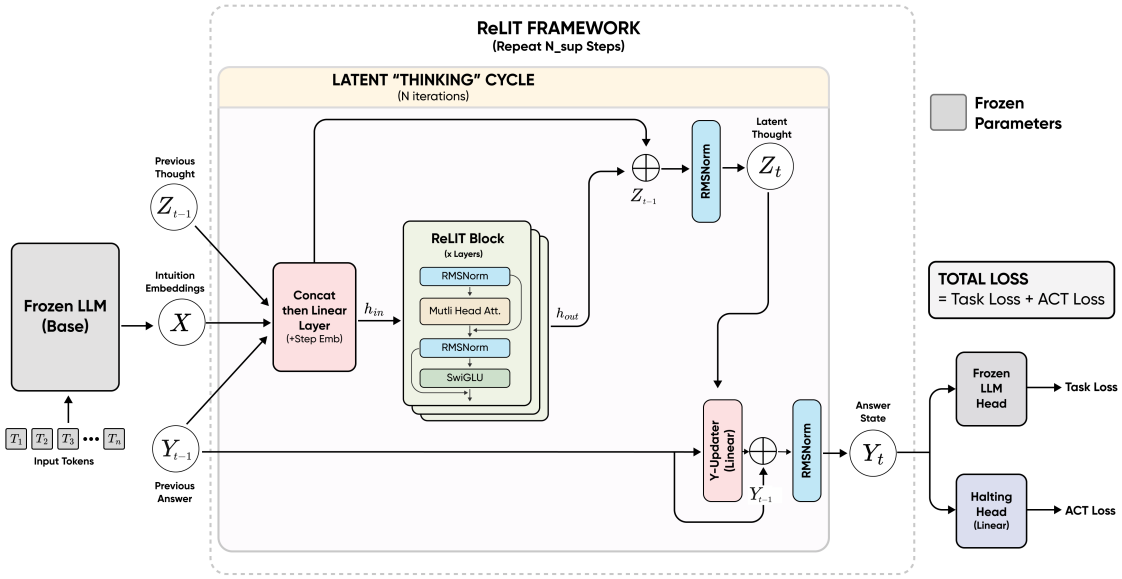

Figure 2: **The ReLIT Pipeline.** An overview of the proposed architecture featuring a tripartite structure: (1) Semantic Grounding via a frozen LLM backbone, (2) Recursive Refinement within the latent ReLIT block, and (3) Hypothesis Decoding via the frozen head.

Furthermore, we deliberately employ a Decoder-only backbone (TinyLlama-1.1B) instead of a bidirectional Encoder to extract the semantic intuition . While Encoders often yield superior embeddings for discriminative tasks, a causal Decoder architecture ensures the framework remains forward-compatible with multi-token generation. This choice aligns with our broader research objective to establish ReLIT as a foundation for **Latent Chain-of-Thought** reasoning, enabling the recursive "thinking" phase to be seamlessly integrated into sequential autoregressive generation in future iterations.

3 Mathematical Formulation

We formulate ReLIT as a discrete dynamical system that iteratively refines a latent hypothesis. Unlike standard Transformers that map input to output in a single pass, our framework introduces a *time* dimension to the thinking process, allowing the model to revisit and improve its internal state before committing to a prediction.

3.1 The Latent State Triad

We define the system state at any recursive step t via three distinct vector fields, where L denotes the sequence length and D represents the hidden dimension.

