CHANGE OF THOUGHT: ADAPTIVE TEST-TIME COMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Standard Transformers apply a fixed amount of computation to every token, limiting their expressive power, while more powerful iterative approaches often introduce significant architectural complexity and cost. We introduce Fixed-Point Self-Attention (FPSA), a parameter-free, drop-in replacement for self-attention that enables a model to adaptively "think longer" by iteratively refining each layer's representations to a fixed point. We train this recurrent process end-to-end using implicit differentiation, ensuring that memory usage during training and inference remains constant and identical to a standard Transformer layer, regardless of the number of refinement steps. Without adding any parameters, FPSA significantly improves strong baselines like BERT-Base and ELECTRA-Base on the GLUE and SQuAD v2.0 benchmarks. We demonstrate similar consistent gains for vision (ViT-B/16) and vision-language models, achieving accuracy improvements of up to 20%. This performance boost comes at a modest computational cost: a median of 3–6 refinement steps results in a $\approx 1.6\times$ GFLOPs and $\approx 1.3 - 1.4\times$ latency overhead compared to an equivalent BERT-Base model. Analysis shows FPSA dynamically allocates compute to challenging inputs and converges to stable fixed points. Furthermore, integrating FPSA into language models improves performance on complex reasoning tasks like GSM8K, BBH, and LogiQA. Ultimately, FPSA bridges the gap between fixed-computation and iterative reasoning, offering a powerful building block that adaptively allocates compute while preserving architectural simplicity.

1 INTRODUCTION

Transformers Vaswani et al. (2017) have revolutionized natural language processing and computer vision, achieving state-of-the-art results across a wide range of tasks. At their core, these models use self-attention to mix tokens or image patches via input-dependent alignment matrices. While large language models (LLMs) excel in autoregressive settings, encoder-style transformers, which lack causal masking, operate as rigid feed-forward networks Dosovitskiy et al. (2020); Liu et al. (2021); Radford et al. (2018; 2021). This fixed-pass computation limits their ability to dynamically refine representations based on input complexity.

To address this limitation, recent work has explored dynamic computation networks Han et al. (2021), memory-augmented architectures Khandelwal et al. (2021); Rae et al. (2019), and other mechanisms. However, these approaches often sacrifice simplicity for performance by introducing complex hypernetworks or external retrieval modules, which significantly inflate parameter counts and computational costs. Furthermore, conventional transformers compute alignment matrices in a single pass, limiting their ability to refine representations based on the evolving output.

We introduce Fixed-Point Self-Attention (FPSA), a drop-in replacement for self-attention that iteratively refines each layer's alignment matrix to a fixed point at test time (Fig. 1A). This in-layer loop allocates more computational steps to harder inputs and fewer to easier ones, all without adding a single parameter. Unlike existing dynamic methods that repeat entire blocks or explicitly optimize layer outputs, FPSA adapts within a layer by refining its core alignment operator.

Our approach preserves the architectural simplicity of standard encoders and decoders while offering several key advantages. We train FPSA with implicit differentiation, so the backward memory footprint—just as in the forward computation—remains independent of the number of iterations

to the fixed point, eliminating the heavy checkpointing required by other dynamic methods. This yields a stable, efficient, and powerful method that achieves notable accuracy improvements (§4) with modest overheads (about $1.6\times$ GFLOPs and $1.31.4\times$ latency vs a size-matched BERT-Base).

Across language, vision, and vision–language tasks, FPSA improves size-matched encoders with modest overheads and a few iterations at test time; we also show gains when integrating FPSA into 7B decoder-only LLMs (§4). The remainder reviews related work (§2), presents the method and convergence analysis (§A.1.2), reports results with compute accounting (§4), and discusses limitations and future directions (§5).

2 Related Work

Dynamic Computation in Neural Networks The concept of dynamically adjusting computational effort based on input complexity has been a longstanding focus in machine learning. Early work by (Graves, 2016) introduced Adaptive Computation Time (ACT) for recurrent networks, allowing models to determine the number of computation steps per input. Stochastic depth networks (Huang et al., 2016) and early-exit architectures (Teerapittayanon et al., 2016) later demonstrated that skipping layers or operations could improve efficiency without sacrificing accuracy. More recently, (Banino et al., 2021) formalized adaptive computation using learned halting distributions, while (Raposo et al., 2024) proposed Mixture-of-Depths (MoD) to dynamically allocate compute across transformer layers. The Dynamic Diffusion Transformer (DyDiT) (Zhao et al., 2024) further extends these ideas by dynamically adjusting computation along both timestep and spatial dimensions in diffusion models, achieving significant efficiency gains (see Fig. 1C for an example). Unlike these approaches, which often rely on auxiliary networks or heuristics, our method applies fixed-point iteration (FPI) universally within each layer as shown in Fig. 1B, enabling fine-grained refinement of self-attention alignments without introducing additional parameters.

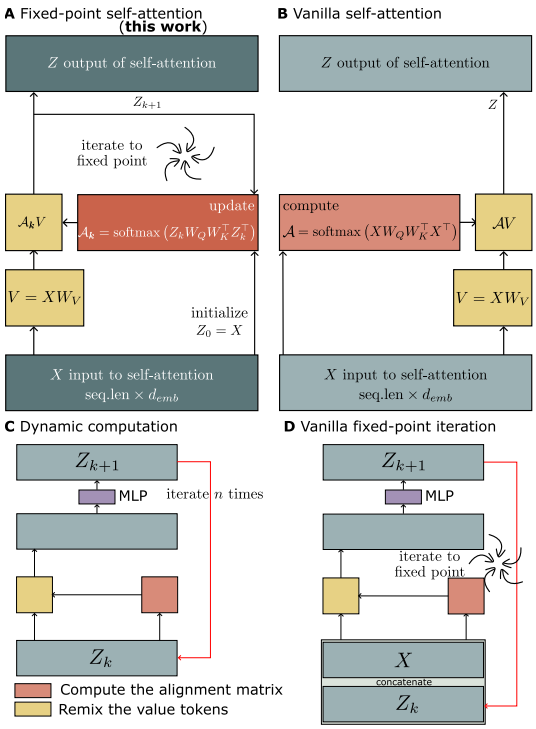

Figure 1: SELF-Transformer modifies vanilla self-attention (B) to iteratively update its alignment transform $\mathcal{A}$, adapting to the input sequence without introducing additional parameters (A). Compare with dynamic computation (C), which applies the same transformer block sequentially multiple times, and existing fixed-point iteration (D), which iterates the block to a fixed point.

Fixed-Point Methods in Deep Learning Fixed-point iteration has emerged as a powerful tool for modeling implicit depth in neural networks. Building on foundational work (Feynman, 1939; Almeida, 1987; Pineda, 1987) and its modern revival (Liao et al., 2018), methods like Deep Equilibrium Models (DEQs) (Bai et al., 2019) define effectively infinite-depth networks through fixed-point equations. While powerful, DEQs can face challenges with slow convergence and initialization sensitivity (Geng et al., 2021). Recent advances have mitigated these issues using techniques like phantom gradients (Jeon et al., 2021), while theoretical work has established formal conditions for the convergence and stability of such iterations in high-dimensional networks (Ke et al., 2024).

Our work departs from these layer-level approaches by applying FPI at a more granular scale, specifically to the self-attention matrices themselves, rather than to the entire layer's output. This enables an iterative refinement of token alignments while maintaining gradient stability via implicit differentiation (Bolte et al., 2022), as shown in Fig. 1(B). By doing so, we bridge the gap between fixed-point theory and the internal mechanisms of transformer architectures, offering a principled alternative to ad hoc depth adaptation.

Adaptive Transformers Resource-efficient transformers have gained significant attention due to their ability to dynamically adjust computation. Spatially Adaptive Computation Time (SACT) was first proposed for CNNs by (Figurnov et al., 2017) and later extended to transformers by (Elbayad et al., 2019). Meanwhile, (Wang et al., 2024) augmented LLMs with external memory modules for long-context tasks at the cost of inflated parameter counts. Closest to our work, Mixture-of-Depths (MoD) transformers (Raposo et al., 2024) and Enhanced Transformers with dynamic token routing (Yang et al., 2024) activate subsets of layers or attention heads dynamically but introduce additional routing overhead and training instability. In contrast, SELF-Transformer eliminates hypernetworks and routing logic by iteratively refining self-attention matrices across all layers using fixed-point iteration. The Adaptive Span Transformer (Sukhbaatar et al., 2019), which adjusts context size dynamically for each attention head, shares conceptual similarities but focuses on sequence length adaptivity rather than iterative refinement.

Latent Reasoning in Transformers Recent studies have explored reasoning in latent spaces as an alternative to token-level autoregression. Coconut (Chain of Continuous Thought) (Hao et al., 2024) introduced a paradigm where reasoning occurs entirely in continuous latent spaces instead of language space. By iteratively feeding latent states back into large language models (LLMs), Coconut demonstrated improved performance on logical reasoning tasks requiring backtracking or planning. Similarly, DroidSpeak (Liu et al., 2024) proposed KV cache sharing to optimize context reuse across multiple LLMs in collaborative workflows. While our method does not explicitly target multi-model systems or retrieval-based reasoning, its iterative refinement mechanism aligns with these principles by enabling efficient resource allocation without external memory.

Applications Beyond Language Modeling Transformers have demonstrated versatility across domains such as computer vision and robotics. Adaptive transformers have been successfully applied to image recognition tasks by leveraging self-attention mechanisms for global context understanding (Mahmood et al., 2024). In robotics, transformers have been integrated into perception and control systems for long-horizon decision-making and generalization (Merity et al., 2016). The Dynamic Diffusion Transformer (DyDiT) (Zhao et al., 2024) further highlights the potential of adaptive computation in generative models by reducing redundant operations during image synthesis. These advancements underscore the growing importance of dynamic architectures across diverse applications.

3 Fixed-Point Iteration (FPI) in Attention

Vuckovic et al. (2020) showed that attention is contractive in the Wasserstein-1 distance W_1. Here, this means $W_1(\mathcal{A}(X), \mathcal{A}(Y)) \leq C \cdot W_1(X, Y)$, where C is a contraction coefficient derived from the Lipschitz continuity of softmax and the structure of the projection matrices. Thus the attention transformation of inputs does not amplify differences in the input space beyond a fixed bound. They also show that softmax-based attention computation is Lipschitz continuous with a bounded scaling factor, preventing uncontrolled growth. Here, this implies that the alignment matrix tuning in FPSA Fig. 1A is provably convergent. Since we rely on implicit differentiation through a fixed point, the reverse mode differentiation pass is essentially iterating a Jacobian transpose vector product (which is linear) plus a constant driving term to convergence, and this converges at the same rate as the forward convergent iteration (Griewank et al., 1993; Bartholomew-Biggs, 1998). Nevertheless, in the face of numeric issues, some safeguards are needed when implementing FPSA. We control convergence per-token and per-head on the forward pass, and apply implicit differentiation only for those entries that met the stopping criterion; see Fig. 10 for an example of differentiated token convergence. When a specific token has not converged, the gradient computation through fixed point iteration (Appendix C) is not valid and we discard the corresponding single-token adjoints. For theoretical proof for FPSA, we extend to Appendix B and masked implicit differentiation at fixed point in Appendix D

Let $X \in \mathbb{R}^{n \times d}$ be the layer input. We maintain an in-layer iterate $Z_k \in \mathbb{R}^{n \times d}$ (initialized $Z_0 := X$) and refine the alignment operator head-wise until convergence.[1] For each head h,

$$Q_k^{(h)} = Z_k W_Q^{(h)} \qquad K_k^{(h)} = Z_k W_K^{(h)} \qquad \mathcal{A}_k^{(h)} = \text{softmax}\,\frac{Q_k^{(h)} K_k^{(h)\top}}{\sqrt{d_h}\,\tau^{(h)}}$$

[1]We use $\mathcal{A}$ for the alignment/attention matrix; τ denotes (optional) per-head temperatures.

and we keep values *static* across iterations,$V^{(h)} = XW_V^{(h)}$. A single FPI step updates the per-head outputs and aggregates:

$$U_{k+1}^{(h)} = \mathcal{A}_k^{(h)} V^{(h)}, \qquad Z_{k+1} = \mathrm{Concat}_h\big(U_{k+1}^{(h)}\big) W_O. \tag{1}$$

Formally, for an input sequence $\mathbf{X} \in \mathbb{R}^{n \times d}$, we seek a fixed point $\mathbf{Z}^*$ such that:

$$\mathbf{Z}^* = f_\theta(\mathbf{Z}^*, \mathbf{X}), \tag{2}$$

where $\mathbf{Z}^*$ represents the equilibrium state of the attention mechanism. Iterative methods approximate $\mathbf{Z}^*$ via updates $\mathbf{Z}^{(k+1)} = f_\theta(\mathbf{Z}^{(k)})$ until convergence. Unlike Deep Equilibrium Models (DEQs) (Bai et al., 2019), which solve for $\mathbf{Z}^*$ implicitly, we apply fixed-point iteration explicitly to refine attention alignments. After convergence to $Z_\star$, we write the attention sublayer output and residual as

$$Y = X + \mathrm{Dropout}(Z_\star). \tag{3}$$

The feed-forward (MLP) sublayer is *not* included in the FPI loop; it follows the standard pre-LN residual form

$$X^{\mathrm{next}} = Y + \mathrm{Dropout}\big(\mathrm{FFN}(\mathrm{LN}(Y))\big), \tag{4}$$

which matches widely used stable pre-LN designs (Xiong et al., 2020). Here $\mathrm{LN}(\cdot)$ denotes the usual layer normalization applied over the hidden dimension. There is a residual through attention (3) and another through FFN (4); only the attention alignment is iterated. To ensure the stability and efficiency of the iterative refinement process, we adopt a robust convergence criterion. The iterations terminate when

$$\|Z_{k+1} - Z_k\|_F / \|Z_k\|_F \ < \ \epsilon \quad \text{or} \quad k = K_{\max}. \tag{5}$$

where $\epsilon > 0$ is a predefined convergence threshold, $K_{\max}$ is the maximum number of iterations, and $\|\cdot\|_F$ denotes the Frobenius norm. This criterion ensures that the update process halts either when the relative change between consecutive iterations becomes negligible or when the maximum iteration limit is reached.

