Recurrent vs. Standard Transformers: Parameter and Compute-Efficient Classification Across Sentiment and Domains

Chenxi Guo, Jiayi Peng, Chaowei Wang, Juncheng Han

1 Abstract

Transformer models achieve strong performance in natural language processing but incur substantial computational and memory costs that scale with depth. This work investigates whether recurrent Transformers with shared weights can provide a more parameter-efficient alternative to standard encoder architectures for text classification. We conduct a controlled comparison between a conventional 6-layer Transformer baseline and recurrent variants that reuse a smaller set of layers through iterative refinement, with all models trained from scratch, isolating the effect of weight sharing under matched effective depth and shared architectural components. Across sentiment classification and multi-domain review tasks, we evaluate parameter efficiency and robustness under varying data scales, input lengths, and domain shifts. Our results show that recurrent Transformers maintain competitive accuracy while substantially reducing parameter count, demonstrating favorable trade-offs between model capacity and efficiency. As a supplementary deployment-oriented analysis, we examine FP16 quantization and observe stable performance with further reductions in model size and inference latency. Together, these findings indicate that iterative recurrence can effectively substitute for depth stacking in encoder-based classification tasks, offering a practical path toward parameter-efficient Transformer deployment.

2 Introduction

Transformer architectures dominate modern natural language processing, but their strong performance typically comes at the cost of large parameter counts and substantial computational overhead. As model depth increases, memory footprint, training cost, and inference latency grow rapidly, limiting the practicality of standard Transformers in resource-constrained or

efficiency-sensitive settings. This motivates the study of parameter-efficient architectural alternatives that can preserve predictive performance while significantly reducing model size.

In this work, we investigate whether recurrent Transformers with shared weights can provide a more parameter-efficient substitute for standard Transformer encoders in text classification tasks. Rather than proposing a new architecture, our goal is to systematically analyze the trade-offs between standard and recurrent Transformers under controlled parameter budgets. We compare a conventional 6-layer Transformer baseline against recurrent variants that reuse a smaller set of layers through iterative refinement, isolating the effect of weight sharing on performance and efficiency. All models share identical modern components (Flash Attention, SwiGLU, RoPE, RMSNorm) to ensure that observed differences stem from architectural structure rather than implementation choices.

Beyond aggregate accuracy, we evaluate robustness across multiple stress settings, including reduced training data regimes, short versus long input sequences, and cross-domain generalization. For completeness, we additionally examine the effect of FP16 quantization on model size and inference latency, treating it as a deployment-oriented optimization rather than a primary modeling contribution. These experiments allow us to assess not only whether recurrent Transformers are parameter-efficient, but under what conditions such efficiency remains reliable. Our findings show that recurrent Transformers can maintain competitive performance with substantially fewer parameters across a range of realistic constraints, supporting their use as a practical and efficient alternative to standard Transformer encoders for classification tasks.

3 Related Work

Recent advances in depth-recurrent transformer architectures offer a promising direction for model compression. The Universal Transformer [1] introduced the concept of applying the same transformation block iteratively, effectively trading parameters for computation. More recent work has extended this idea: the Mixture-of-Recursions (MoR) framework [2] combines parameter sharing with adaptive computation, dynamically allocating depth per token. Similarly, the Tiny Recursive Model (TRM) [5] demonstrates that a compact ~7M parameter network with recursive reasoning can outperform much larger models on complexed tasks like ARC-AGI benchmarks.

These developments are particularly relevant in the context of reasoning-intensive tasks, where deeper processing has been shown to improve performance. By reusing layers across multiple iterations, recurrent transformers can achieve an effective depth that exceeds their physical layer count, enabling smaller models to develop richer representations without the parameter overhead of traditional deep networks.

Modern transformer implementations also benefit from architectural innovations such as Rotary Position Embedding (RoPE) [3], which encodes relative positional information directly

into attention computations, and SwiGLU activation functions [4], which provide improved gradient flow through gated linear units. These components have become standard in efficient transformer designs.

4 Methods

4.1 Theory

This section describes the core architectural components shared by our models. By strictly standardizing these modern components (RoPE, SwiGLU, RMSNorm), we ensure that observed performance differences arise solely from the architectural distinction (depth stacking vs. recurrence) rather than implementation discrepancies.