1. **The Semantic Intuition (x):** A static semantic anchor derived from the frozen backbone ($\text{LLM}_{\text{frozen}}$). It remains constant to prevent semantic drift:

$$x = \text{LLM}_{\text{frozen}}(input), \qquad x \in \mathbb{R}^{L \times D} \tag{2}$$

2. **The Latent Reasoning State (z_t):** A "scratchpad" memory vector capturing the intermediate chain of thought. It is initialized from a learnable Gaussian distribution:

$$z_0 \sim \mathcal{N}(0, \sigma^2 I), \qquad z_t \in \mathbb{R}^{L \times D} \tag{3}$$

3. **The Answer State (y_t):** The evolving representation of the final output, initialized as a copy of the semantic intuition and refined incrementally:

$$y_0 = x, \qquad y_t \in \mathbb{R}^{L \times D} \tag{4}$$

3.2 Recursive Dynamics and Recall-Then-Learn

We model reasoning as a discrete dynamical system governed by a transition function F_θ, evolving a latent state (y_t, z_t) conditioned on the static anchor x. To enable deep recurrence without the memory overhead of Backpropagation Through Time (BPTT)Bird & Polivoda (2025), we utilize a **Recall-Then-Learn** protocol. This approximations a stable fixed point via a gradient-free warm-up, ensuring a minimal memory footprint.

3.2.1 Recursive Transition Function

Let h_t denote the fused hidden representation at step t. The state update $(y_{t+1}, z_{t+1}) = F_\theta(y_t, z_t; x)$ is defined by the following transformation:

$$h_t = W_c[x\,;\, y_t\,;\, z_t] + e_t \tag{5}$$

$$z_{t+1} = \mathrm{Norm}_z\big(z_t + \mathcal{R}_\theta(h_t)\big) \tag{6}$$

$$y_{t+1} = \mathrm{Norm}_y\big(y_t + W_y[y_t\,;\, z_{t+1}]\big) \tag{7}$$

where $[-;-;-]$ denotes concatenation, W_c and W_y are learned linear projections, e_t is a step-dependent embedding, and $\mathcal{R}_\theta$ is the recursive Transformer block.

3.2.2 Recall-Then-Learn Protocol

We approximate the logical fixed point by decoupling the training depth from the inference depth through two distinct phases:

Phase I: Recall (Warm-up). Updates are performed with gradients disabled to allow the system to settle into a stable reasoning path:

$$(y_{t+1}, z_{t+1}) = F_\theta(y_t, z_t; x), \qquad t = 0, \ldots, T-2 \tag{8}$$

Phase II: Learn (Optimization). A single gradient-enabled update is performed on the matured state:

$$(y_T, z_T) = F_\theta\big(\mathrm{stopgrad}(y_{T-1}, z_{T-1}); x\big) \tag{9}$$

By detaching the states between supervision steps (BPTT) Bird & Polivoda (2025), the framework enables deep, multi-step deliberation while maintaining a manageable memory footprint.

3.2.3 Projection and Inference

The final refined answer state is projected back into the vocabulary space using the frozen output projection of the base LLM (W_{lm}):

$$p(\cdot \mid s) = \mathrm{Softmax}\big(W_{\text{lm}}\, y_T\big) \tag{10}$$

During training, we apply **Deep Supervision Lee et al. (2014)** by computing cross-entropy loss on y_t at each macro-step. This stabilizes training by encouraging intermediate states to remain aligned with the target output throughout the recursion.

3.3 Recursive Latent Refinement with Adaptive Halting

The predefined states of y and z are iteratively refined through a series of N supervision steps, each containing T recursive passes. We utilize residual updating and Layer Normalization to ensure training stability.

To optimize the computation budget we employ Adaptive Computation Time (ACT)Graves (2017) mechanism where a Halt Head predicts a confidence-based halting probability, allowing the model to dynamically mask and scale the loss for samples that have reached logical convergence.

4 TRAINING METHODOLOGY

To better illustrate the training methodology, this subsection will delve in detail into the process. Please refer to the pseudocode provided in the appendix.