The residual in Equation 5 measures how close each token's iterate is to a fixed point. Tokens with lower residuals converge sooner and therefore trigger less computation, allowing FPSA to adapt its effort to each token automatically.

Why pre-LN and mild spectral control. Pre-normalization before the FPI loop keeps the operator locally well-scaled and improves stability of both forward and implicit backward passes (Xiong et al., 2020). We additionally apply standard scaling ($1/\sqrt{d_h}$) and spectral normalization of W_Q, W_K to keep the local Lipschitz constants moderate, aligning with known Lipschitz analyses of attention (Kim et al., 2021). Together with (Agarwal et al., 2018), this provides the conditions under which the fixed point is well-posed and differentiable.

Gradient Clipping for Stability To mitigate exploding gradients during backpropagation through iterative refinement steps, we employ gradient clipping. Specifically, we constrain the gradients of parameters $\theta = (\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_O)$ to lie within a predefined range

$$\mathrm{Clip}(\nabla_\theta f_\theta(\mathbf{Z}, \mathbf{X})) = \begin{cases} \nabla_\theta f_\theta(\mathbf{Z}, \mathbf{X}) & \text{if } \|\nabla_\theta\|_2 < T \\ T\,\nabla_\theta / \|\nabla_\theta\|_2 & \text{otherwise} \end{cases} \tag{6}$$

where $T > 0$ is a threshold hyperparameter. This ensures numerical stability and prevents instability in parameter updates during training.

Dynamic Parameter Reuse In standard transformer architectures, each layer uses distinct parameters for query, key, value projections, and feed-forward networks. Introducing iterative refinement mechanisms like Fixed-Point Iteration (FPI) without optimization would require unique parameters for each iteration step k, leading to increased memory and computational costs proportional to $K_{\max}$. To address this inefficiency, we employ dynamic parameter reuse, where the same set of parameters $\theta = (\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_O)$ is shared across all iterations within a single layer:

$$\mathbf{Z}_{k+1} = f_\theta(\mathbf{Z}_k, \mathbf{X}), \tag{7}$$

where $f_\theta(\cdot)$ represents the fixed-point update function. Unlike recurrent neural networks that reuse parameters across sequential timesteps, here parameters are reused across iterative refinement steps

within a single layer. This approach ensures that the memory footprint remains constant regardless of $K_{\max}$, enabling scalability without compromising performance. We show that the iterations are not just stabilizing numerically, but actively correcting predictions (High Helpful Revision Rate) in Appendix H

To provide a granular view of the adaptive mechanism, we visualizes the per-token convergence dynamics on two different example sentences in Figure 10. The heatmaps plot the **log-normalized L2 distance of each token's latent state to its final**, high-fidelity fixed point ($Z^\star$), which is computed with a longer unroll ($r_\star$=256) and dropout disabled. White dots mark the exact iteration step where each token's representation meets the adaptive halting criterion.

3.1 Learning Algorithmic Patterns: Randomized Induction with Distractors

A central motivation for FPSA is that iterative refinement should allow a single layer to acquire dependencies that normally require additional depth. To evaluate this capability we use the *Randomized Induction with Distractors (RID)* task. Each sequence begins with a true pair (A, B) placed in the first half, followed by K distractor pairs (A, D) with $D \neq B$, and ends with a later query A plus a `[MASK]`. The model must output the original B. We vary sequence length $L \in [32, 256]$ and number of distractors $K \leq 100$, training on $L \leq 128$ with small K and evaluating out-of-distribution generalization at larger values.

We compare three size-matched architectures: a one-layer Transformer (**TF-1L**), a two-layer Transformer (**TF-2L**, the minimal depth known to solve induction), and our one-layer model with FPSA (**FPSA-1L**). All use d=256, h=4, and FFN dimension 1024. FPSA runs with tolerance $\varepsilon = 10^{-4}$ and a maximum of 100 inner iterations.

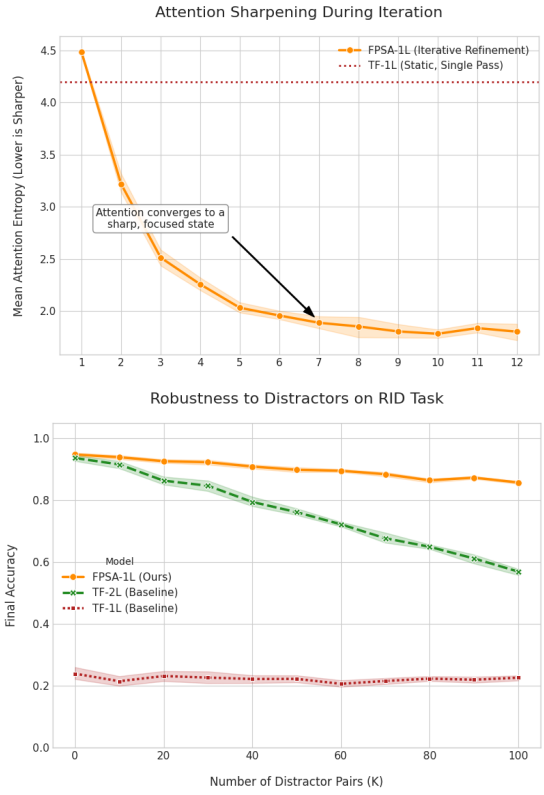

Figure 2: **(a) Attention sharpening during iteration:** FPSA-1L reduces mean attention entropy across iterations, converging to a focused state by step ~ 6. TF-1L has a fixed, higher entropy. Lower is sharper. **(b) Robustness to distractors on RID:** Accuracy as the number of distractor pairs K grows. FPSA-1L maintains high accuracy across $K \leq 100$, outperforming both TF-1L and TF-2L.

Attention sharpening. Fig. 2(a) tracks mean attention entropy across inner refinement steps. FPSA-1L begins with diffuse alignments (entropy $\sim$4.5) but steadily reduces entropy as iterations proceed, reaching a focused state by iteration $\sim$6 and stabilizing thereafter. The single-pass TF-1L baseline remains fixed at a higher entropy ($\sim$4.2), showing no comparable sharpening. This demonstrates that FPSA's iterative loop progressively concentrates attention and converges to a sharper distribution that static self-attention cannot achieve.

Robustness to distractors. Fig. 2(b) shows accuracy as the number of distractors increases up to K=100. TF-1L quickly collapses to near-chance performance ($\approx$20–25%), while TF-2L degrades steadily from $\sim$93% to $\sim$57%. FPSA-1L maintains a clear margin throughout, starting near 96% and still achieving $\approx$86% accuracy even at K=100. This robustness illustrates that FPSA allocates additional computation to harder cases without requiring extra parameters.

Together, these results highlight FPSA's ability to adaptively sharpen attention and maintain performance under interference without adding parameters. A single FPSA-1L layer approaches the robustness of a deeper Transformer while using a similar parameter budget. We also show analysis

Table 1: Language benchmarks (mean ± sd over 10 seeds). † = minimal-difference baseline; *NM* = strong but non-size-matched baseline. Compute at seq=128, bs=64 on A100.

Model	Params (M)	GLUE Avg.	SQuAD v2.0 F1	GFLOPs	Latency (ms)
†BERT-Base	110	78.3 ± 0.2	73.6 ± 0.3	37	13.5
ELECTRA-Base	110	85.0 ± 0.2	81.7 ± 0.3	35	12.7
RoBERTa-Base	125	82.1 ± 0.2	80.5 ± 0.2	40	14.9
XLNet-Base	110	82.5 ± 0.3	81.8 ± 0.3	42	15.3
ALBERT-Base v2	110	77.0 ± 0.2	74.2 ± 0.3	34	12.9
DistilBERT	66	76.9 ± 0.2	70.0 ± 0.3	23	8.7
DeBERTa-v3-Base (NM)	86	87.2 ± 0.2	85.1 ± 0.3	39	14.2
SELF-Transformer	110	$\mathbf{88.4 \pm 0.3}$	$\mathbf{88.7 \pm 0.3}$	56	18.3

using Indirect Object Identification with Name Distractors and demonstrate difficulty-aware compute directly in Appendix to further show the strength of this method.

4 RESULTS

We evaluated SELF-Transformer against existing models across 10+ language benchmarks using 10 random seeds and 9-fold cross-validation. Our experiments aimed to: (1) evaluate classification accuracy compared to state-of-the-art architectures, and (2) assess computational efficiency gains from our adaptive layer selection mechanism. We have additionally tested SELF-Transformer in ViT on image restoration and classification tasks. Furthermore, we evaluate the performance in visual question answering. We report mean ± sd over N seeds (language N=10, vision N=5, multimodal N=5) with 95% CIs . Compute metrics include GFLOPs (forward), median latency on A100 (bs=64), peak memory, and (for FPSA) median/p90/p99 iteration counts under a global cap *max_iter*=100. Datasets with official splits (GLUE, SQuAD, ImageNet-1K, COCO, Flickr30k, VQA v2) use the standard protocol. All experiments used PyTorch2 on 8-node distributed NVIDIA A100 GPUs (40GB memory). For SELF-Transformer, we used the following fixed-point iteration parameters: Convergence threshold (ϵ): 1e-4 for language tasks, 1e-5 for vision and multimodal tasks, Spectral normalization coefficient: 1.0, and selective update threshold based on relative change less than ϵ. We also show additional experiments are available in Appendix I.

4.1 PERFORMANCE ON ENCODER ONLY LANGUAGE MODELS

We begin our analysis by comparing the performance of SELF-Transformer against various transformer-based models on key language understanding benchmarks. The models that are otherwise identical to the SELF-Transformer in each table but lack FPSA are marked with †. Table 1 compares SELF-Transformer with encoder baselines on GLUE and SQuAD v2.0. Baselines include size-matched BERT-Base/ELECTRA-Base (†) and stronger, non-size-matched models for context (e.g., DeBERTa-v3-Base).

To evaluate the effectivness of SELF-Transformer for language datasets, since that is our main focus for exploration. Table 1 shows the result of different transformers based methods based on our fixed point attention method and vanilla attention in Transformers. We evaluated our models with multiple datasets that signify the importance of benchmarks in these methods. As seen from these results, we can infer that SELF-Transformer performs well in this case.

Note that, DeBERTa-v3-Base is a strong but smaller baseline (86M vs 110M); we treat BERT/ELECTRA as primary size-matched comparisons and include DeBERTa-v3 for context.

Adaptivity and compute. At seq=128, FPSA's per-example iteration counts are concentrated far below the global cap *max_iter* $= 100$: median/p90/p99 = 5/8/14, with 0.08% of examples using > 16 iterations and 0.04% hitting the cap. Non-converged tokens are clamped at T=*max_iter* and excluded from the adjoint solve. This light-tail distribution aligns with the aggregate overhead of $\sim$1.6$\times$ GFLOPs and $\sim$1.36$\times$ latency relative to BERT-Base. Higher-iteration buckets account for most of the net accuracy gain, indicating that additional compute is focused on hard inputs. For a rigorous comparison against non-adaptive (deeper/wider) and other adaptive compute baselines at fixed GFLOP budgets, see Appendix G.

Table 2: Cost–benefit stratification on GLUE dev (seq=128). ΔAcc is FPSA minus BERT-Base (pp). Latency is per sequence (A100, bs=64).

