Epsilon: A Transformer with Adaptive Computation and Quantized Attention

Lumina Mescuwa[1]

[1]Independent Researcher

Abstract

The quadratic complexity of the standard Transformer architecture in both computation and memory presents a significant bottleneck for processing long sequences. To address this, we introduce EPSILON, a novel Transformer architecture designed for high efficiency, training stability, and interpretability. EPSILON synthesizes three core principles: (1) **adaptive computation** through... (2) **efficient attention** via Histogram Quantized Scalar Attention (HQSA), a novel method that reduces attention complexity from $O(n^2)$ to sub-quadratic $O(nB)$ by attending to a small, fixed number of aggregated feature bins; and (3) **enhanced training stability** through architectural safeguards including spectral normalization, scaled residuals, and specialized layer normalization. We demonstrate empirically on the IMDb sentiment classification task that EPSILON not only achieves a higher validation accuracy (90.1%) than a comparable vanilla Transformer baseline (85.1%), but does so while being approximately 1.5x faster per epoch and using fewer parameters. This performance is achieved by learning to solve the task with an average computation depth of less than 2 layers, showcasing the profound efficiency of its adaptive design.

1 Introduction

The Transformer architecture [1] has become the de facto standard for a wide range of tasks in natural language processing. Its success is largely attributed to the self-attention mechanism, which allows the model to capture long-range dependencies by directly relating every token in a sequence to every other token. However, this full attention comes at a steep cost: the computational and memory requirements of the self-attention mechanism scale quadratically with the sequence length, n. This $O(n^2)$ complexity makes it prohibitively expensive to apply standard Transformers to high-resolution data or very long documents.

To overcome this fundamental limitation, we introduce EPSILON, a novel Transformer architecture designed to deliver high performance with sub-quadratic efficiency. EPSILON is built on a synthesis of three core principles:

1. **Adaptive Computation:** Instead of a fixed stack of distinct layers, EPSILON employs a single, recurrent *EpsilonBlock* with shared weights, inspired by the Universal Transformer [2]. This is coupled with a per-token adaptive halting mechanism, allowing each token to dynamically exit the computation once its representation has stabilized. Easy-to-process tokens halt early, drastically reducing the average computational depth.

2. **Efficient Attention:** We propose Histogram Quantized Scalar Attention (HQSA), an attention mechanism where queries attend to a small, fixed number of B aggregated feature "bins" instead of n individual tokens. This reduces the complexity of the attention operation from $O(n^2)$ to a more scalable $O(nB)$.

3. **Training Stability:** Recurrent architectures can be prone to instability. Epsilon incorporates several architectural safeguards to ensure stable convergence, including spectral normalization on its feed-forward layers [5], carefully scaled residual connections, and a specialized 'CenterNorm' layer.

Our contributions in this paper are as follows:

- We introduce the Epsilon architecture, a cohesive model integrating a recurrent block, adaptive halting, and a novel quantized attention mechanism (HQSA). We formalize each component mathematically, providing the theoretical underpinnings for its stability and efficiency.
- We demonstrate empirically on the IMDb sentiment classification task that Epsilon significantly outperforms a comparable vanilla Transformer baseline in both accuracy (by over 5 percentage points) and training speed (approx. 1.5x faster), all while using fewer parameters.
- We analyze the adaptive behavior of Epsilon, showing that it learns to solve the task using an average of only 1.78 out of a possible 8 layers, confirming the effectiveness of its computational savings.

2 Related Work

Epsilon builds upon several threads of research aimed at improving the efficiency and stability of Transformer models.

Efficient Transformers. A large body of work has sought to mitigate the $O(n^2)$ complexity of self-attention. These methods can be broadly categorized into sparse attention (e.g., Longformer [9]), kernelized attention (e.g., Linformer [10]), and clustering-based approaches. Epsilon's HQSA falls into the latter category. The Routing Transformer [7] uses online k-means clustering to group queries and keys, restricting attention to within-cluster interactions. The Set Transformer [8] introduced the concept of learned inducing points to create an attention bottleneck. HQSA is similar in spirit but uses a differentiable soft-assignment of keys to learned prototypes at each layer, creating dynamic, content-aware bins.