We start by sending the input prompt to the frozen base model (TinyLlama) and extract the hidden layer x from the last layer. Then we initialize y (current answer) and z (latent internal reasoning). The input for the ReLIT Block h_{in} is created by concatenating x,y_{t-1},z_{t-1}; passing them through a linear layer and adding step embeddings. The ReLIT Block refers to a recursive Transformer Block containing a RMS Normalization layer, MutliHead Attention layer, another RMS Normalization layer and a SwiGLU Network. The output of the ReLIT Block h_{out} is used for the residual updating of z_t. The underlying process was inspired from Tiny Recursive Model(TRMs)'s training process. We perform this latent reasoning step N_{latent} times per update of y (the current answer). This update of y_t can be referred to as latent recursion. We perform latent recursion $T-1$ times without gradient and then perform a single gradient-enabled update on the matured state. This is known as Truncated BPTT (Backpropagation Through Time). We then speak i.e. project back to tokens using the frozen LLM Head.

For every Epoch we perform the aforementioned steps $N_{supervision}$ times and calculate the task loss and the ACT loss. The task loss is the Cross Entropy loss between the logits produced and logits expected. The ACT loss is the Binary Cross Entropy loss, which predicts if we have reached the correct output and can halt. It allows us to scale the loss for samples that have reached logical convergence.

5 RESULTS AND DISCUSSION

5.1 EVALUATION ON GLORE

To validate the logical depth and recursive reasoning capabilities of our framework, we conduct extensive evaluations on the **General Logic Reasoning Evaluation (GLoRE) liu et al. (2025)** benchmark. We select five diverse datasets: *ProofWriter* Tafjord et al. (2021), *RuleTaker* Clark et al. (2020), *HELP* Kavumba et al. (2021), *NaN-NLI* Truong et al. (2022), and *TaxiNLI* Joshi et al. (2020)—to rigorously assess performance across propositional logic, counterfactual reasoning, and multi-hop path planning tasks.

While these benchmarks are traditionally evaluated under few-shot prompting paradigms, the parameter efficiency of ReLIT enables direct supervision without the prohibitive cost of full-model fine-tuning. Accordingly, we freeze the base foundation model and train only the ReLIT block on the designated training splits, reporting accuracy on held-out evaluation sets. We consider a wide range of data regimes, from extremely low-resource settings (e.g., NaN-NLI with 200 samples) to larger logical corpora (e.g., ProofWriter with 5,000 samples), demonstrating the adaptability of ReLIT across supervision scales. Detailed dataset statistics and training hyperparameters are provided in Appendix A.

5.2 DATASET EXPLAINATIONS

5.2.1 PROOFWRITER

ProofWriter evaluates deductive logic by requiring the model to chain multiple implications (e.g., $A \rightarrow B \rightarrow C$). We trained on 5,000 samples and tested on 3,000. The model exhibits a clear "thinking" phase, halting at an average depth of 3.6 steps as it moves from initial uncertainty to a stable logical fixed point. This haulting value signifies that model's answer are consistent after 3-4 supervision loops. This process works by mathematically shifting the entity vector (e.g., **Bob**) through semantic regions (e.g., **Feline** $\rightarrow$ **Carnivore**) via residual updates until the logic is resolved (Intution example is shown in fig 4). Essentially, the ReLIT treats reasoning as a geometric navigation task, where the final stable vector position directly maps to the correct logical verdict. A remarkable accuracy of 98.6% as given in **Table1** shows the latent and Implicit thinking of **ReLIT** framework which is even higher than top LLM performers on the particular data.

Model	NLI			TF		Avg
	HL	TN	NN	RT	PW	
RoBERTa	39.47	49.91	**90.02**	53.50	55.92	57.76
LLaMA	32.26	41.91	47.29	48.89	53.78	44.83
Falcon	28.49	44.66	53.31	56.11	53.33	47.18
Mixtral-8x7B	33.27	40.86	50.13	46.84	44.80	43.18
ChatGPT	42.13	57.30	56.59	54.74	53.95	52.94
GPT-4	46.01	60.08	76.74	60.19	59.66	60.54
o1 mini	**63.69**	81.41	71.55	75.90	75.33	73.58
DeepSeek R1	62.05	75.74	72.58	75.29	80.51	73.23
QwQ-32B	61.53	**81.96**	75.59	78.10	82.40	**75.92**
ReLIT (Ours)	53.43	56.30	55.20	**97.60**	**98.60**	72.23