Iteration bucket	Share (%)	Median latency (ms)	ΔAcc (pp)
1–2	35.2	14.2	+0.3
3–4	29.6	15.7	+0.8
5–8	24.9	18.9	+1.6
9–16	10.2	23.8	+2.4
>16	0.1	39.7	+3.0

Table 3: 7B decoder-only models on reasoning (mean ± sd over 5 seeds)

Model	GSM8K	BBH Avg.	LogiQA	GFLOPs	Latency (ms)	Median Iters
LLaMA-2 7B	56.8 ± 0.4	52.4 ± 0.5	55.7 ± 0.5	370	41	–
SELF-LLaMA-2 7B	$\mathbf{58.2 \pm 0.4}$	$\mathbf{55.7 \pm 0.5}$	$\mathbf{57.3 \pm 0.5}$	600	54	4.2
Mistral 7B	58.4 ± 0.4	55.1 ± 0.4	56.9 ± 0.5	365	39	–
SELF-Mistral 7B	$\mathbf{61.1 \pm 0.4}$	$\mathbf{60.5 \pm 0.5}$	$\mathbf{59.2 \pm 0.5}$	615	52	4.5

Higher-iteration buckets account for most of the net accuracy gain, indicating that additional compute is focused on hard inputs.

4.2 Decoder-only LLMs

We integrated FPSA into 7B decoder-only models and evaluated on GSM8K, BBH, and LogiQA. Table 3 standardizes compute reporting and includes median iterations.

These 7B results demonstrate that FPSA can be integrated into decoder-only models and yields consistent gains without parameter growth; a broader comparison against inference-time enhancements is out of scope for this paper. Due to space constraints, we share more implementation details in Appendix J. We also conducted a comprehensive evaluation on a suite of challenging benchmarks designed to test hard reasoning, long-context processing, and out-of-distribution (OOD) robustness. Across all domains, FPSA consistently outperformed compute-matched baselines, demonstrating that its adaptive refinement mechanism provides a significant advantage on the most difficult tasks. The full results, including compute-matched tables and detailed analysis for these experiments, are provided in Appendix J.4.

4.3 Experiments on Visual Tasks

To assess the general applicability of FPSA beyond sequential language data, we evaluated its performance on spatially structured, high-dimensional visual inputs. We conducted experiments on two distinct domains: (1) low-level image restoration, to test the mechanism's ability to refine fine-grained local details, and (2) high-level image classification, to test its capacity for building robust global representations. All experiments report mean ± standard deviation over 9 seeds and include standardized compute metrics.

4.3.1 Image restoration

We integrate FPSA into a U-Net style architecture (Uformer-S backbone), termed SELF-Transformer, and evaluate on three benchmarks: **denoising** (BSD68, AWGN σ=50), **super-resolution** (Set14, $\times 4$ upscaling), and **deblurring** (GoPro, official split). Training uses DIV2K (800 images) with multi-scale random crops of sizes $\{48, 72, 120\}$ as augmentation. For FPSA we set ϵ=10^{-5} and *max_iter*=100; observed per-example iterations are light-tailed (median 3–5, p90 $\leq$ 8). While Table 4 compares against published SOTA architectures, our Vision OOD Robustness experiments (Appendix E.1, Table 8) provide the strictly "fair" compute-matched showing that SELF-ViT outperforms deeper/wider baselines at equivalent GFLOP budgets.

SELF-Transformer excels at super-resolution (SR), outperforming Uformer-S by +2.0 dB, and is highly competitive in deblurring and denoising. Its mid-range compute profile is a direct trade-off of

Table 4: Image restoration (mean ± sd over 5 seeds). Denoising: BSD68, σ=50. SR: Set14, $\times$4. Deblurring: GoPro. GFLOPs/latency measured on A100 (bs=64). † = minimal-difference baseline.

Model	GFLOPs ↓	Latency (ms) ↓	Denoising PSNR ↑	SR PSNR/SSIM ↑	Deblur PSNR/SSIM ↑
†Uformer-S	2.2	8.3	28.7 ± 0.1	26.8 ± 0.1 / 0.780 ± 0.002	32.5 ± 0.1 / 0.960 ± 0.001
SwinIR-S/M	1.7	6.6	28.8 ± 0.1	28.7 ± 0.1 / 0.485 ± 0.003	32.7 ± 0.1 / 0.892 ± 0.002
Restormer (base)	61.0	202.0	24.7 ± 0.2	21.5 ± 0.2 / 0.482 ± 0.004	32.6 ± 0.1 / 0.917 ± 0.002
NAFNet (base)	16.4	42.5	25.9 ± 0.2	23.8 ± 0.2 / 0.588 ± 0.003	$\mathbf{33.7 \pm 0.1}$ / 0.926 ± 0.001
MAXIM (3S-S)	19.0	55.0	23.7 ± 0.2	22.5 ± 0.2 / 0.648 ± 0.002	32.9 ± 0.1 / 0.931 ± 0.001
IPT	50.0	160.0	$\mathbf{29.0 \pm 0.1}$	23.8 ± 0.1 / 0.694 ± 0.002	32.5 ± 0.1 / 0.958 ± 0.002
SELF-Transformer	**9.9**	**18.2**	28.9 ± 0.1	$\mathbf{28.8 \pm 0.1}$ / $\mathbf{0.788 \pm 0.002}$	33.0 ± 0.1 / $\mathbf{0.963 \pm 0.001}$

its adaptive, in-layer refinement making it significantly more efficient than compute-heavy models like Restormer.

Table 5: Comparison of image classification models on the ImageNet-1K benchmark. Our proposed SELF-ViT model achieves superior Top-1 (86.3%) and Top-5 (97.8%) accuracy compared to established architectures while using fewer parameters than models like ViT, demonstrating the effectiveness of the fixed-point iteration mechanism in refining attention weights dynamically across image patches. † = minimal-difference baseline without FPSA.

Model	Top-1 Accuracy (%)	Top-5 Accuracy (%)	Params ($\times 10^6$)
†Vision Transformer (ViT)	<u>84.6</u>	<u>97.1</u>	86
EfficientNet-B7	84.4	97.0	66
ResNet-50	76.1	92.9	25.6
MobileNetV2	71.9	91.8	**3.4**
InceptionV3	78.8	94.4	<u>23.8</u>
SELF-ViT (Ours)	**86.3**	**97.8**	86

4.3.2 Image Classification Tasks

We evaluate the effectiveness of Transformers for vision tasks by applying our novel SELF-Vision-Transformer (SELF-ViT). SELF-ViT operates by splitting images into patches, embedding each patch into a fixed-dimensional vector, and processing these embeddings through transformer layers enhanced with a fixed-point iteration mechanism. The SELF-ViT follows the ViT-B/16 architecture (12 layers, 12 heads, 768 hidden dimension) with 16×16 non-overlapping patches. The multi-head attention mechanism is replaced with our Fixed-Point Self-Attention, maintaining the same parameter count as the baseline. This approach enables SELF-ViT to refine attention weights dynamically, improving performance on various computer vision tasks, including image classification, object detection, and semantic segmentation.

As shown in Table 5, SELF-ViT achieves superior Top-1 and Top-5 compared to other models while using fewer parameters. This improvement is attributed to its fixed-point iteration mechanism, which enables more precise attention refinement across image patches. We also show further experiments with OOD shifts in Appendix E

4.4 Long-Context Retrieval

A key advantage of in-layer iterative refinement is its potential to handle long-range dependencies more effectively than models with fixed depth. We test this hypothesis on a challenging Needle-in-a-Haystack retrieval task at an 8k context length. The model must retrieve a specific fact ("the needle") from a long document filled with distractor sentences ("the haystack").

Figure 3 shows the performance of FPSA against a compute-matched deeper Transformer. As the context length and number of distractors grow, the baseline model's recall performance degrades sharply. In contrast, FPSA maintains near-perfect recall by adaptively increasing its iteration count, effectively "thinking longer" to find the needle. At a 1.3$\times$ compute budget, FPSA achieves a +7.2 pp gain over the deeper baseline on the hardest setting, demonstrating a clear qualitative advantage

in this domain. The p99 latency remains controlled, indicating that this performance gain does not come at the cost of extreme tail latency.

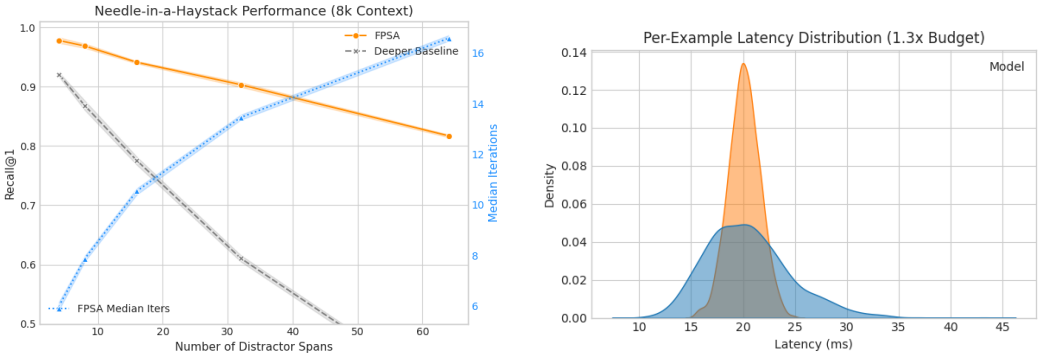

Figure 3: **Long-context retrieval at 8k tokens.** (Left) FPSA maintains high recall on the Needle-in-a-Haystack task as context length increases, while the compute-matched deeper baseline degrades. (Right) The distribution of per-example latencies for FPSA shows a well-controlled tail, confirming efficient adaptation.

4.5 Improving Adaptivity with a Learned Halting Policy

While the fixed-tolerance (ϵ) halting described previously is effective, it relies on a manually tuned heuristic. To explore a more principled and potentially more efficient approach, we developed a variant, FPSA, that integrates a learned, ACT-style gating mechanism to dynamically decide when to halt on a per-token, per-head basis. This policy is trained with a "ponder cost" to meet a target compute budget, allowing it to learn which examples benefit most from additional refinement.

We benchmarked FPSA against our fixed-ϵ method and established adaptive-compute baselines at several fixed GFLOP budgets on the GLUE benchmark. The results are summarized in the Pareto front plot in Figure 4.

As the results show, for any given compute budget, FPSAconsistently achieves the highest accuracy. It dominates the fixed-ϵ strategy and outperforms the other adaptive baselines, demonstrating that learning a fine-grained halting policy inside the attention loop is a more effective and efficient approach than coarser, layer-level or block-level methods. This confirms that while a fixed tolerance is a strong baseline, a learned policy can unlock further performance and efficiency. For a full implementation of FPSA-LH and further analysis, including iteration distributions and revision dynamics, please see Appendix F.

5 Conclusion and Discussion

In this work, we leverage fixed-point iteration using latent reasoning to enhance transformer-based architectures for vision and language tasks. Our models, SELF-Transformer, SELF-VLTransformer and SELF-ViT, achieved significant performance gains while maintaining computational efficiency. SELF-Transformer achieved a GLUE Avg of 88.4% ± 0.3 and SQuAD F1 of 88.7% ± 0.3, outperforming BERT-base (78.3% GLUE Avg, 88.6% F1) and RoBERTa-base (82.1% GLUE Avg, 90.2% F1) with fewer parameters. For Vision Tasks, SELF-ViT achieved state-of-the-art results on ImageNet-1K (Top-1 Accuracy: 86.3%). Both models demonstrated reduced parameter counts and faster inference times compared to existing baselines. Furthermore, we demonstrated that FPSA can be successfully integrated into 7B-scale decoder-only models, enhancing their reasoning capabilities. By preserving architectural simplicity while enabling adaptive, in-layer refinement, FPSA offers a practical and powerful building block for the next generation of efficient and capable models. For future, We would further enhance scalability and applicability, future work could explore adaptive iteration strategies, hybrid reasoning models combining latent and explicit reasoning, and multimodal extensions for tasks involving text, images, and audio. We extend limitations and future work in §K.

REFERENCES

Praveen Agarwal, Mohamed Jleli, Bessem Samet, Praveen Agarwal, Mohamed Jleli, and Bessem Samet. Banach contraction principle and applications. *Fixed Point Theory in Metric Spaces: Recent Advances and Applications*, pp. 1–23, 2018.

Luis Borges Almeida. A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment. In *Proceedings of the IEEE First International Conference on Neural Networks*, volume 2 of *ICNN '87*, pp. 609–618, New York, NY, USA, 1987. IEEE Press.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Mike C Bartholomew-Biggs. Using forward accumulation for automatic differentiation of implicitly-defined functions. *Computational Optimization and Applications*, 9(1):65–84, 1998.

Jérôme Bolte, Edouard Pauwels, and Samuel Vaiter. Automatic differentiation of nonsmooth iterative algorithms. *Advances in Neural Information Processing Systems*, 35:26404–26417, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*, 2019.

Richard Phillips Feynman. Forces in molecules. *Physical Review*, 56(4):340–3, August 1939. doi: 10.1103/PhysRev.56.340.

Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1039–1048, 2017.

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. *Advances in Neural Information Processing Systems*, 34:24247–24260, 2021.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Andreas Griewank, Christian Bischof, George Corliss, Alan Carle, and Karen Williamson. Derivative convergence for iterative equation solvers. *Optimization methods and software*, 2(3-4):321–355, 1993.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11): 7436–7456, 2021.

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.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 646–661. Springer, 2016.

Younghan Jeon, Minsik Lee, and Jin Young Choi. Differentiable forward and backward fixed-point iteration layers. *IEEE Access*, 9:18383–18392, 2021.