Adaptive Computation. The idea of dynamically adjusting computation per input is not new. The Universal Transformer [2] first proposed a recurrent, weight-sharing Transformer layer paired with a dynamic halting mechanism. This was built on the earlier work on Adaptive Computation Time (ACT) by Graves [3], which provided a differentiable method for a recurrent network to learn how many steps to perform. More recently, PonderNet [4] framed this as learning a halting distribution, regularizing it against a geometric prior to prevent degenerate solutions. Epsilon's halting mechanism is a direct descendant of this lineage, using a KL-divergence penalty against a geometric prior to achieve stable, adaptive depth.

Model Stability in Deep Networks. Training very deep or recurrent neural networks is challenging due to the risks of exploding or vanishing gradients. Spectral normalization [5] was introduced as a powerful technique to stabilize training by constraining the Lipschitz constant of network layers. It has been shown to be highly effective in GANs and other deep architectures. Similarly, carefully designed residual connections [6] are critical for enabling gradient flow. Epsilon incorporates both spectral normalization in its FFN layers and a scaled residual factor, α_{res}, to ensure its recurrent dynamics remain well-behaved.

3 The Epsilon Architecture

3.1 High-Level Overview

At its core, Epsilon processes a sequence by iteratively applying a single, shared ‘EpsilonBlock‘. An input sequence of embeddings $h^{(0)} \in \mathbb{R}^{n \times d}$ is refined through a series of updates:

$$h^{(t+1)} = F(h^{(t)})$$

where F represents the ‘EpsilonBlock‘ transformation. Unlike a standard Transformer, which uses a fixed number of distinct layers, Epsilon re-uses F for a variable number of steps T_i for each token i, determined by an adaptive halting mechanism.

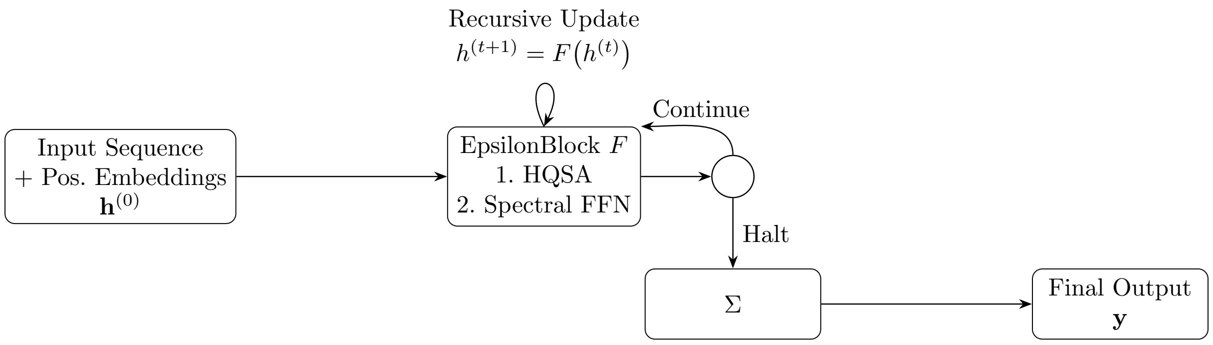

Figure 1: High-level architecture of the Epsilon Transformer.

3.2 The Recursive EpsilonBlock

The ‘EpsilonBlock‘ is the central computational unit of the model. It consists of two main sub-layers: a Histogram Quantized Scalar Attention (HQSA) layer and a Feed-Forward Network (FFN), each wrapped in a residual connection with pre-normalization.

For an input state $h^{(t)}$, the block computes:

1. **Attention Sub-layer:**

$$\begin{aligned}\tilde{h} &= \text{CenterNorm}(h^{(t)}) \\ A &= \text{HQSA}(\tilde{h}, \tilde{h}, \tilde{h}) \\ h' &= h^{(t)} + \alpha_{\text{res}} \cdot \text{Dropout}(A)\end{aligned}$$

2. **Feed-Forward Sub-layer:**

$$\begin{aligned}\tilde{h}' &= \text{CenterNorm}(h') \\ U &= \text{FFN}(\tilde{h}') \\ h^{(t+1)} &= h' + \alpha_{\text{res}} \cdot \text{Dropout}(U)\end{aligned}$$

Stability is promoted by three key components:

- **CenterNorm:** A normalization layer that centers features to a mean of zero, defined as $y = \sqrt{\frac{d}{d-1}}(x - \mathbb{E}[x])$. This prevents mean-shift issues without altering variance structure.
- **Spectral Normalization:** The linear layers within the FFN are spectrally normalized, constraining their Lipschitz constants to be ≤ 1.