Table 1: **Performance on select GLoRE benchmark tasks.** Tasks are grouped by category: Natural Language Inference (NLI) and Theorem Finding / Logic (TF). *HL*: HELP, *TN*: TaxiNLI, *NN*: NaN-NLI, *RT*: RuleTaker, *PW*: ProofWriter. All results are accuracy (%). Best results are in **bold**.Results for LLMs have been taken from GLoRE liu et al. (2025)

5.2.2 RULETAKER

The RuleTaker dataset similarly evaluates the model's capacity for rule-based inference, often involving larger rule sets and more complex distractors. We trained it on 5000 samples and evaluated on 3000 samples and observed that model follows the same geometric intuition established in the ProofWriter task, utilizing the recursive loops to navigate through the logical constraints of the prompt. For this dataset, the average halting depth increases to **5.2 steps**. This slightly higher value reflects the increased complexity of the rules, requiring the ReLIT-Block to perform additional residual updates to reach a stable logical fixed point. Despite the increased depth, the underlying process remains consistent as can be noticed in fig 3: the model incrementally shifts the latent states until the entity-property relationships are resolved. The fact that the halting depth naturally scales with task complexity proves that the ReLIT is dynamically adapting its "thinking time" to the difficulty of the rule set. The accuracy of 97.6% again prooves our architecture capability to reason on logical dataset without Chain-of-Thought(CoT).

5.2.3 NAN-NLI

The NaN-NLI dataset presents a unique challenge due to extreme data scarcity, with only 200 samples available for training and 50 for evaluation. Despite these severe computational and data constraints, the model achieves good accuracy comparable to models such as ChatGPT, demonstrating the robust semantic grounding provided by the frozen LLM backbone.

Due to the limited training signal,we get **55.2%** accuracy which is still comparable to **ChatGPT's** accuracy. The model exhibits a higher average halting depth of **7.8 steps** which can be noticed in fig. 3. This increased depth illustrates the consequences of low-data regimes: the ReLIT must perform more internal recursions to stabilize its logical state and resolve the linguistic nuances of the inference task. However, the critical finding is that the outputs remain consistent after this stabilization phase. The high confidence in the final supervision loops proves that the model is not hallucinating or making stochastic guesses; rather, it is performing deliberate reasoning to overcome the lack of extensive training data. This result reinforces our conclusion that the ReLIT-backbone coupling is a highly sample-efficient architecture for logical tasks.

5.2.4 HELP

On the HELP benchmark, we trained on 800 samples and evaluated on 200, achieving a notable accuracy of **53.43%** even beating **ChatGPT** and **GPT-4**. While this result is competitive with much larger models, the core significance lies in the model's reliability; with an average halting depth of **6.8 steps**, the **ReLIT** framework demonstrates that it is not relying on stochastic guesses

Context: Anne is quiet. Anne is not young.
Bob is kind. Bob is young.
Dave is rough. Dave is round.
Kind, young things are not smart.

Question: Bob is kind.

Answer: No->Yes->No->Yes->Yes.....->Yes

Context: Not all people have had the opportunities you have had.

Hypothesis: Not all people have had the opportunities you have not had.

Answer: No->Yes->Neutral->Yes->Yes->No->Neutral->Neutral.....->Neutral

Figure 3: **Latent Reasoning Trajectories and Adaptive Halting.** Visualization of N supervision steps during ReLIT inference on *RuleTaker* (left) and *NaN-NLI* (right). The trajectories illustrate the iterative refinement of the state vector $\boldsymbol{v}$. The blue highlights denote the dynamically identified **halting steps**, where the latent state achieves numerical stability, indicating logical convergence before final answer projection.