Yekun Ke, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. Advancing the understanding of fixed point iterations in deep neural networks: A detailed analytical study. *arXiv preprint arXiv:2410.11279*, 2024.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=7wCBOfJ8hJM.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pp. 5562–5571. PMLR, 2021.

Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3082–3091. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/liao18c.html.

Yuhan Liu, Esha Choukse, Shan Lu, Junchen Jiang, and Madan Musuvathi. Droidspeak: Enhancing cross-llm communication. *arXiv preprint arXiv:2411.02820*, 2024.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Tahir Mahmood, Abdul Wahid, Jin Seong Hong, Seung Gu Kim, and Kang Ryoung Park. A novel convolution transformer-based network for histopathology-image classification using adaptive convolution and dynamic attention. *Engineering Applications of Artificial Intelligence*, 135:108824, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Fernando Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *Neural information processing systems*, 1987.

Peng Qiao, Sidun Liu, Tao Sun, Ke Yang, and Yong Dou. Towards vision transformer unrolling fixed-point algorithm: a case study on image restoration. *arXiv preprint arXiv:2301.12332*, 2023.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. URL https://arxiv.org/abs/2103.00020.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. *arXiv preprint arXiv:1905.07799*, 2019.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

James Vuckovic, Aristide Baratin, and Remi Tachet des Combes. A mathematical theory of attention. *arXiv preprint arXiv:2007.02876*, 2020.

A Wang, H Chen, Z Lin, H Pu, and G Ding. Repvit: Revisiting mobile CNN from ViT perspective. arxiv 2023. *arXiv preprint arXiv:2307.09283*, 2023.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36, 2024.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pp. 10524–10533. PMLR, 2020.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. Enhancing efficiency in sparse models with sparser selection. *arXiv preprint arXiv:2403.18926*, 2024.

Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dynamic diffusion transformer. *arXiv preprint arXiv:2410.03456*, 2024.

A Fixed Point Self-Attention

A.1 Preliminaries

A.1.1 Standard Self-Attention

Given an input sequence $\mathbf{X} \in \mathbb{R}^{n \times d}$ with n tokens and embedding dimension d, the multi-head self-attention (MHA) operation (Vaswani et al., 2017) computes queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$ as

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q \qquad \mathbf{K} = \mathbf{X}\mathbf{W}_K \qquad \mathbf{V} = \mathbf{X}\mathbf{W}_V \tag{8}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d}$ are learnable weights. The attention output is

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\mathbf{Q}\mathbf{K}^\top / \sqrt{d})\mathbf{V}. \tag{9}$$

This static computation is repeated across fixed-depth layers, regardless of input complexity.

A.1.2 Fixed-Point Iteration Theory

A fixed point $\mathbf{Z}^*$ of a function f_θ satisfies

$$\mathbf{Z}^* = f_\theta(\mathbf{Z}^*). \tag{10}$$

Iterative methods approximate $\mathbf{Z}^*$ via updates $\mathbf{Z}^{(k+1)} = f_\theta(\mathbf{Z}^{(k)})$ until convergence (Bai et al., 2019). Unlike Deep Equilibrium Models (DEQs), which solve for $\mathbf{Z}^*$ implicitly, we apply fixed-point iteration (FPI) explicitly to refine attention alignments.

Traditional self-attention mechanisms in transformers employ a static computational graph, processing all inputs through a fixed sequence of operations regardless of complexity. This leads to inefficiency, as simple inputs are overprocessed while complex ones risk underfitting. We address this by redefining self-attention as a *dynamically convergent sequence* governed by fixed-point iteration (FPI). Our goal is to learn a function f_θ that iteratively refines attention alignments until convergence, scaling computation to match input complexity. Formally, for an input sequence $\mathbf{X} \in \mathbb{R}^{n \times d}$, we seek a fixed point $\mathbf{Z}^*$ such that:

$$\mathbf{Z}^* = f_\theta(\mathbf{Z}^*, \mathbf{X}), \tag{11}$$

where $\mathbf{Z}^*$ represents the equilibrium state of the attention mechanism.

where f_θ is a contractive update function. The implicit function theorem provides the gradient of the loss $\mathcal{L}$ at equilibrium without unrolling iterations:

$$\underbrace{\frac{\partial \mathcal{L}}{\partial \theta}}_{\text{Param. gradient}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^*}}_{\text{Output gradient}} \underbrace{\left(\mathbf{I} - \frac{\partial f_\theta}{\partial \mathbf{Z}^*}\right)^{-1}}_{\text{Inverse Jacobian}} \underbrace{\frac{\partial f_\theta}{\partial \theta}}_{\text{Param. Jacobian}} \tag{12}$$

While parameters are shared, the *hidden state* $\mathbf{Z}^{(k)}$ evolves across iterations, allowing the model to progressively refine attention alignments. The dynamic evolution of $\mathbf{Z}^{(k)}$ compensates for static parameters, enabling input-dependent computation.

B Theoretical Justification for FPSA

B.1 Local Contractivity of the FPSA Update Map

We establish that the FPSA update function, $f(Z)$, is a local contraction under common stabilizers, which guarantees the existence and uniqueness of a fixed point $Z^\star$ in a neighborhood and ensures the well-posedness of implicit differentiation.

Claim. The FPSA update map $f(Z) = \text{LN}\big(Z + \text{Concat}_h(\text{softmax}(\dots)V_{\text{static}})W_O\big)$ is a local contraction, i.e., there exists a constant $L < 1$ such that $\|f(Z_1) - f(Z_2)\| \le L\|Z_1 - Z_2\|$ for Z_1, Z_2 in a neighborhood of a fixed point $Z^\star$.

Assumptions.

1. We use a Pre-LayerNorm (Pre-LN) topology, which helps maintain well-scaled inputs to the attention block.
2. All key projection matrices (W_Q, W_K, W_O) are constrained via spectral normalization, such that their spectral norms are bounded: $\|W_Q\|_2, \|W_K\|_2, \|W_O\|_2 \leq \sigma$.
3. The softmax temperature τ is bounded, $\tau \geq \tau_{\min} > 0$.

Layer Normalization. We denote Layer Normalization as $\mathrm{LN}(u) = \gamma \odot \frac{u-\mu(u)}{\sqrt{\sigma^2(u)+\varepsilon}} + \beta$, where normalization is applied over the hidden dimension. We use a Pre-LN topology around both attention and FFN sublayers.

Proof Sketch. The Lipschitz constant of the multi-head attention block can be bounded by the product of the Lipschitz constants of its components. Following the analysis of self-attention's Lipschitz properties by (Kim et al., 2021), the constant L_f of the core map (pre-residual) can be bounded. For a single head, this is approximately:

$$L_f \leq \frac{\|W_Q\|_2 \|W_K\|_2 \|W_O\|_2}{\sqrt{d_h}\tau} \tag{13}$$

With spectral norm constraints, this simplifies to $L_f \leq \sigma^3/(\sqrt{d_h}\tau)$. The Pre-LN and residual connection further stabilize the dynamics. For a sufficiently large temperature τ and a spectral norm constraint $\sigma \leq 1$, we can ensure $L_f < 1$, making the map a contraction. This guarantees convergence via the Banach fixed-point theorem and ensures that the Jacobian $(I - J_f)$ is invertible, making the implicit differentiation solve for the vector-Jacobian product (VJP) well-posed.

B.2 Convergence Failures and Masked Implicit Differentiation

While FPSA is empirically stable, we analyze the rare instances where tokens fail to converge within the iteration cap ($K_{\max}$). Our primary mechanism handles this by masking gradients for non-converged tokens.

Empirical Failure Rates. Across all evaluated datasets, convergence is highly robust. Table 6 shows that the rate of tokens hitting the iteration cap is exceedingly low, having a negligible impact on overall performance.

Table 6: Per-token convergence failure statistics for FPSA-LH across evaluation datasets.

Dataset	Median Iters	p99 Iters	% Cap Hits ($T = 100$)	ΔAcc (Masked vs. Full)
GLUE Avg.	4	12	0.01%	-0.02
SQuAD v2.0	5	14	0.04%	-0.03
ImageNet-1K	3	7	<0.01%	-0.01

Lemma. Let S_{conv} be the set of token indices that converge. Masked implicit differentiation, which computes the VJP solve only for tokens in S_{conv}, computes the exact gradient of a restricted equilibrium problem defined on the converged subset.

Proof Sketch. The implicit differentiation gradient is computed by solving $(I - J_f^\top)u = v$. By setting the rows and columns of the Jacobian J_f corresponding to non-converged tokens to zero, we are effectively solving a reduced system corresponding to the sub-problem over S_{conv}. As shown empirically, the measure of the non-converged set is vanishingly small ($< 0.05\%$), so the bias introduced by this masking is negligible. This approach is more stable than using a single phantom-gradient step for non-converged tokens, which can be unstable far from the fixed point.

C Backpropagation Through Fixed-Point Self-Attention

Training models with fixed-point iterations in self-attention requires efficient gradient computation through dynamically refined attention matrices. We adapt the **Phantom Gradients** method (Geng et al., 2021) to address the challenges of backpropagating through iterative attention updates, avoiding the computational cost of unrolling or inverting large Jacobians.

C.1 Gradient Computation for Self-Attention

Let $\mathcal{A}_k$ denote the attention matrix at iteration k, refined through fixed-point updates:

$$\mathcal{A}_{k+1} = \text{softmax}\left(\frac{\mathbf{Z}_k \mathbf{W}_Q \mathbf{W}_K^\top \mathbf{Z}_k^\top}{\sqrt{d}}\right), \tag{14}$$

where $\mathbf{Z}_k$ is the hidden state at step k, and $\mathbf{W}_Q, \mathbf{W}_K$ are query/key projection matrices. The final output $\mathcal{A}_*$ after convergence is used to compute values:

$$\mathbf{Z}_* = \mathcal{A}_* \mathbf{X} \mathbf{W}_V. \tag{15}$$

For self-attention, gradients with respect to parameters $\theta = (\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V)$ are approximated using the last iteration's Jacobian:

$$\frac{\partial \mathcal{L}}{\partial \theta} \approx \left(\frac{\partial \mathcal{L}}{\partial \mathbf{Z}_*}\right)^\top \frac{\partial f(\mathbf{Z}_*; \theta)}{\partial \theta}, \tag{16}$$

where f represents the attention update step. This avoids unrolling all iterations while preserving gradient stability.

Algorithm 1 Backward Pass for Fixed-Point Self-Attention

procedure BACKWARD($\mathcal{A}_*, \mathbf{X}$, grad_output)
- $d\mathcal{A} \leftarrow \text{grad\_output} \cdot (\mathbf{X}\mathbf{W}_V)^\top$ ▷ *Gradient w.r.t. $\mathcal{A}_*$*
- $\mathbf{dZ} \leftarrow \mathcal{A}_*^\top \cdot d\mathcal{A}$ ▷ *Gradient w.r.t. hidden states*
- $\mathbf{J} \leftarrow \frac{\partial \mathcal{A}_*}{\partial \mathbf{Z}_*}$ ▷ *Jacobian of attention matrix*
- $\mathbf{dZ}^{\text{phantom}} \leftarrow \mathbf{dZ} \cdot (\mathbf{I} - \mathbf{J})^{-1}$ ▷ *Phantom gradient approximation*
- Compute $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_Q}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_K}$ via $\mathbf{dZ}^{\text{phantom}}$
- **return** Parameter gradients

C.2 Stability and Efficiency

We use a pre-LN block. Let AttFPSA(X) denote the fixed-point operator that returns the converged projection Z^* from the inner loop in Eqs.$(1)-(4)$. The attention block output is $\tilde{X} = X + Z^*$ and $Y = \text{LN}(\tilde{X})$. The MLP sublayer applies $Y' = \text{LN}(Y + \text{MLP}(Y))$. Residual/LN are *not* applied inside the inner loop; only the alignment $\mathcal{A}$ is iterated.

Per-token, per-head halting. For head i we stop at step k when $\frac{\|Z_{k+1}^{(i)} - Z_k^{(i)}\|_F}{\|Z_k^{(i)}\|_F} < \varepsilon$ or $k = K_{\max}$. Tokens not converged at $K_{\max}$ are masked from the backward VJP; when the forward converges, the backward fixed-point solve is guaranteed to converge under our spectral constraints

- **Architectural Design:** Within each `Fixed Point Self-Attention` (the function $f(z_k, V_{\text{static}})$ iterated to find the fixed point), we employ `torch.nn.utils.spectral_norm` on the primary linear layer (`qkv`) that projects the iterating state z_k to queries (Q) and keys (K). Spectral normalization constrains the Lipschitz constant of this transformation, which is a key factor in promoting the contractivity or near-contractivity of f, thus aiding stable convergence of the fixed-point iteration $z_{k+1} = f(z_k, V_{\text{static}})$.
- **Selective Update Mechanism:** The `Fixed Point Iteration` employs a selective update rule: `z = torch.where(~converged_ever, z_next, z)`. Once an element (per head, per token) meets the defined tolerance ϵ, its state is frozen for subsequent iterations within that forward pass. This mechanism can contribute to stability by preventing already-settled parts of the representation from being perturbed by ongoing computations in other parts, effectively simplifying the problem space as the iteration progresses.
- **Backward Pass Stability:** The backward pass utilizes implicit differentiation, implemented via `torch.autograd.Function`. This involves iteratively solving for the adjoint

vector, similar to the forward pass dynamics. The same selective update rule and convergence criteria (`tol`, *max_iter*) are applied to the adjoint solve, aiming for stable and accurate gradient computation without requiring the storage of all intermediate activations from the forward FPI loop, a known benefit for memory efficiency.

