BERT/GPT with Inner-Thinking Cycles: Iterative Refinement via Dynamic Head Routing

Bringing Latent Multi-Cycle Reasoning into Standard Transformers

Eran Ben Artzy
eranb92@gmail.com

December 2025

Abstract

Standard Transformers process each input through a fixed number of layers with similar structure of heads, limiting their ability to adapt computation to task difficulty. Meanwhile, latent multi-step reasoning models demonstrate that iterative internal computation can solve algorithmic tasks, but often diverge from mainstream Transformer.

I introduce the *Pointer-over-Heads Transformer* (PoT), a drop-in modification of multi-head attention that adds (i) **dynamic head-wise routing** and (ii) **iterative refinement cycles** and full compatibility with PyTorch `MultiheadAttention`. A two-timescale hierarchical controller, produces per-token routing weights over attention heads, enabling the model to compose different attention patterns across refinement steps.

On the Sudoku-Extreme benchmark, PoT achieves **80% grid accuracy**—a +25 percentage point improvement over the HRM baseline (55%)—and **92% cell accuracy**, outperforming Tiny Recursive Models (87%). The architecture enables BERT/GPT-style models to "think" through multiple internal cycles while maintaining the same parameter count. The full implementation is available at https://github.com/Eran-BA/PoT.

1 Introduction

Transformers [1] have become the dominant architecture across language, vision, and multimodal learning. However, their computation pattern is rigid: each input passes through the same fixed stack of layers regardless of task difficulty. This fixed-depth design limits adaptive computation—simple inputs consume the same resources as complex ones, and multi-step reasoning must be compressed into a single forward pass.

Recent latent reasoning models demonstrate that iterative internal computation can solve algorithmic tasks such as Sudoku-Extreme [3] and maze navigation using relatively few parameters. These models refine an internal state over multiple steps rather than producing explicit chain-of-thought text. However, they often rely on custom architectures that are incompatible with mainstream Transformer implementations and pretrained checkpoints.

This work asks: *Can we add inner-thinking cycles to BERT/GPT-style architectures while preserving parameter parity and implementation compatibility?*

The answer is yes. I introduce the Pointer-over-Heads Transformer (PoT), which enables standard Transformers to perform iterative refinement through dynamic attention head routing. The key insight is that different attention heads can specialize in different reasoning patterns, and a learned controller can compose these patterns across multiple refinement steps—effectively giving the model "thinking time" within a single forward pass.

Contributions.

1. **Pointer-over-Heads (PoT) Attention:** A drop-in replacement for standard multi-head attention that adds dynamic per-token, per-head routing.
2. **Two-Timescale Controller:** A hierarchical routing mechanism with fast (f_{L}) and slow (f_{H}) components that enables compositional multi-step reasoning.
3. **Inner-Thinking Architecture:** Integration of iterative refinement into BERT/GPT-style models, yielding 16 effective reasoning steps per forward pass while maintaining parameter parity.
4. **State-of-the-Art Results:** On Sudoku-Extreme, PoT achieves 80% grid accuracy (+25pp over HRM) and 92% cell accuracy (+5pp over TRM).

2 Background and Related Work

2.1 Fixed-Depth Transformers

Standard Transformers [1] stack multi-head self-attention and feed-forward blocks. Both depth and head aggregation are fixed at architecture design time. While effective for many tasks, this rigidity limits adaptive computation: the model cannot allocate more "thinking" to harder examples.

2.2 Iterative Refinement and Recursive Transformers

Several works explore iterative computation:

- **Adaptive Computation Time (ACT)** [2]: Learns when to halt computation via a differentiable pondering mechanism.
- **Tiny Recursive Models (TRM)**: Reuse blocks to simulate deeper networks with fewer parameters, achieving 87% cell accuracy on Sudoku.

These approaches reuse uniform computation. PoT differs by *routing over attention heads* at each step, allowing different reasoning patterns per iteration.

2.3 Hierarchical Reasoning Model (HRM)

The HRM [3] introduces two-timescale recurrent modules for algorithmic reasoning:

- f_{L} (low-level): Updates every step—fast, reactive computation.
- f_{H} (high-level): Updates every T steps—slow, strategic planning.

HRM achieves 55% grid accuracy on Sudoku-Extreme. PoT adapts this two-timescale design to route over attention heads within standard Transformer blocks, achieving 80% grid accuracy.

2.4 Mixture-of-Experts and Routing

Mixture-of-experts (MoE) models route tokens to different expert sub-networks for efficiency. PoT differs fundamentally:

- Routes over *attention heads*, not experts
- Processes *all tokens* densely (no sparse routing)
- Re-routes at *every refinement step*
- Goal is adaptive reasoning, not computational efficiency

3 Pointer-over-Heads Transformer

3.1 Overview

The PoT architecture adds inner-thinking cycles to standard Transformers through three nested mechanisms:

1. **Refinement Loop:** The model applies its attention blocks R times per forward pass, iteratively refining token representations.
2. **Two-Timescale Controller:** At each refinement step, a hierarchical controller produces routing weights over attention heads.
3. **Weighted Head Aggregation:** Head outputs are combined using learned routing weights rather than uniform concatenation.