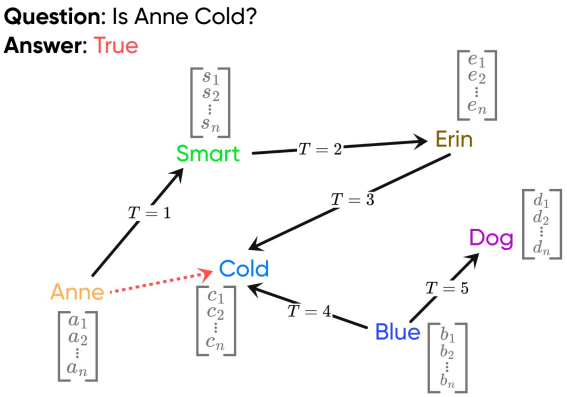

Figure 4: **Latent Reasoning Trajectory on ProofWriter.** This visualization depicts the iterative evolution of the state vector within the continuous latent space. Starting from the initial grounding (*Anne*), the model continuously updates its output vector $\vec{y}$ across T steps, transitioning through intermediate logical predicates (*Smart* $\rightarrow$ *Erin* $\rightarrow$ *Cold*). The trajectory demonstrates how the recurrent ReLIT block refines high-dimensional "hidden thoughts" to converge onto the final logical state (*Cold*) without generating intermediate tokens.

or hallucinations. Instead, the stabilization of the latent state across recursive loops ensures that the final answers are the product of consistent logical reasoning. This proves that our **ReLIT** is succesfully reasoning on logical datasets instead of random guesses.

5.2.5 TAXI NLI (TN)

On the Taxi NLI (TN) benchmark, ReLIT demonstrates robust reasoning capabilities despite limited training due to computational constraints we trained it on 2000 samples and evaluated on 800 samples. While achieving an accuracy of **56.30%**, the model's core significance lies in its avoidance of stochastic guesses and hallucinations. By stabilizing the latent state across recursive loops, ReLIT ensures that its outputs are the product of consistent logical inference rather than random patterns. This proves the model's ability to maintain logical integrity even in complex NLI tasks with reduced training overhead.

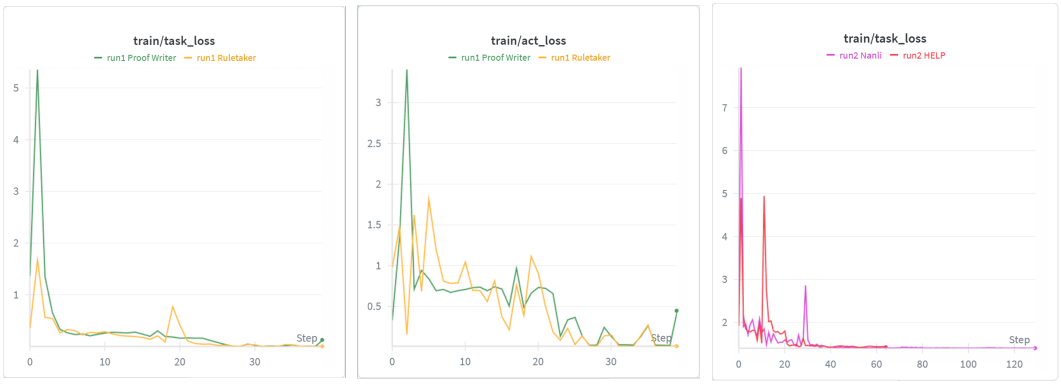

Figure 5: **Training Dynamics and Convergence Profiles.** Visualization of the optimization process across multiple benchmarks. The left and center panels show the **task loss** and **ACT loss** for *ProofWriter* and *RuleTaker*, respectively, demonstrating a sharp initial descent followed by stability. The right panel illustrates the **task loss** for *NaN-NLI* and *HELP*, where the smooth convergence toward a lower bound indicates effective latent state grounding and reliable learning of logical dependencies over training steps.