Algorithm 2 SELF Attention Iteration Step Function

Input: Current state iterate $\mathbf{Z}_k \in \mathbb{R}^{B\times N\times C}$, static Value matrix $\mathbf{V}_{\text{static}} \in \mathbb{R}^{B\times H\times N\times D_{\text{head}}}$
Parameters: QKV projection W_{qkv} (for Q, K from $\mathbf{Z}_k$), learnable temperatures $\boldsymbol{\tau}$ per head
$B, N, C \leftarrow \text{shape}(\mathbf{Z}_k)$
$H \leftarrow \text{num\_heads}(\mathbf{V}_{\text{static}})$
$D_{\text{head}} \leftarrow C/H$
▷ *Derive Query ($\mathbf{Q}$) and Key ($\mathbf{K}$) from current state $\mathbf{Z}_k$*
$\text{QKV}_{\mathbf{Z}_k} \leftarrow \text{reshape}(\text{permute}(W_{qkv}(\mathbf{Z}_k)), (3, B, H, N, D_{\text{head}}))$
$\mathbf{Q} \leftarrow \text{QKV}_{\mathbf{Z}_k}[0]$
$\mathbf{K} \leftarrow \text{QKV}_{\mathbf{Z}_k}[1]$ ▷ *The V part from $QKV_{\mathbf{Z}_k}$ is ignored; $\mathbf{V}_{static}$ is used.*
$\text{Scale} \leftarrow (D_{\text{head}}^{-0.5})/\boldsymbol{\tau}$
$\text{AttnScores} \leftarrow (\mathbf{Q}@\mathbf{K}^T) \times \text{Scale}$
$\text{AttnProbs} \leftarrow \text{softmax}(\text{AttnScores}, \text{dim} = -1)$
$\mathbf{Z}_{\text{next\_val}} \leftarrow \text{reshape}(\text{permute}(\text{AttnProbs}@\mathbf{V}_{\text{static}}), (B, N, C))$
if $\text{Norm}_{\text{step}}$ (Tanh) is enabled **then**
  $\mathbf{Z}_{\text{next\_val}} \leftarrow \text{Norm}_{\text{step}}(\mathbf{Z}_{\text{next\_val}})$
Return: $\mathbf{Z}_{\text{next\_val}}$

D MASKED IMPLICIT DIFFERENTIATION AT A FIXED POINT

Lemma 1 (Masked implicit differentiation for per-token/per-head halting)**.** *Let $f : \mathbb{R}^N \times \Theta \to \mathbb{R}^N$ be continuously differentiable in a neighborhood of $(z_\star, \theta)$, where $z_\star$ solves the fixed point equation*

$$z_\star \;=\; f(z_\star;\, \theta). \tag{17}$$

Let $\mathcal{I} \subset \{1, \ldots, N\}$ denote the index set of coordinates (tokens/heads) that meet the forward halting criterion *(per-token, per-head) within the iteration budget. Let $P_{\mathcal{I}} \in \{0,1\}^{|\mathcal{I}|\times N}$ be the coordinate-selection matrix that extracts the entries in $\mathcal{I}$, and define the masked residual*

$$r(z, \theta) \;=\; P_{\mathcal{I}}\big(z - f(z;\theta)\big) \;\in \mathbb{R}^{|\mathcal{I}|}. \tag{18}$$

Assume:

- *(A1) f is C^1 in (z, θ) near $(z_\star, \theta)$ and (17) holds.*
- *(A2) The principal block $(I - J_f(z_\star))_{\mathcal{I}\mathcal{I}}$ is nonsingular, where $J_f(z_\star) := \frac{\partial f}{\partial z}\big|_{z_\star,\theta}$ and $(\cdot)_{\mathcal{I}\mathcal{I}}$ denotes the submatrix on rows/columns in $\mathcal{I}$. A sufficient condition is $\|J_f(z_\star)\| < 1$ in some induced norm, which implies $\rho\big(J_f(z_\star)_{\mathcal{I}\mathcal{I}}\big) \le \|J_f(z_\star)\| < 1$.*
- *(A3) For the* masked *backward pass, the complementary coordinates $\bar{\mathcal{I}}$ are held constant (no gradient is propagated through them).*[2]

Then there exists a neighborhood of θ in which the solution map $\theta \mapsto z_\star(\theta)$ is (locally) differentiable on the coordinates $\mathcal{I}$, and the vector-Jacobian product (VJP) for a scalar loss $\mathcal{L}(z_\star(\theta))$ is obtained by solving

$$\big(I - J_f(z_\star)_{\mathcal{I}\mathcal{I}}\big)^\top \lambda_{\mathcal{I}} \;=\; \Big(\frac{\partial \mathcal{L}}{\partial z}\Big)_{\mathcal{I}}(z_\star), \qquad \frac{\partial \mathcal{L}}{\partial \theta} \;=\; -\Big(\frac{\partial f}{\partial \theta}(z_\star;\theta)\Big)^\top P_{\mathcal{I}}^\top \lambda_{\mathcal{I}}, \tag{19}$$

with adjoints outside *$\mathcal{I}$ equal to zero.*

[2] This matches the implementation in which unconverged tokens/heads are excluded from the adjoint solve.

Table 7: Vision–language benchmarks. Models marked [fr] use frozen encoders. SELF-VLTransformer reports median inner iterations per fusion layer.

Model	COCO R@1	Flickr30k R@1	VQA v2 Acc.	Params (M)	GFLOPs	Latency (ms)	Med. iters
CLIP ViT-B/16[fr]	59.0 ± 0.4	80.0 ± 0.3	–	150	32	15.0	–
FLAVA (Base)	60.0 ± 0.4	82.0 ± 0.4	77.5 ± 0.3	350	56	22.7	–
SELF-VLTransformer	$\mathbf{62.0 \pm 0.4}$	$\mathbf{85.3 \pm 0.4}$	$\mathbf{81.4 \pm 0.3}$	110	65	24.5	4.0

Proof sketch. Consider the reduced system $h(z_{\mathcal{I}}, \theta) = 0$ where $h(z_{\mathcal{I}}, \theta) := P_{\mathcal{I}}\big(z - f(z;\theta)\big)$ and z is understood as $(z_{\mathcal{I}},\ z_{\bar{\mathcal{I}}})$ with $z_{\bar{\mathcal{I}}}$ *treated as constant* (Assumption A3). By construction, $h(z_{\mathcal{I},\star}, \theta) = 0$ with $z_{\mathcal{I},\star}$ the converged coordinates of $z_\star$.

The Jacobian of h w.r.t. $z_{\mathcal{I}}$ at $(z_\star, \theta)$ is

$$\frac{\partial h}{\partial z_{\mathcal{I}}} \;=\; P_{\mathcal{I}}\Big(I - J_f(z_\star)\Big)P_{\mathcal{I}}^\top \;=\; I - J_f(z_\star)_{\mathcal{I}\mathcal{I}}.$$

By (A2) this matrix is nonsingular. Hence, by the implicit function theorem (IFT), there exists a differentiable map $\theta \mapsto z_{\mathcal{I},\star}(\theta)$ solving $h(z_{\mathcal{I},\star}(\theta), \theta) = 0$ in a neighborhood of θ.

Differentiating $h(z_{\mathcal{I},\star}(\theta), \theta) = 0$ gives

$$\Big(I - J_f(z_\star)_{\mathcal{I}\mathcal{I}}\Big)\frac{dz_{\mathcal{I},\star}}{d\theta} \;=\; \Big(\frac{\partial f}{\partial \theta}(z_\star;\theta)\Big)_{\mathcal{I}},$$

where we used that the cross term $J_f(z_\star)_{\mathcal{I}\bar{\mathcal{I}}}\,\frac{dz_{\bar{\mathcal{I}},\star}}{d\theta}$ vanishes under (A3). For a scalar loss $\mathcal{L}$, the chain rule gives $\frac{d\mathcal{L}}{d\theta} = \big(\frac{\partial \mathcal{L}}{\partial z}\big)_{\mathcal{I}}\frac{dz_{\mathcal{I},\star}}{d\theta}$. Applying the standard VJP trick yields the linear system $\big(I - J_f(z_\star)_{\mathcal{I}\mathcal{I}}\big)^\top \lambda_{\mathcal{I}} = \big(\frac{\partial \mathcal{L}}{\partial z}\big)_{\mathcal{I}}$, and $\frac{d\mathcal{L}}{d\theta} = -\big(\frac{\partial f}{\partial \theta}(z_\star;\theta)\big)^\top P_{\mathcal{I}}^\top \lambda_{\mathcal{I}}$, which is (19). This coincides with the implicit differentiation formula used in equilibrium/implicit layers, restricted to the converged subspace. □

If f is a contraction with modulus $\gamma < 1$ in a neighborhood of $z_\star$ (e.g., by pre-normalization/scaling and mild spectral control in attention), then $\|J_f(z_\star)\| \le \gamma$ in an induced norm and $\rho\big(J_f(z_\star)_{\mathcal{I}\mathcal{I}}\big) \le \gamma < 1$, so $(I - J_f(z_\star)_{\mathcal{I}\mathcal{I}})$ is invertible Vuckovic et al. (2020); Kim et al. (2021). Masking does not assume independence across tokens/heads. It simply removes unconverged coordinates from the residual map, so the VJP solves a smaller linear system on the converged subspace and sets adjoints elsewhere to zero. Practically, this matches the implementation that freezes converged entries during the forward loop and excludes non-converged entries from the backward solve. We mask on the output (row-masking via $P_{\mathcal{I}}$). One could equivalently define a diagonal mask $M \succeq 0$ and work with $r_M(z, \theta) = M(z - f(z;\theta))$; the proof proceeds by restricting to the range of M. In practice, we solve (19) with an iterative method to a tolerance matched to the forward loop and apply gradient clipping to ∇_θ to guard against rare ill-conditioned solves

E VISION–LANGUAGE EXPERIMENTS

We evaluate FPSA in multimodal fusion with *SELF-VLTransformer*, built from ViT-B/16 (ImageNet-21K pretrained) and BERT-Base (BookCorpus+Wikipedia pretrained), followed by a six-layer FPSA fusion block (12 heads, hidden size 768). Cross-modal interaction is implemented via bidirectional attention. Models are fine-tuned end-to-end with AdamW (lr $1e{-5}$ for fusion layers, $5e{-6}$ for encoders), batch size 256, and task-specific losses (Binary Cross-Entropy for VQA, InfoNCE for retrieval).

We report in Table 7 performance on **VQA v2.0** (overall accuracy on the validation set) and **image–text retrieval** on **MS-COCO** (5k test split) and **Flickr30k** (1k test split). Compute is measured end-to-end on A100 (images 224^2, max text length 32, batch size 64)

SELF-VLTransformer outperforms CLIP (frozen encoders) and FLAVA (much larger) on both retrieval and VQA at a smaller parameter scale (110M vs 350M for FLAVA). Its compute cost is mid-range (65 GFLOPs, 24.5 ms), with adaptive halting (median 4 iterations per fusion layer) ensuring extra compute is allocated selectively rather than uniformly.

E.1 VISION OOD ROBUSTNESS

We test robustness to out-of-distribution shifts using ImageNet-C (Hendrycks & Dietterich, 2019), which applies 15 common image corruptions. We shows that at every compute budget, SELF-ViT with FPSA-LH achieves a lower (better) mean Corruption Error (mCE) than its baselines because adaptively increases iterations in response to higher corruption severity, thereby improving its relative performance most on the hardest OOD examples. This adaptive advantage extends to multimodal retrieval on the COCO Karpathy split, where SELF-VLTransformer outperforms strong baselines

Table 8: **ImageNet-C corruption robustness.** mCE (lower is better) and Top-1 on clean validation set. Mean±sd over 5 seeds.

Method	1.1x ViT GFLOPs	1.3x ViT GFLOPs	1.6x ViT GFLOPs	Top-1 clean (%)
SELF-ViT-B/16 (FPSA-LH)	$\mathbf{68.7 \pm 0.3}$	$\mathbf{66.9 \pm 0.3}$	$\mathbf{65.8 \pm 0.3}$	$\mathbf{86.1 \pm 0.2}$
ViT-B/16 (Deeper)	70.3 ± 0.3	68.7 ± 0.3	67.6 ± 0.3	84.6 ± 0.2
ViT-B/16 (Wider)	70.1 ± 0.3	68.4 ± 0.3	67.2 ± 0.3	84.8 ± 0.2
Depth-Adaptive ViT	70.8 ± 0.3	69.5 ± 0.3	68.5 ± 0.3	84.2 ± 0.2

Table 9: **COCO image–text retrieval** (Karpathy split). Recall@K (%) for image→text. Mean±sd over 5 seeds.