This enables the model to "think" through problems by composing different attention patterns across iterations—similar to how humans might approach a puzzle by alternating between local deduction and global pattern recognition.

3.2 Dynamic Head-Wise Routing

Let d_{model} denote the hidden dimension and H the number of attention heads. Given token representations $\mathbf{X}^{(t)} \in \mathbb{R}^{N \times d_{\text{model}}}$ at refinement step t, each head computes standard attention:

$$\text{Head}_k(\mathbf{X}^{(t)}) = \text{Softmax}\left(\frac{\mathbf{X}^{(t)} W_k^Q (\mathbf{X}^{(t)} W_k^K)^\top}{\sqrt{d_k}}\right) \mathbf{X}^{(t)} W_k^V$$

In standard multi-head attention, head outputs are concatenated uniformly. PoT instead combines heads using *per-token, per-step* routing weights $\alpha_{i,k}^{(t)}$:

$$\tilde{h}_i^{(t)} = \sum_{k=1}^{H} \alpha_{i,k}^{(t)} \cdot \text{Head}_k(\mathbf{X}^{(t)})_i \tag{1}$$

where $\sum_k \alpha_{i,k}^{(t)} = 1$ via softmax normalization. Critically, the routing weights α change at each refinement step t, allowing the model to emphasize different heads as reasoning progresses.

3.3 Controller Architectures

The routing weights are produced by a controller that operates across the *depth axis* (refinement iterations), not across the input sequence. We implement two controller variants:

3.3.1 GRU Controller (Default)

A two-timescale hierarchical controller inspired by HRM [3]:

Low-level module (f_{L}). Updates at *every* refinement step, enabling fast, reactive adjustments:

$$z_L^{(t)} = \text{GRU}_L\left([x_{\text{pool}}^{(t)}; z_H^{(t)}],\, z_L^{(t-1)}\right) \tag{2}$$

where $x_{\text{pool}}^{(t)}$ is a pooled summary of token representations and $[\cdot\,;\cdot]$ denotes concatenation.

High-level module (f_{H}). Updates every T steps, maintaining slow, strategic context:

$$z_H^{(t)} = \begin{cases} \text{GRU}_H\left(x_{\text{pool}}^{(t)},\, z_H^{(t-1)}\right) & \text{if } t \bmod T = 0 \\ z_H^{(t-1)} & \text{otherwise} \end{cases} \tag{3}$$

Routing computation. The final routing weights combine token features with controller state:

$$\boldsymbol{\alpha}_i^{(t)} = \text{Softmax}\left(W_r \cdot [x_i^{(t)}; z_L^{(t)}] \,/\, \tau\right) \tag{4}$$

where τ is a learnable temperature parameter.

3.3.2 Causal Depth Transformer Controller

An alternative controller that replaces GRU cells with self-attention over the depth axis, i.e. over the thinking history, Unlike GRUs which only have implicit access to past states through compressed hidden states, the Causal Depth Transformer can *explicitly attend to any previous refinement step*:

1. **Pool tokens:** Compress $\mathbf{X}^{(t)}$ to a single vector $u^{(t)} \in \mathbb{R}^{d_{\text{ctrl}}}$ plus learned depth positional embedding.
2. **Append to cache:** Build sequence $U = [u^{(0)}, u^{(1)}, \dots, u^{(t)}]$.
3. **Causal attention:** Run a Transformer encoder with causal mask—step t attends only to steps $0 \dots t$.
4. **Route per-token:** Combine output $r^{(t)}$ with each token $x_i^{(t)}$ to produce $\alpha_i^{(t)}$.

This architecture offers explicit attention over the depth history and enables parallel training via causal masking. The implementation uses 2 layers, 8 attention heads, and $d_{\text{ctrl}} = 256$.

3.4 Full Architecture

For complex reasoning tasks, we organize PoT blocks into a two-level hierarchy:

- **L-level (fast reasoning):** Runs L_{cycles} iterations per H-cycle
- **H-level (slow reasoning):** Runs H_{cycles} outer iterations

Total refinement steps per forward pass: $R = H_{\text{cycles}} \times L_{\text{cycles}}$.
Each level contains:

- Multiple PoT attention blocks
- RMSNorm for stable training
- SwiGLU feed-forward networks
- Residual connections

Gradient efficiency. Following HRM, only the final iteration receives gradients during training, reducing memory cost while maintaining performance.