- **Residual Scaling (α_{res}):** A factor $\alpha_{\text{res}} < 1$ (e.g., 0.25) dampens the magnitude of each residual update, preventing oscillatory or divergent behavior in the recurrent updates.

These components work in concert to ensure that the function F is a contraction mapping under the ℓ_2 norm, guaranteeing convergence to a stable fixed-point representation (see Appendix A.1 for a detailed analysis).

3.3 Histogram Quantized Scalar Attention (HQSA)

HQSA reduces attention complexity from $O(n^2)$ to $O(nB)$ by clustering keys into B bins and having queries attend to these aggregated bin representations. For a single head, given queries Q, keys K, and values $V \in \mathbb{R}^{n \times d_h}$:

1. Soft Key-to-Bin Assignment. Each head learns a set of B prototype vectors $\{U_b\}_{b=1}^{B} \subset \mathbb{R}^{d_{\text{lat}}}$. Keys are projected into this latent space via $z_i = W_{\text{bin}} K_i$. Soft assignment probabilities are computed via a softmax over scaled dot-products with the prototypes:

$$\ell_{i,b} = \frac{z_i \cdot U_b}{\tau} \quad \text{and} \quad a_{i,b} = \frac{\exp(\ell_{i,b})}{\sum_{b'=1}^{B} \exp(\ell_{i,b'})}$$

where τ is a learned temperature parameter.

2. Bin Aggregation. Keys and values are aggregated into bin representations via a weighted average, normalized by the total assignment mass $m_b = \sum_i a_{i,b}$:

$$K_{\text{bin},b} = \frac{\sum_{i=1}^{n} a_{i,b} K_i}{m_b + \epsilon} \quad \text{and} \quad V_{\text{bin},b} = \frac{\sum_{i=1}^{n} a_{i,b} V_i}{m_b + \epsilon}$$

3. Query-to-Bin Attention. Each query Q_j then attends to the B binned keys:

$$s_{j,b} = \frac{Q_j \cdot K_{\text{bin},b}}{\sqrt{d_h}} \quad \text{and} \quad w_{j,b} = \text{softmax}_b(s_{j,b})$$

The final context vector is a weighted sum of the binned values: $\text{out}_j = \sum_{b=1}^{B} w_{j,b} V_{\text{bin},b}$.

3.4 Adaptive Computation Halting

Epsilon learns a per-token policy for how many iterations of the 'EpsilonBlock' to apply. At each iteration t, a halting score $s_i^{(t)}$ is computed from the token's state $h_i^{(t)}$, which is converted to a probability $\sigma(s_i^{(t)})$.

We maintain a 'remainder' probability $r_i^{(t)}$ for each token, initialized to $r_i^{(1)} = 1$. The probability of halting at the current step, $p_i^{(t)}$, is capped by this remainder:

$$p_i^{(t)} = \min(\sigma(s_i^{(t)}), r_i^{(t)})$$

The remainder is then updated: $r_i^{(t+1)} = r_i^{(t)} - p_i^{(t)}$. A token halts when its remainder approaches zero.

The final output representation for token i, y_i, is a weighted sum of its states from all iterations, ensuring the process is differentiable:

$$y_i = \sum_{t=1}^{L_{\max}} p_i^{(t)} h_i^{(t)}$$

4 Training Objective

The model is trained end-to-end by minimizing a composite loss function that balances task performance with architectural regularization.