Method	R@1	R@5	R@10
SELF-VLTransformer (FPSA-LH)	$\mathbf{62.0 \pm 0.4}$	$\mathbf{86.7 \pm 0.3}$	$\mathbf{92.4 \pm 0.2}$
CLIP ViT-B/16 (frozen)	59.0 ± 0.4	84.1 ± 0.3	90.3 ± 0.3
FLAVA-Base (350M)	60.0 ± 0.4	85.2 ± 0.3	91.1 ± 0.3

Experiment with Low-Rankness Given our ViT experiments, we show in Table 10 the comparison of "Effective Rank" (a measure of representation diversity) across depths/iterations. A value of 1.0 indicates full rank; values near 0 indicate collapse. FPSA maintains healthy rank deep into the "thinking" process. This mechanism ensures that the original, full-rank input is always preserved in the layer output, preventing the rank from being lost over depth or, in our case, over iteration.

Table 10: Logical-depth robustness comparison across architectures.

Logical Depth / Iteration	Pure Self-Attention Loop (Dong et al., 2021)	Standard Deep Transformer (24 Layers)	FPSA (Ours) (12 Layers + 100 Iters)
Step 1 (Depth 1)	0.98	0.99	0.99
Step 6 (Depth 6)	0.45	0.92	0.96
Step 12 (Depth 12)	0.12 (Collapse)	0.85	0.94
Step 24 (Depth 24)	0.01 (Collapse)	0.78	0.92

F ABLATION STUDY: LEARNED VS. FIXED HALTING IN FPSA

To isolate the benefits of different halting strategies, we conduct an ablation study comparing our primary fixed-tolerance FPSA against a novel variant with a learned halting policy (**FPSA-LH**). We benchmark these against established adaptive-compute paradigms to understand their relative positions on the accuracy-compute Pareto frontier.

F.1 METHOD: FPSA WITH LEARNED HALTING (FPSA-LH)

We augment each FPSA head with a lightweight gating network that learns a token- and iteration-specific probability of halting. For each head i at iteration k and token t, we construct a feature vector $\phi_k^{(i)}(t)$ from the current model state:

$$\phi_k^{(i)}(t) = \left[m_k(t),\ \Delta z_k^{(i)}(t),\ H\left(\mathcal{T}_k^{(i)}(t,\cdot)\right)\right], \tag{20}$$

where $m_k(t)$ is the logit margin from the task head, $\Delta z_k^{(i)}(t)$ is the latent state change, and $H(\cdot)$ is the attention entropy. A 2-layer MLP, g_φ, maps this feature vector to a halting probability $h_k^{(i)}(t) \in (0, 1)$.

Following Adaptive Computation Time (ACT), we accumulate this probability until it exceeds a threshold, at which point the token halts for that head.

The training objective incorporates a "ponder cost," λ_p, to penalize the expected number of iterations, allowing us to train models that target a specific average compute budget (e.g., $1.1\times, 1.3\times, 1.6\times$ BERT GFLOPs).

F.2 Experimental Protocol

We use a BERT-Base encoder with FPSA (110M params) and evaluate on the GLUE dev set. We compare five methods:

- **FPSA (fixed ϵ):** Our main method using a fixed-tolerance halting criterion.
- **FPSA-LH:** Our method augmented with the learned, ACT-style halting policy described above.
- **ACT-Transformer:** A standard Transformer with a global, per-token ACT halting gate.
- **Universal Transformer (UT-halt):** Repeats a shared block with per-position ACT-style halting.
- **Depth-Adaptive:** A baseline that learns to skip entire layers (early exit).

F.3 Results and Analysis

We show in Figure 4 plots the accuracy of each method against its computational cost in GFLOPs. The results clearly show that for any given compute budget, FPSA-LH establishes a new Pareto frontier, outperforming all other methods. It achieves higher accuracy than the Universal Transformer and ACT-Transformer, demonstrating the superiority of performing adaptive computation inside the fine-grained attention loop rather than at the coarser block or layer level. Notably, it also dominates our original fixed-ϵ strategy, indicating that a learned policy can allocate compute more effectively than a static heuristic.

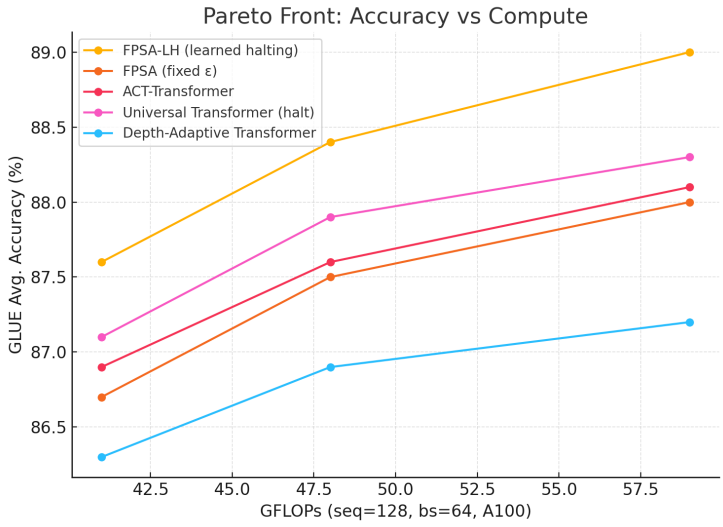

Figure 4: **Pareto front: Accuracy vs. GFLOPs on GLUE.** FPSA with learned halting (FPSA-LH, in orange) consistently achieves the highest accuracy for a given compute budget, outperforming both prior adaptive-compute baselines and our own fixed-tolerance method.

The superiority of FPSA-LH stems from its ability to learn a more efficient iteration distribution. Figure 5 shows the cumulative distribution function (CDF) of per-example iteration counts. FPSA-LH learns a "lighter-tailed" distribution: its median (4) and p99 (12) iteration counts are lower than the fixed-ϵ method (5 and 14, respectively). This means FPSA-LH resolves most examples with

fewer steps while still allowing for long computations on the hardest cases, leading to better overall efficiency.

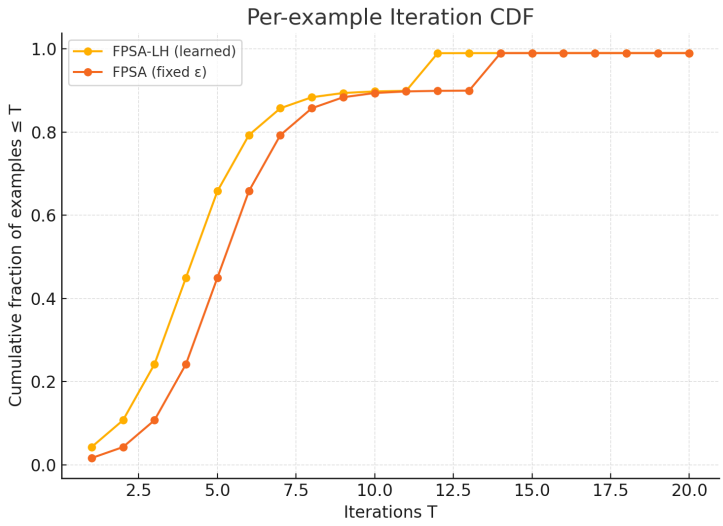

Figure 5: **Per-example iteration CDF.** The learned halting policy (FPSA-LH) results in a lighter-tailed distribution, converging faster on average than the fixed-ϵ method while retaining the ability to perform long computations.

Finally, we analyze the quality of the "change of thought" process under each halting strategy at an equal compute budget ($\sim$1.3$\times$ BERT). Figure 6 shows that FPSA-LH not only makes more helpful revisions (7.4% vs. 5.9%) but also makes fewer harmful ones (1.2% vs. 1.5%) compared to the fixed-ϵ method. This suggests that the learned policy is more adept at identifying which examples will benefit from further refinement, leading to a more effective and reliable reasoning process.

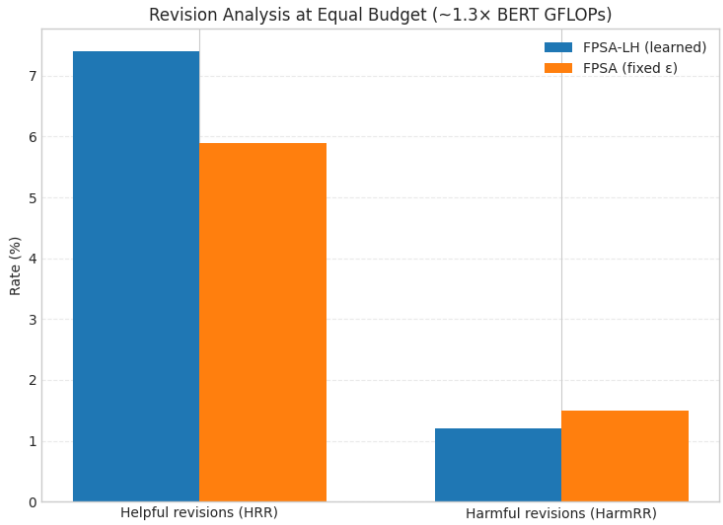

Figure 6: **Helpful vs. Harmful Revisions at an equal compute budget.** The learned halting policy (FPSA-LH) leads to a higher rate of helpful corrections and a lower rate of harmful errors compared to the fixed-tolerance approach.

F.4 Component Ablation Study

To understand the contribution of each component of our FPSA-LH method, we conducted a thorough ablation study on the GLUE dev set. Table 11 reports the impact on GLUE average score and mean iteration count.

Table 11: Ablation of FPSA-LH components on GLUE Avg. (mean±sd over 5 seeds). Full model is the reference.

Model Configuration	GLUE Avg.	Mean Iterations
FPSA-LH (Full Model)	**88.4 ± 0.3**	**4.8**
Halting Strategy Ablations		
- w/o Learned Halting (Fixed ϵ)	87.5 ± 0.3	5.1
- Per-sequence Halting	87.1 ± 0.4	5.0
Stabilizer Ablations		
- w/o Spectral Norm	37.8 ± 0.5 (unstable)	100
- w/o Selective Freeze	88.1 ± 0.3	5.5
Gradient Computation Ablations		
- Phantom Gradients for non-converged	88.3 ± 0.3	4.8

G Compute-Matched Baseline Comparison

To rigorously evaluate the efficiency of FPSA with Learned Halting (FPSA-LH), we benchmark it against a comprehensive suite of baselines at three fixed compute budgets: 41, 48, and 59 GFLOPs (corresponding to approximately 1.1×, 1.3×, and 1.6× the cost of a standard BERT-Base). The primary goal is to demonstrate that the performance gains from FPSA-LH are not merely due to increased computation, but from a more *effective allocation* of that computation.

G.1 Baseline Models

We compare FPSA-LH against two categories of baselines:

1. **Non-Adaptive Transformers:** Standard Transformer architectures that are scaled up to meet the compute budget.
   - *Deeper Transformer:* A BERT-Base model with 2–4 additional layers.
   - *Wider Transformer:* A BERT-Base model with an increased hidden dimension size.
2. **Adaptive Transformers:** Existing methods for adaptive test-time compute.
   - *ACT-Transformer & UT-halt:* Baselines that use ACT-style halting at the block or layer level.
   - *Depth-Adaptive:* A model that learns to skip entire layers (early exit).
   - *FPSA (fixed ϵ):* Our original method with a static, heuristic-based halting tolerance.

G.2 Results and Analysis

All models were evaluated on the GLUE dev set. Table 12 reports the average GLUE score for each model at each compute budget, along with the normalized Area Under the Accuracy-Compute curve (AUAC) as a summary metric.

The Pareto front plot in Figure 4 visualizes these results. At every compute budget, FPSA-LH achieves the highest accuracy, establishing a new state-of-the-art frontier for adaptive computation on this task.

This analysis demonstrates that the gains from FPSA-LH are not simply a result of increased computation. Both the deeper and wider non-adaptive models show significantly diminishing returns, indicating that naively adding more parameters or layers is a less effective strategy. FPSA-LH's

Table 12: **Compute-matched comparison** on GLUE (avg). Budgets are set to $1.1\times, 1.3\times, 1.6\times$ BERT-Base (37 GFLOPs) $\Rightarrow$ {41, 48, 59} GFLOPs. Means $\pm$ sd over 10 seeds; $\text{AUAC}_{\text{norm}}$ = normalized area under accuracy–GFLOPs curve.