6 Future Work and Conclusion

6.1 Future Work: Deep Thinking, Singular Expression

The current implementation of the **ReLIT** block demonstrates that recursive latent structures can inherit deep language understanding via implicit reasoning over frozen LLM memory. We envision several high-impact directions for this architecture:

- **Specialized Semantic Encoders:** ReLIT can serve as a sophisticated replacement for BERT Devlin et al. (2019)-based models, specifically in tasks like **FinBERT** Huang et al. (2023) for financial sentiment analysis. In these domains, the model must "think deep but speak once" i.e. process complex market nuances through multiple latent recursions before outputting a single, high-integrity semantic score.
- **Recursive Classifiers:** Beyond generative tasks, ReLIT can be utilized as a robust classifier for document-level entailment or paragraph-to-paragraph semantic matching, where the recursive loops allow the model to resolve contradictions that standard feed-forward encoders miss.
- **Scalability to Multi-Token Generation:** While our current work focuses on single-token reasoning (thinking deeply before providing a definitive answer), a natural extension is the transition to multi-token autoregressive training. Although this presents significant computational challenges and requires high-density hardware for gradient accumulation across recursive steps, such an approach could allow the ReLIT block to maintain a continuous "stream of consciousness" during long-form generation. If computational resources permit, this would test whether the implicit vector math observed in our study scales to the complex dependencies of full-paragraph synthesis.

6.2 Conclusion

In this paper, we introduced a novel "sandwich" architecture that decouples linguistic intuition from logical reasoning. By situating the **ReLIT** block within the latent space of a frozen LLM, we demonstrate that models can achieve high-performance logical reasoning without the need for explicit Chain-of-Thought (CoT) prompting.Our results across benchmarks like ProofWriter and HELP prove that implicit, latent thinking is both data-efficient and mathematically stable. We conclude that the future of robust AI lies not in simply scaling parameter counts, but in developing specialized recursive engines that can leverage pre-trained semantic memories to solve the underlying geometric structure of logic.

ACKNOWLEDGMENTS

We thank the members of MDG SPACE, IIT Roorkee for their support and encouragement throughout this work.

REFERENCES

George Bird and Maxim E. Polivoda. Backpropagation through time for networks with long-term dependencies, 2025. URL https://arxiv.org/abs/2103.15589.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4102–4109, 2020.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2019. URL https://arxiv.org/abs/1807.03819.

Yuntian Deng, Kiran Chandrasekar, Makesh Sreedhar Chidambaram, and Alexander M Rush. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Yihang Gao, Chuanyang Zheng, Enze Xie, Han Shi, Tianyang Hu, Yu Li, Michael K Ng, Zhenguo Li, and Zhaoqiang Liu. Algoformer: An efficient transformer framework with algorithmic structures. *arXiv preprint arXiv:2402.13572*, 2024.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

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

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Allen H Huang, Hui Wang, and Yi Yang. FinBERT: A large language model for extracting information from financial text. *Contemporary Accounting Research*, 40(2):806–841, 2023.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025.

Pratik Joshi, Somak Aditya, Aalok Satheesh, and Mohit Bansal. TaxiNLI: Taking a ride up the NLU hill. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pp. 41–55, 2020.

Isaac Kavumba et al. HELP: A dataset for human-level explainable planning and reasoning, 2021.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets, 2014. URL https://arxiv.org/abs/1409.5185.

Hanmeng liu, Zhiyang Teng, Ruoxi Ning, Yiran Ding, Xiulai Li, Xiaozhang Liu, and Yue Zhang. Glore: Evaluating logical reasoning of large language models, 2025. URL https://arxiv.org/abs/2310.09107.