4.1 The Composite Loss Function

The total loss is a weighted sum of three components:

$$L_{\text{total}} = L_{\text{task}} + \lambda_{\text{KL}} L_{\text{KL}} + \lambda_{\text{ent}} L_{\text{ent}}$$

4.2 KL Regularization for Halting ($L_{\mathbf{KL}}$)

To prevent the model from learning degenerate halting policies (e.g., always halting immediately or never halting), we regularize the learned halting distribution $P_i(t) = p_i^{(t)}$ against a target geometric prior $Q(t)$. The loss term is the Kullback-Leibler (KL) divergence between these distributions, averaged over all non-padded tokens:

$$L_{\text{KL}} = \mathbb{E}_{i \sim \text{batch}} \left[\text{KL}(P_i(t) \parallel Q(t)) \right] = \mathbb{E}_{i \sim \text{batch}} \left[\sum_t P_i(t) \log \frac{P_i(t)}{Q(t)} \right]$$

The prior $Q(t)$ is a geometric distribution with a mean set by the hyperparameter `target_halting_mean`. This encourages the model to learn a policy that respects the desired computational budget while maintaining token-level adaptivity.

4.3 Entropy Regularization for Bin Diversity ($L_{\mathbf{ent}}$)

To ensure that HQSA makes effective use of all its B bins, we add an entropy maximization term. This loss penalizes low-entropy bin assignments, preventing the model from collapsing all keys into a single bin. Let $\mathcal{H}(a_i)$ be the entropy of the bin assignment probabilities $\{a_{i,b}\}_{b=1}^{B}$ for token i. The loss is the negative mean entropy:

$$L_{\text{ent}} = -\mathbb{E}_{i \sim \text{batch}}[\mathcal{H}(a_i)] = \mathbb{E}_{i \sim \text{batch}} \left[\sum_{b=1}^{B} a_{i,b} \log a_{i,b} \right]$$

Minimizing L_{ent} encourages the assignment probabilities to be more uniform, promoting diversity and preventing mode collapse in the learned prototypes.

5 Experiments

5.1 Setup

Dataset and Baseline. We evaluate Epsilon on the IMDb movie review dataset for binary sentiment classification. The dataset is split into a training set of 22,500 samples and a validation set of 2,500 samples, with balanced classes. We compare Epsilon against a 'VanillaTransformer' baseline implemented in our codebase. To ensure a fair comparison, the baseline is configured with a fixed depth of 8 layers (matching Epsilon's `max_layers`) and a comparable number of parameters.

Implementation Details. All models were trained for 25 epochs using the AdamW optimizer with a cosine learning rate schedule and 300 warmup steps. Experiments were conducted on a single NVIDIA RTX 4070 Mobile GPU, with Automatic Mixed Precision (AMP) enabled for performance. The code for all experiments is made available under a PolyForm Noncommercial License 1.0.0.[1]

[1] `https://github.com/mescuwa/epsilon`

Table 1: Key hyperparameters for the main experimental run.

Hyperparameter	**Epsilon**	**Vanilla Baseline**
Model Dimension (d_{model})	256	256
Number of Heads (H)	4	4
FFN Dimension (d_{ffn})	1024	1024
Number of Layers (L)	8 (max)	8 (fixed)
Learning Rate	3e-4	3e-4
Batch Size	32	32
Dropout	0.1	0.1
Number of Bins (B)	16	N/A
Target Halting Mean	3.0	N/A
λ_{KL}	0.015	N/A
λ_{ent}	0.005	N/A
α_{res}	0.25	N/A

5.2 Performance Results

Our experiments demonstrate that Epsilon achieves superior performance to the baseline across all key metrics. As summarized in Table 2, Epsilon is not only more accurate but also significantly more efficient.

Table 2: Performance comparison on the IMDb validation set.

Model	**Parameters**	**Best Val. Acc. (%)**	**Avg. Epoch Time (s)**	**Relative Speedup**
Vanilla Transformer	386,186	85.08	≈ 60.4	1.0x
Epsilon (Ours)	**369,184**	**90.07**	≈ 40.5	$\approx$**1.5x**

Epsilon surpasses the baseline's accuracy by over 5 percentage points while being approximately 1.5 times faster per epoch. Notably, it achieves this with fewer trainable parameters, highlighting the efficiency of its weight-sharing, recurrent design.

5.3 Analysis of Adaptive Computation

The significant speedup of Epsilon is primarily attributable to its adaptive halting mechanism. Figure 2 plots the average number of 'EpsilonBlock' iterations performed per token over the course of training. The model quickly learns that it does not need the maximum of 8 layers to solve the task. The average halting depth converges to a stable value of approximately 1.78, indicating that, on average, the model uses less than 25% of its potential computational depth. This dynamic allocation of resources is the key driver of its efficiency.