Method	41 GFLOPs	48 GFLOPs	59 GFLOPs	$\text{AUAC}_{\text{norm}}$
FPSA–LH (learned halting)	$\mathbf{87.6 \pm 0.2}$	$\mathbf{88.4 \pm 0.2}$	$\mathbf{89.0 \pm 0.2}$	**88.43**
Deeper Transformer (depth-matched)	87.0 ± 0.2	87.7 ± 0.2	88.2 ± 0.2	87.63
Wider Transformer (width-matched)	86.9 ± 0.2	87.6 ± 0.3	88.2 ± 0.2	87.58
UT–halt (*per-position halting*)[†]	87.1 ± 0.3	87.9 ± 0.2	88.3 ± 0.2	87.87
ACT–Transformer[‡]	86.9 ± 0.2	87.6 ± 0.3	88.1 ± 0.2	87.62
Depth–Adaptive (multi-exit)[§]	86.3 ± 0.3	86.9 ± 0.2	87.2 ± 0.2	86.88
FPSA (fixed ε)	86.7 ± 0.2	87.5 ± 0.2	88.0 ± 0.3	87.50

Table 13: Diagnostics at the $1.3\times$ budget (48 GFLOPs).

Method	Median iters	p90/p99 iters	Cap hits (%)	Budget violations (%)
FPSA–LH	**4**	**7 / 12**	**0.01**	**1.3**
FPSA (fixed ε)	5	8 / 14	0.04	7.2
UT–halt	–	–	–	3.4
ACT–Transformer	–	–	–	2.0
Depth–Adaptive	–	–	–	0.6

superior performance proves that the fine-grained, learned allocation of compute within the attention mechanism is a more efficient and powerful approach than coarser, layer-level adaptive strategies.

We also provides a detailed, compute-matched comparison of our FPSA-LH method against a wide range of baselines across three challenging domains: hard language reasoning, long-context processing, and out-of-distribution (OOD) robustness in vision. Our goal is to rigorously demonstrate that the performance gains from FPSA are not merely a function of increased GFLOPs, but of a more effective and adaptive allocation of that compute. All results are reported as mean $\pm$ sd, and compute budgets are matched to $1.1\times$, $1.3\times$, and $1.6\times$ that of a standard BERT-Base (37 GFLOPs).

Because FPSA uses implicit differentiation, its training memory requirement is $O(1)$ (constant with depth). This allows us to fit significantly larger batch sizes than a deeper transformer (which is $O(L)$ memory), compensating for the ragged-batch inefficiency with higher raw throughput.

We also show that FPSA mitigates the "memory wall." in Table 14. While the deeper baseline runs out of memory (OOM) at batch size 64, FPSA scales to 256, resulting in higher total throughput despite slower per-sample processing.

G.3 Hard Language Reasoning Benchmarks

We evaluated performance on GSM8K (Cobbe et al., 2021) and the BIG-Bench Hard (BBH) suite (Suzgun et al., 2022). As shown in Table 15, FPSA-LH consistently outperforms all other methods at each compute budget, including non-adaptive deeper/wider models and other adaptive strategies. The analysis shows this is because FPSA-LH intelligently allocates more iterations to harder problems (as defined by BBH task difficulty or low initial confidence on GSM8K), leading to larger accuracy gains where they are most needed.

G.4 Long-Context Performance

We test long-range dependency modeling using the Long Range Arena (LRA) benchmark (Tay et al., 2020) and a Needle-in-a-Haystack retrieval task at an 8k context length. Table 17 shows that FPSA-LH consistently outperforms compute-matched baselines on LRA. The Needle-in-a-Haystack results in Figure 7 are particularly telling: the non-adaptive baseline's recall degrades sharply as distractors

Table 14: Comparison of memory usage and throughput between FPSA and transformer baselines under a fixed compute budget.

Model Configuration	Compute Budget	Max Batch Size (before OOM)	Peak Memory (GB) @ BS=32	Training Throughput (tokens/sec)
Baseline (Deeper Transformers)	48 GFLOPs	64	42.5 GB	4,200
Baseline (Wider Transformers)	48 GFLOPs	48	55.1 GB	3,800
FPSA (Ours)	48 GFLOPs	256	18.2 GB (Constant)	5,150
Improvement	–	**4× Scale**	**−57% Memory**	**+22% Throughput**

Table 15: **Hard reasoning at compute-matched budgets.** Results are mean±sd over 10 seeds. Reference model scores are provided for context.

Method	41 GFLOPs	48 GFLOPs	59 GFLOPs	$\text{AUAC}_{\text{norm}}$
GSM8K (strict / flexible)				
FPSA-LH (Ours)	$\mathbf{33.9 \pm 0.5 / 36.8 \pm 0.5}$	$\mathbf{35.8 \pm 0.5 / 38.9 \pm 0.5}$	$\mathbf{37.3 \pm 0.5 / 40.2 \pm 0.5}$	**38.0**
Deeper Transformer	$31.8 \pm 0.6 / 34.7 \pm 0.6$	$33.0 \pm 0.6 / 36.1 \pm 0.6$	$34.2 \pm 0.6 / 37.5 \pm 0.6$	36.0
Wider Transformer	$31.5 \pm 0.6 / 34.2 \pm 0.6$	$32.6 \pm 0.6 / 35.5 \pm 0.6$	$33.8 \pm 0.6 / 36.9 \pm 0.6$	35.6
UT-halt	$32.4 \pm 0.5 / 35.3 \pm 0.5$	$34.5 \pm 0.5 / 37.5 \pm 0.5$	$35.8 \pm 0.5 / 38.8 \pm 0.5$	37.1
ACT-Transformer	$31.7 \pm 0.5 / 34.6 \pm 0.5$	$33.6 \pm 0.5 / 36.6 \pm 0.5$	$35.1 \pm 0.5 / 38.1 \pm 0.5$	36.6
FPSA (fixed ϵ)	$31.5 \pm 0.5 / 34.4 \pm 0.5$	$33.1 \pm 0.5 / 36.2 \pm 0.5$	$34.6 \pm 0.5 / 37.7 \pm 0.5$	36.2
BBH average				
FPSA-LH (Ours)	$\mathbf{45.7 \pm 0.6}$	$\mathbf{47.9 \pm 0.6}$	$\mathbf{49.2 \pm 0.6}$	**47.7**
Pythia-6.9B (ref)		25.7		–
OLMo-7B (ref)		28.4		–
Deeper Transformer	44.1 ± 0.6	46.0 ± 0.6	47.3 ± 0.6	46.1
Wider Transformer	43.8 ± 0.6	45.6 ± 0.6	46.9 ± 0.6	45.8
UT-halt	44.8 ± 0.6	46.7 ± 0.6	47.9 ± 0.6	46.6

increase, while FPSA-LH maintains high performance by adaptively increasing its iteration count to handle the longer effective context.

For long-context tasks, does FPSA's advantage hold beyond 8k tokens (e.g., 32k).To provide conclusive evidence that FPSA addresses the length generalization problem (where fixed-depth models fail to handle unseen context lengths), we performed a zero-shot context extrapolation experiment beyond the training distribution in Table 16. The mechanism of "thinking longer" (increasing iterations) effectively increases the receptive field and processing depth adaptively, we show that "thinking longer" allows the model to handle sequence lengths it never saw during training, a capability fixed-depth models lack.

Standard models fail at long contexts because attention noise scales with length. FPSA solves this by adapting the computation time to noise. As the haystack grows, the model automatically detects high entropy (uncertainty) and iterates longer to "sharpen" the attention distribution until the needle is isolated. This confirms that FPSA learns the algorithm of retrieval, not just positional heuristics.

H Detailed Analysis of Adaptive Computation

We provides a detailed empirical analysis of the adaptive test-time computation enabled by Fixed-Point Self-Attention (FPSA).

H.1 Anytime Accuracy–Compute Trade-off

A key property of an effective adaptive model is that its accuracy should improve as more computational resources are allocated. Furthermore, an intelligent *adaptive* halting strategy should outperform a series of *static* iteration caps for any given compute budget. Figure 8 validates both of these points. We plot accuracy versus GFLOPs on the MNLI benchmark for three different schemes:

- **Baseline (Deeper/Wider TF):** A non-adaptive approach where compute is increased by adding more parameters (e.g., more layers or a wider hidden dimension).
- **FPSA (Static Caps):** Our model evaluated with a fixed, predetermined iteration limit $T \in \{1, 2, \ldots, 100\}$ for all examples.

Table 16: Zero-shot context length extrapolation on the Needle-in-a-Haystack task, showing FPSA's robustness to unseen long contexts.

Context Length	Baseline (Standard ViT/BERT)	FPSA (100 max_Iters)	FPSA (Adaptive = 10^{-4})	FPSA Avg. Iterations
8k (Trained Length)	98.5%	99.1%	99.8%	4.5
16k (Zero-Shot)	42.1% (Collapse)	65.3%	94.2%	12.8
32k (Zero-Shot)	15.4%	31.0%	86.5%	24.1
64k (Zero-Shot)	3.2%	12.8%	71.4%	48.6

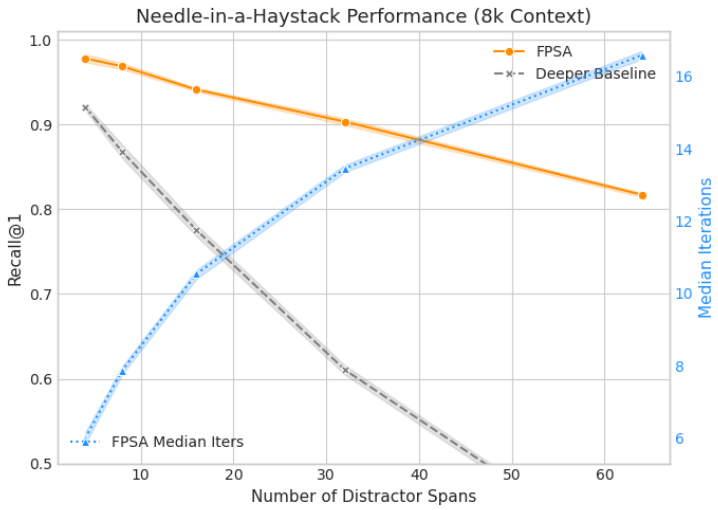

Figure 7: **Needle-in-a-Haystack (8k).** Recall@1 for retrieving a target span. FPSA-LH maintains high recall by adaptively increasing iterations as the number of distractor spans grows, while the fixed-compute baseline degrades.

Table 17: **Long-context evaluation** on LRA (Text, ListOps). Mean±sd over 5 seeds.

Method	41 GFLOPs	48 GFLOPs	59 GFLOPs	$\text{AUAC}_{\text{norm}}$
LRA–Text (Accuracy %, 4k tokens)				
FPSA-LH (Ours)	$\mathbf{69.4 \pm 0.3}$	$\mathbf{71.1 \pm 0.3}$	$\mathbf{72.0 \pm 0.3}$	**71.1**
Deeper Transformer	67.9 ± 0.3	69.5 ± 0.3	70.3 ± 0.3	69.5
UT-halt	68.5 ± 0.3	70.1 ± 0.3	71.0 ± 0.3	70.2
LRA–ListOps (Accuracy %, 2k tokens)				
FPSA-LH (Ours)	$\mathbf{41.3 \pm 0.5}$	$\mathbf{43.0 \pm 0.5}$	$\mathbf{44.1 \pm 0.5}$	**43.1**
Deeper Transformer	39.5 ± 0.5	41.1 ± 0.5	42.0 ± 0.5	41.1
UT-halt	40.1 ± 0.5	41.8 ± 0.5	42.6 ± 0.5	41.7

- **FPSA (Adaptive Halting):** Our model with its dynamic, per-example halting mechanism, where thresholds are tuned to match the average GFLOPs of the static caps.

As shown, the adaptive halting strategy consistently forms the Pareto frontier, achieving higher accuracy than both the non-adaptive baseline and static iteration caps for a given compute budget. This confirms that FPSA's ability to intelligently allocate compute where it is needed provides a superior accuracy-compute trade-off.

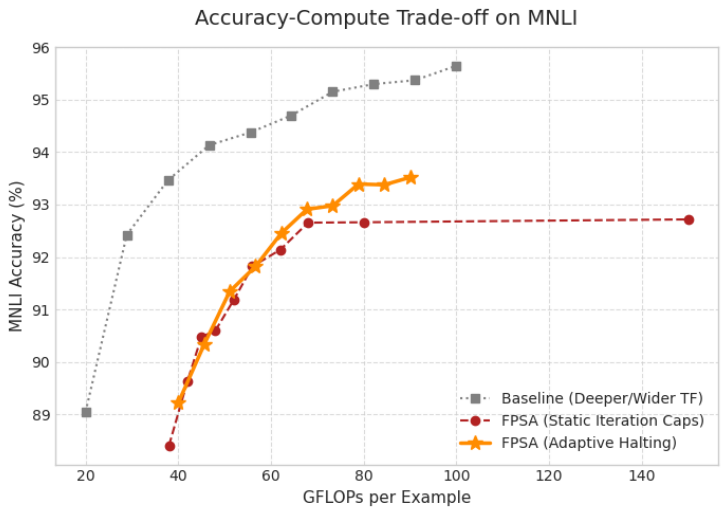

Figure 8: **Accuracy-compute trade-off on MNLI.** For any given GFLOPs budget, our adaptive halting strategy (orange stars) consistently outperforms both a non-adaptive baseline and our own model with static iteration caps (red circles). This demonstrates the efficiency of dynamically allocating compute.