Holger Lyre. "understanding ai": Semantic grounding in large language models, 2024. URL https://arxiv.org/abs/2402.10992.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

DiJia Su, Sainbayar Sukhbaatar, Arthur Szlam, Edouard Grave, Gabriel Synnaeve, and Pierre-Etienne Mazar'e. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces. *arXiv preprint arXiv:2410.09918*, 2024.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Thinh Hung Truong, Yulia Otmakhova, Timothy Baldwin, Trevor Cohn, Jey Han Lau, and Karin Verspoor. Not another negation benchmark: The nan-nli test suite for sub-clausal negation, 2022. URL https://arxiv.org/abs/2210.03256.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

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

Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, Zhenhe Wu, Yongchi Zhao, Tianyu Liu, Jian Yang, Wangchunshu Zhou, Chujie Zheng, Chongxuan Li, Yuyin Zhou, Zhoujun Li, Zhaoxiang Zhang, Jiaheng Liu, Ge Zhang, Wenhao Huang, and Jason Eshraghian. A survey on latent reasoning, 2025. URL https://arxiv.org/abs/2507.06203.

A Appendix

A.1 Convergence Analysis and the Information Bottleneck

In this appendix, we analyze the convergence properties of the ReLIT framework. In particular, we study the dependence of the recursive reasoning process on the initial semantic embedding extracted from the frozen backbone. We show that the model's ability to converge to a correct solution is fundamentally bounded by the information preserved by this embedding, resulting in a theoretical *semantic bottleneck*.

A.2 Fixed-Point Stability and Dependence

We model ReLIT inference as a discrete dynamical system governed by a transition operator F_θ. Let s denote the input prompt and y^* the corresponding ground-truth target. The frozen foundation model $\mathcal{M}$ projects the prompt into a static semantic anchor

$$x = \mathcal{M}(s). \tag{11}$$

Let (y_t, z_t) denote the answer and latent reasoning states at recursion step t. Under the assumption that the spectral radius of the Jacobian of the transition operator satisfies

$$\rho\left(\frac{\partial F_\theta}{\partial(y, z)}\right) < 1, \tag{12}$$

the recursive sequence $\{(y_t, z_t)\}_{t=0}^{\infty}$ converges to a unique fixed point $(y^\dagger, z^\dagger)$ by the Banach Fixed-Point Theorem. The fixed point is defined implicitly as

$$(y^\dagger, z^\dagger) = F_\theta(y^\dagger, z^\dagger; x). \tag{13}$$

Since the initial states y_0 and z_0 are either deterministic functions of x or random noise drawn independently of s, the resulting fixed point is effectively a deterministic function of the anchor x alone. We therefore define the asymptotic mapping

$$y^\dagger = \Phi_\theta(x). \tag{14}$$

This formulation reveals a critical limitation: regardless of the recursive depth, the reasoning capacity of F_θ is restricted to transforming and reparameterizing the information already present in x. The recursion cannot recover information about s that was discarded during the projection $\mathcal{M}(s)$.

A.3 The Information Bottleneck

We formalize this limitation using information-theoretic arguments. Consider the Markov chain induced by the ReLIT architecture:

$$s \;\longrightarrow\; x \;\longrightarrow\; y^\dagger. \tag{15}$$

By the *Data Processing Inequality*, any post-processing of x cannot increase its mutual information with the original prompt s. Consequently, the mutual information between the predicted output and the input is bounded as

$$I(s; y^\dagger) \leq I(s; x). \tag{16}$$

For the model to converge to the correct solution y^*, the anchor x must act as a sufficient statistic of the prompt for the task. Formally, convergence is achievable if and only if the conditional entropy satisfies

$$H(y^* \mid x) \to 0. \tag{17}$$

Equation (A.7) captures the central bottleneck of the ReLIT framework. If the frozen backbone $\mathcal{M}$ is lossy—discarding semantic nuances required to infer y^*—then $H(y^* \mid x) > 0$. In this regime, the fixed point $y^\dagger$ will converge to a suboptimal solution regardless of the recursive depth, as the reasoning engine operates on an incomplete semantic representation.