4 Sudoku-Extreme Benchmark

4.1 Task Description

Sudoku-Extreme consists of minimal-clue puzzles (17–23 given digits) that require deep logical reasoning. Given a 9×9 grid with blanks (represented as 0), the model must predict all 81 cell values. This task tests multi-step constraint propagation—a capability that fixed-depth models struggle with.

4.2 Dataset

- **Base puzzles:** 10,000 Sudoku-Extreme instances
- **Augmentation:** 100 symmetry/permutation transforms per puzzle
- **Training set:** 1,000,000 augmented examples
- **Validation:** Held-out puzzles with fresh augmentations

4.3 Model Configuration

Table 1: HybridPoHHRM configuration for Sudoku-Extreme

Parameter	**Value**
Hidden dimension (d_{model})	512
Attention heads (H)	8
FFN dimension	2048
H-cycles / L-cycles	2 / 8
H-layers / L-layers	2 / 2
Controller period (T)	4
Controller dimension	256
Total refinement steps	16
Dropout	0.0
Total parameters	∼25.8M

4.4 Training Details

- **Optimizer:** AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$)
- **Learning rate:** 3×10^{-4} with cosine decay
- **Warmup:** 2,000 steps
- **Batch size:** 768
- **Epochs:** 1,000
- **Hardware:** Single NVIDIA A100 GPU (∼10 hours)

5 Results

5.1 Main Results

We compare PoT against two strong baselines on Sudoku-Extreme:

Table 2: Sudoku-Extreme benchmark results. PoT achieves state-of-the-art performance, outperforming HRM by +25 percentage points on grid accuracy.

Model	**Cell Accuracy (%)**	**Grid Accuracy (%)**
Tiny Recursive Models (TRM)	87.0	–
Hierarchical Reasoning Model (HRM)	–	55.0
Pointer-over-Heads (ours)	**92.0**	**80.0**

Key findings.

- PoT achieves **92% cell accuracy**, a +5 percentage point improvement over TRM (87%).
- PoT achieves **80% grid accuracy**, a +25 percentage point improvement over HRM (55%).
- The combination of head-wise routing and two-timescale control enables effective multi-step constraint propagation.

6 Analysis

6.1 Why Does PoT Outperform Baselines?

We hypothesize that PoT's strong performance stems from three factors:

1. **Dynamic head composition:** Different attention heads specialize in different constraint types (row, column, 3×3 box). The learned routing composes these constraints dynamically at each refinement step.
2. **Two-timescale reasoning:** The slow controller (f_H) maintains a global solving strategy, while the fast controller (f_L) handles immediate constraint propagation. This mirrors human Sudoku solving—alternating between local deduction and global pattern recognition.
3. **Sufficient iteration depth:** With 16 refinement steps, PoT has enough "thinking time" for multi-step logical chains without requiring explicit chain-of-thought generation.

6.2 Controller Flexibility

The PoT architecture is agnostic to the specific controller implementation. The codebase provides a unified factory supporting multiple controller types:

- **GRU** (default): Two-timescale HRM-style controller with f_L and f_H modules
- **Causal Depth Transformer**: Explicit attention over thinking history.
- **LSTM**: Standard LSTM with stronger gating than GRU
- **xLSTM**: Extended LSTM with exponential gating [5]

- **minGRU**: Simplified single-gate GRU with fewer parameters

Both the GRU and Causal Depth Transformer controllers achieve comparable performance, validating the architectural flexibility. The transformer controller is preferred due to its stability during training.

7 Conclusion

I presented the Pointer-over-Heads Transformer (PoT), an architecture that brings inner-thinking cycles to BERT/GPT-style models. By introducing dynamic head-wise routing guided by a two-timescale controller, PoT enables iterative refinement within a single forward pass while maintaining full compatibility with standard Transformer implementations.

Key results:

- **80% grid accuracy** on Sudoku-Extreme (+25pp over HRM)
- **92% cell accuracy** (+5pp over TRM)
- Drop-in compatible with PyTorch `MultiheadAttention`

Implications. The success of PoT shows that pretrained language models or any world model could benefit from similar inner-thinking mechanisms. Rather than scaling model size, we can scale "thinking time" by adding iterative refinement with dynamic routing—potentially enabling smaller models to solve harder reasoning tasks.

Acknowledgments

The author thanks Viswanadh Vadlamani for financial support and valuable advice throughout this project.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[2] Alex Graves. Adaptive Computation Time for Recurrent Neural Networks. *arXiv preprint arXiv:1603.08983*, 2016.

[3] Sapient Intelligence. Hierarchical Reasoning Model. *arXiv preprint arXiv:2506.21734*, 2025.

[4] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[5] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended Long Short-Term Memory. *arXiv preprint arXiv:2405.04517*, 2024.

[6] Eran Ben Artzy. Pointer-over-Heads Transformer: Dynamic Multi-Head Attention with Adaptive Routing. Zenodo, 2025. DOI: 10.5281/zenodo.17958199. Code: `https://github.com/Eran-BA/PoT`