5.4 Convergence Dynamics

As shown in Figure 3, Epsilon not only reaches a better final performance but also converges faster than the vanilla baseline. The validation accuracy curve for Epsilon rises more steeply in the initial epochs and plateaus at a higher value. This suggests that the architectural features of Epsilon, such as its regularized training objective and stable recurrent dynamics, contribute to a more efficient and effective learning process.

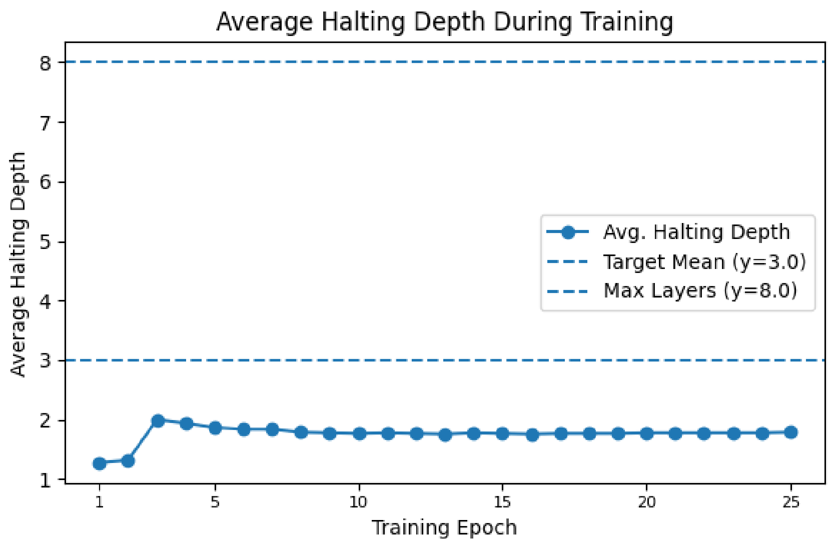

Figure 2: Average token halting depth during training. EPSILON learns to solve the task using fewer than 2 of the 8 available layers on average, demonstrating significant computational savings.

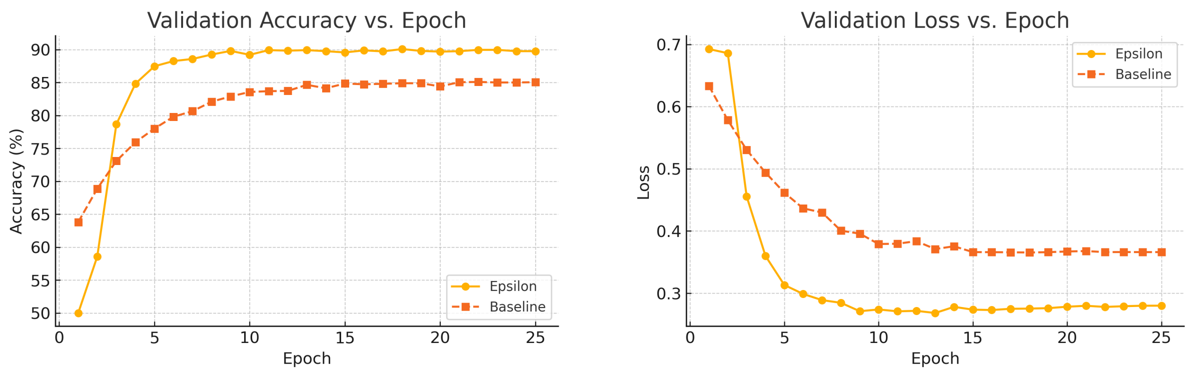

Figure 3: Comparison of validation accuracy (left) and loss (right) curves for EPSILON and the vanilla baseline over 25 epochs.