A.4 Relaxation via Trainable Injection

To mitigate this bottleneck, we consider relaxing the assumption that the semantic anchor x is fixed. By injecting a small set of trainable parameters ψ into the backbone (e.g., via Low-Rank Adaptation), the embedding becomes

$$x_\psi = \mathcal{M}_\psi(s), \tag{18}$$

allowing the representation to adapt to the downstream reasoning task.

The optimization objective then implicitly shifts from minimizing only the task loss to increasing the mutual information between the anchor and the target:

$$\max_{\psi} \; I(x_\psi; y^*). \tag{19}$$

By optimizing ψ, the backbone learns to preserve task-relevant semantic features, effectively reducing the conditional entropy in Eq. (A.7) and ensuring that the fixed point $y^\dagger$ lies within the valid solution manifold.

Alternatively, introducing a cross-attention mechanism that allows the reasoning state to query the prompt representation directly at each recursion step would bypass the bottleneck in Eq. (A.6), effectively transforming the anchor into a dynamic variable x_t.

A.5 Experiments Hyperparameters

Dataset	Nsup	Head	Layers	BS	LR
HELP	8	16	4	16	$8e{-5}$
TaxiNLI	12	16	4	64	$8e{-5}$
NaNLI	16	16	4	16	$8e{-5}$
RuleTaker	6	16	4	128	$1e{-4}$
ProofWriter	3	8	1	256	$2e{-5}$

Table 2: Hyperparameters for Different Datasets

```
FOR (input_tokens, true_labels) IN train_dataloader:
    # 1. FROZEN LLM EMBEDDING EXTRACTION
    WITH NO_GRADIENT:
        frozen_outputs = Frozen_LLM(input_tokens)
        x = frozen_outputs.last_hidden_state  # The base context vector

    # 2. STATE INITIALIZATION
    y, z = initialize_from_x(x)

    FOR step FROM 1 TO N_SUPERVISION:

        # 3. RECURSIVE REFINEMENT
        WITH NO_GRADIENT:
            FOR j FROM 1 TO (T_RECURSION - 1):
                FOR i FROM 1 TO N_LATENT:
                    z =Norm(z+ReLIT_Block(x,y,z)+positional_embeddings)
                    y = Norm(y+ReLIT_Block(y,z))
        FOR i FROM 1 TO N_LATENT:
            z = Norm(z+ReLIT_Block(x,y,z)+positional_embeddings)
            y = Norm(y+ReLIT_Block(y,z))

        # 4. PREDICTION & LOSS
        y_logits = OUTPUT_HEAD(y)
        q_halt   = Q_HEAD(y)

     loss_task = CrossEntropy(y_logits, true_labels)


        # 5. ACT Loss (Learning to predict correctness)
        WITH NO_GRADIENT:
            accuracy = Calculate_Accuracy(y_logits, true_labels)
            target_halt = (accuracy > 0.99)
        loss_act = BinaryCrossEntropyWithLogits(q_logits, target_halt)

        # 6. Dynamic Loss Weighting & Masking
        step_total_loss = loss_task + (0.1 * loss_act)

        # 7. OPTIMIZATION (Per Supervision Step)
        step_total_loss.backward()
        Clip_Gradients(GRAD_CLIP)
        optimizer.step()
        optimizer.zero_grad()

        # 8. STATE MANAGEMENT
        y, z = y.detach(), z.detach()

        # 9. ADAPTIVE HALTING (Update for next step)
        WITH NO_GRADIENT:
            should_halt = (Sigmoid(q_logits) > 0.5) AND (step ≥ 2)
            halted_mask = halted_mask OR should_halt
        # 10. CONVERGENCE CHECK (Optional Early Exit)
        IF Relative_Change(step_total_loss) < LOSS_EPS:
            BREAK step loop
```

Figure 6: **Pseudocode of the ReLIT Recursive Update Process.** The algorithm details the initialization of the latent scratchpad z, the T-step recurrent loop, and the final decoding phase.