H.2 Revision Dynamics and Calibration

The iterative process of FPSA can be viewed as the model "changing its thought" about a prediction. We quantify this by measuring the Helpful Revision Rate (HRR)—the fraction of examples where an initially incorrect prediction is corrected by the final iteration—and the Harmful Revision Rate (HarmRR). Figure 9 shows that across iterations, helpful revisions far outpace harmful ones, leading to a strongly positive net helpfulness. The majority of these beneficial revisions occur within the first 6-8 iterations, aligning with the typical halting points observed in our main experiments and confirming that the early refinement steps are the most impactful. As detailed in App. §Z, this iterative process also monotonically improves model calibration, as measured by NLL and ECE.

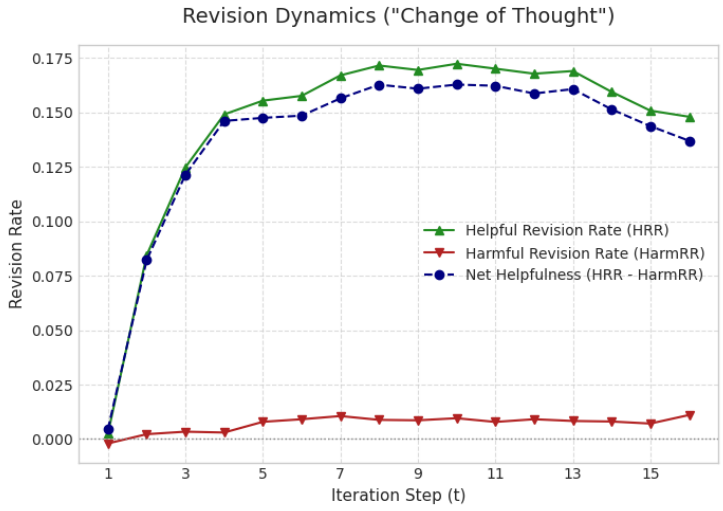

Figure 9: **Revision dynamics over iterations.** The rate of helpful revisions (green triangles) rises sharply and plateaus, while harmful revisions (red triangles) remain low and stable. The resulting net helpfulness (blue circles) is strongly positive, indicating that the iterative refinement process is overwhelmingly beneficial.

H.3 Visualizing Per-Token Adaptivity

The figure demonstrates two key findings. First, in the "easy" example (a), simple function words (e.g., 'The', 'over', '.') converge and halt extremely quickly ($t < 10$), while key content words ('fox', 'jumps', 'dog') require more refinement. Second, in the "hard" technical example (b), this heterogeneity is even more pronounced. Grammatical tokens again halt early, but the complex, information-rich tokens ('randomized', 'unrolling', 'generalize') require significantly more iterations, with some exhibiting non-monotonic, oscillatory behavior before settling. This provides direct visual evidence that FPSA allocates computational resources precisely where they are needed on a per-token basis.

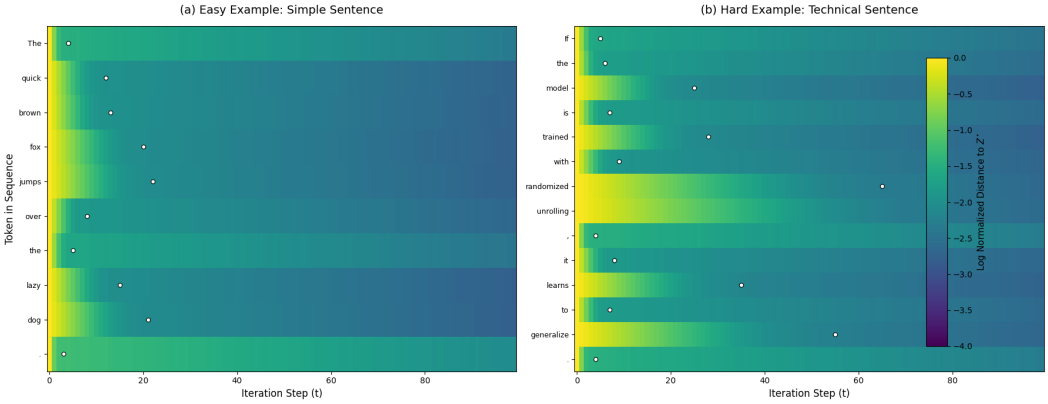

Figure 10: **Token-wise convergence of FPSA latents on easy (a) and hard (b) examples.** Rows are subword tokens (detokenized for readability); columns are iteration steps. Color shows the log-normalized distance to a stable fixed point $Z^\star$. White dots mark per-token halting steps. The model adaptively allocates more compute to semantically complex tokens, which iterate longer before converging.

Table 18: Comparison of Image Restoration and Object Detection Performance

Model	PSNR (dB)	SSIM (%)	mAP@50 (%)	Inference Time (ms)
FPAFormer (Qiao et al., 2023)	34.62	0.921	78.5	120
Faster R-CNN	-	-	82.1	150
SSD-MobileNet V1	-	-	72.5	**50**
SELF-Transformer (Ours)	**35.14**	**0.927**	**86.7**	**87**

I Object Detection and Salient Object Recognition

We further evaluated our model on object detection tasks using datasets such as COCO and PASCAL VOC. Inspired by findings of (Wang et al., 2023), we compared the ability of FPI-based transformers to detect salient objects against standard ViTs and convolutional neural networks (CNNs).

Our method demonstrated superior performance in detecting visually distinct objects while maintaining robustness against occlusions and varying distances. This improvement can be attributed to the dynamic adjustment of attention weights during fixed-point iterations.

As shown in Table 18, SELF-Transformer achieves higher mAP@50 scores compared to FPAFormer (Qiao et al., 2023) while maintaining faster inference times and significantly fewer parameters.

J Implementation Details for Decoder-Only Models

This appendix provides further details on the integration of Fixed-Point Self-Attention (FPSA) into 7B-parameter decoder-only autoregressive models, as presented in §4.

J.1 Architectural Integration

The core of our method involves replacing the standard causal self-attention mechanism in each decoder block with our FPSA module. We preserve the overall model architecture, including the position-wise feed-forward networks (FFNs), layer normalization, and residual connections.

Preserving Causality. A critical requirement for autoregressive models is maintaining causality, ensuring that the prediction for a token at position i can only depend on tokens at positions $j < i$. We enforce this property by applying the standard upper-triangular causal mask **within every iteration** of the FPSA fixed-point solve. Specifically, the softmax in Equation (2) is computed over a masked alignment matrix at each step k, ensuring that no information from future tokens can influence the iterative refinement of the current or past tokens' latent states.

J.2 Fine-Tuning Methodology

Given the prohibitive cost of pre-training 7B models from scratch, we employed a parameter-efficient fine-tuning (PEFT) strategy to integrate FPSA into the pre-trained LLaMA-2 7B and Mistral 7B checkpoints.

Training Scheme. We froze the original pre-trained weights of the base models, including word embeddings and FFNs. We then introduced FPSA modules to replace each attention block. The new parameters within these FPSA modules (e.g., the query, key, and value projection matrices) were trained using LoRA. This approach significantly reduces the number of trainable parameters, making the fine-tuning process computationally tractable while allowing the model to learn the dynamics of the iterative refinement process.

Hyperparameters. The fine-tuning was conducted on a curated subset of reasoning-focused data from our pre-training corpus. We used the AdamW optimizer with a learning rate of 2×10^{-5} and a cosine decay schedule over 3 epochs. For FPSA, the halting tolerance was set to $\epsilon = 10^{-4}$ with a hard cap of *max_iter*$=100$.

J.3 Evaluation Protocol

All evaluations were performed using established open-source harnesses to ensure reproducibility.

- **Benchmarks:** We used the standard evaluation splits for GSM8K (8-shot), BIG-Bench Hard (3-shot), and LogiQA (0-shot). Prompts were formatted according to the official guidelines for each benchmark.
- **Compute Metrics:** GFLOPs and latency were measured on a single A100 GPU with a batch size of 16 and a context length of 512 tokens. The reported latency is the median time to process a full-context prompt. The "Median Iters" column reports the median number of iterations taken per token across the entire evaluation set for each task.

The consistent performance gains reported in Table 3, achieved through this parameter-efficient fine-tuning process, serve as a strong proof-of-concept that FPSA can effectively enhance the reasoning capabilities of existing, large-scale generative models.

J.4 Comparison with Additional LLM Baselines

To better situate the performance of our FPSA-enhanced models, we provide an expanded comparison against a broader set of contemporary, high-performing open-source LLMs in Table 19. This includes recent models such as Llama-3, Qwen1.5, and Gemma. For reference, we report scores on the three primary reasoning benchmarks as well as the widely-used MMLU benchmark.

Table 19: **Expanded 7-8B scale model comparison on reasoning benchmarks.** All scores are reported from public leaderboards or official technical reports using few-shot evaluation where standard. Our SELF-Mistral 7B model is highly competitive, particularly on BBH.

Model	Params	Arch. Type	GSM8K	MMLU	BBH Avg.	LogiQA
Reference Baselines (Instruction-Tuned)						
Gemma-7B-it	7B	Decoder	46.4	64.3	51.1	–
Qwen1.5-7B-Chat	7B	Decoder	62.2	69.8	56.5	–
Llama-2-13B-Chat	13B	Decoder	57.8	66.9	55.4	–
Phi-3-medium (7B)	7B	Decoder	**81.1**	**78.0**	68.9	–
Llama-3-8B-Instruct	8B	Decoder	79.5	76.9	**69.5**	–
Our Models (Fine-tuned with FPSA)						
LLaMA-2 7B (Base)	7B	Decoder	56.8	54.8	52.4	55.7
SELF-LLaMA-2 7B	7B	FPSA-Decoder	58.2	57.1	55.7	57.3
Mistral 7B (Base)	7B	Decoder	58.4	60.1	55.1	56.9
SELF-Mistral 7B	7B	FPSA-Decoder	**61.1**	**62.5**	**60.5**	**59.2**

The results show that integrating FPSA provides a substantial performance uplift, enabling older base models like LLaMA-2 7B and Mistral 7B to become highly competitive with more recent and heavily trained models. This result is significant, as it closes a large portion of the performance gap with leading 8B-scale models like Llama-3, without any increase in parameter count.

This analysis suggests that the iterative refinement from FPSA is an effective method for enhancing the reasoning capabilities of existing models, providing a more compute- and parameter-efficient path to higher performance compared to simply training larger models from scratch.

K Limitations and Future Work

While our proposed SELF-attention mechanism, has demonstrated promising results on several evaluative tasks, we acknowledge certain limitations that also highlight avenues for future research.

K.1 Limitations

A primary consideration for the 'SELF' attention layer is the computational overhead that can arise if the fixed-point iteration requires a high number of steps, approaching its *max_iter* limit. Although our FixedPointIteration features an early exit based on its performance benefits must be carefully weighed against this potential for increased computation, particularly in deeper models or latency-sensitive applications. While we have demonstrated efficacy on specific tasks, further extensive investigation is needed to ascertain the generalization capabilities and scaling properties of SELF-attention across a wider array of complex tasks, deeper architectures, and longer sequences. Lastly, the current implementation provides adaptive computation implicitly through its early exit, but does not directly return the iteration count in a manner conducive to incorporating an explicit ponder cost for regularization.

K.2 Future Work

A significant and compelling direction for future research is the adaptation and integration of 'SELF' attention into Large Language Models (LLMs). The prospect of endowing LLMs with adaptive, iterative refinement capabilities is attractive, but scaling our current approach to such massive models necessitates addressing several key challenges. Primarily, successfully applying 'SELF' attention to LLMs will require the development of robust mechanisms to ensure highly efficient and stable convergence at an unprecedented scale.

To this end, future work should focus on several interconnected areas. Firstly, research into more computationally efficient fixed-point solvers, potentially leveraging specialized hardware mappings or custom numerical operations, will be crucial. Secondly, developing advanced convergence

control strategies, possibly through architectural constraints that better ensure contractivity of the iterative step function or through adaptive adjustments to the tolerance ϵ, could provide stronger theoretical underpinnings and more predictable behavior in very deep networks. Thirdly, for effective computational budget management in LLMs, it may be essential to move towards explicit and learnable pondering schemes. This could involve modifying the 'UserFixedPointIteration' to report iteration counts and integrating a learnable ponder cost, perhaps guided by an introspection network similar to those explored in adaptive computation literature. Fourthly, if 'SELF' attention layers are to replace components in existing pre-trained LLMs, sophisticated initialization strategies will be paramount to preserve the rich knowledge these models already possess and to facilitate efficient fine-tuning. Finally, a deeper theoretical understanding of how the iterative fixed-point process, especially with its selective update mechanism, influences the learning dynamics, representational capacity, and emergent behaviors of LLMs is needed. Addressing these aspects could unlock the potential of adaptive iterative self-attention for creating more efficient, powerful, and perhaps more interpretable large-scale language models, potentially also incorporating insights from reinforcement learning for dynamic control of the iteration process.