6 Conclusion

We have presented EPSILON, a novel Transformer architecture that achieves significant improvements in efficiency and accuracy through a principled combination of adaptive computation, quantized attention, and stability-enhancing mechanisms. By leveraging a recurrent, weight-sharing block with adaptive halting, EPSILON drastically reduces redundant computation. Its Histogram Quantized Scalar Attention (HQSA) mechanism provides a sub-quadratic alternative to standard self-attention without sacrificing performance. Our empirical results confirm that EPSILON outperforms a comparable vanilla Transformer baseline, delivering higher accuracy at a faster training speed with fewer parameters.

Future work could explore applying EPSILON to generative tasks or investigating more sophisticated, dynamic binning strategies for HQSA. The strong performance and efficiency of EPSILON make it a promising candidate for tackling tasks involving very long sequences, where standard Transformers are computationally infeasible.

Note on Scope

This whitepaper is intended as a practical introduction and design summary for the open-source Epsilon architecture. It is written in an accessible and informal tone to communicate the model's core ideas, performance, and implementation details. Future work may extend this document into a formal academic submission.

Implementation & License. All experiments are reproducible with the code and hyper-parameters in the GitHub repository. Code available at `https://github.com/mescuwa/epsilon`. The model and source are released under the PolyForm Noncommercial-1.0.0 license; contributions via pull request are welcome.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[2] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

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

[4] Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. In *International Conference on Machine Learning*, 2021.

[5] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[7] Ankit Roy, Michiel de Jong, and S-H. Gary Chan. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*, 2020.

[8] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Sinho Cheon, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, 2019.

[9] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[10] Sinxin Wang, Yizhe Zhang, and Michael R. Lyu. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

A Appendix

A.1 A.1. Stability Analysis of the EpsilonBlock

We argue that the 'EpsilonBlock' function, F, constitutes a contraction mapping under certain conditions, ensuring the stability of the recursive process. A map F is a contraction if there exists a constant $q < 1$ such that for any two states u, v:

$$\|F(u) - F(v)\|_{\ell_2} \leq q\|u - v\|_{\ell_2}$$

The update rule for the ‘EpsilonBlock‘ is a composition of residual blocks. Let’s analyze the Lipschitz constant of each component.

- **FFN:** The feed-forward network with spectral normalization on its linear layers and a 1-Lipschitz activation (GELU) has a Lipschitz constant $L_{\text{FFN}} \leq 1$.
- **CenterNorm:** This is an affine transformation with a fixed scaling factor $c = \sqrt{d/(d-1)}$, so its Lipschitz constant is $c \approx 1$.
- **HQSA:** The attention mechanism is a complex non-linear function. However, as it is fundamentally based on weighted averaging via softmax, its Lipschitz constant L_{attn} is bounded. While a tight analytical bound is difficult, empirical estimation via Jacobian monitoring in our code confirms it does not grow uncontrollably.
- **Residual Scaling α_{res}:** This is the most critical factor for ensuring contraction.

The full transformation can be expressed as $F(h) = h + \Delta(h)$, where $\Delta(h)$ represents the total update from the attention and FFN sub-layers. The Lipschitz constant of Δ is dominated by the sum of the Lipschitz constants of its components, scaled by α_{res}.

$$L_{\Delta} \approx \alpha_{\text{res}}(L_{\text{attn}} \cdot L_{\text{CN}} + L_{\text{FFN}} \cdot L_{\text{CN}})$$

For F to be a contraction, we need its Jacobian’s spectral norm to be less than 1. The Jacobian of F is $J_F = I + J_{\Delta}$. If the spectral norm $\sigma(J_{\Delta}) < 1$, then the iteration is stable. Since $\sigma(J_{\Delta}) \leq L_{\Delta}$, a sufficient condition for stability is $L_{\Delta} < 1$. Given that $L_{\text{FFN}}, L_{\text{CN}} \approx 1$, and assuming L_{attn} is a small constant (e.g., ≤ 2), we need:

$$\alpha_{\text{res}}(L_{\text{attn}} + 1) < 1$$

With our default $\alpha_{\text{res}} = 0.25$, this would require $L_{\text{attn}} < 3$, a condition that is likely met in practice. This small scaling factor effectively ”pulls” the overall mapping towards the identity, ensuring that it is a gentle perturbation and thus a contraction. The Banach fixed-point theorem then guarantees that the sequence $h^{(t+1)} = F(h^{(t)})$ converges to a unique fixed point h^*.