4.1.1 Transformer Layer Formulation

Both models are built upon a standard pre-norm Transformer encoder layer. Given an input representation $\mathbf{h}_l \in \mathbb{R}^d$ at layer l, a single Transformer layer computes:

$$\mathbf{h}'_l = \mathbf{h}_l + \mathrm{MHA}(\mathrm{Norm}(\mathbf{h}_l)),$$

$$\mathbf{h}_{l+1} = \mathbf{h}'_l + \mathrm{FFN}(\mathrm{Norm}(\mathbf{h}'_l)),$$

where MHA denotes multi-head self-attention, FFN is a position-wise feed-forward network, and Norm corresponds to RMSNorm.

4.1.2 Recurrent Transformer Formulation

The recurrent Transformer replaces depth stacking with iterative refinement using shared parameters. Let $\mathbf{h}^{(r)}$ denote the hidden state at recurrence step r. One recurrent iteration applies a stack of N shared Transformer layers:

$$\mathbf{h}^{(r+1)} = \mathcal{F}(\mathbf{h}^{(r)}) + \alpha \cdot \mathbf{h}^{(r)},$$

where $\mathcal{F}$ represents the shared Transformer block and α is a residual scaling factor. After R recurrence steps, the effective depth is $D_{\text{eff}} = N \times R$, allowing the model to match the representational depth of the baseline while using substantially fewer parameters.

4.1.3 SwiGLU Feed-Forward Network

We adopt the SwiGLU activation to improve gradient flow. Given input $\mathbf{x}$, the feed-forward network is:

$$\text{FFN}(\mathbf{x}) = \mathbf{W}_3(\text{SiLU}(\mathbf{W}_1\mathbf{x}) \odot (\mathbf{W}_2\mathbf{x})),$$

where $\odot$ denotes element-wise multiplication. SwiGLU's gating mechanism improves representation capacity compared to standard GELU-based FFNs.

4.1.4 Rotary Positional Embeddings (RoPE)

To encode positional information, we apply RoPE to query and key vectors. For position m and embedding dimension pair $(2i, 2i+1)$, the rotation is:

$$\begin{pmatrix} x'_{2i} \\ x'_{2i+1} \end{pmatrix} = \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix} \begin{pmatrix} x_{2i} \\ x_{2i+1} \end{pmatrix},$$

with frequencies $\theta_i = 10000^{-2i/d}$, ensuring attention scores depend only on relative positions.

4.1.5 RMS Normalization

We use RMSNorm for computational efficiency. Given input $\mathbf{x} \in \mathbb{R}^d$:

$$\text{RMSNorm}(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}} \cdot \gamma,$$

where γ is a learnable scale parameter. RMSNorm maintains training stability while reducing overhead by omitting mean-centering.

4.1.6 Loss Function for Optimization Objective

We optimize both models using the standard Cross-Entropy loss. Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is the input sequence and y_i is the ground truth label, the training objective is to minimize:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} \mathbb{1}(y_i = c) \log P(c|x_i; \theta)$$

where C is the number of classes, $\mathbb{1}(\cdot)$ is the indicator function, and $P(c|x_i;\theta)$ is the probability predicted by the model (after Softmax).

4.2 Model Architectures

We evaluate two encoder-only transformer architectures designed to isolate the effect of iterative recurrent refinement versus standard depth stacking. Both models utilize identical architectural components—Flash Attention, SwiGLU feed-forward networks, rotary positional embeddings (RoPE), and RMSNorm—to ensure a controlled comparison.

4.2.1 Baseline Transformer

The baseline follows a conventional pre-norm transformer encoder with 6 layers and a hidden dimensionality of 384. Table 1 summarizes the architectural configuration.

Table 1. Baseline Transformer Configuration

Component	Value
Number of layers	6
Hidden dimension	384
Attention heads	6
FFN intermediate size	1536
Total parameters	25,912,706

Each layer consists of a multi-head self-attention block followed by a feed-forward network (FFN), both wrapped with pre-norm residual connections:

$$h'_l = h_{l-1} + \text{MHA}(\text{Norm}_1(h_{l-1}))$$
$$h_l = h'_l + \text{FFN}(\text{Norm}_2(h'_l))$$

Classification is performed using the final hidden state corresponding to the [CLS] token [6]:

$$\hat{y} = \text{softmax}(W_c\, h_L[0])$$

4.2.2 Recurrent Transformer

The recurrent architecture employs iterative refinement to achieve an effective depth comparable to the baseline while using substantially fewer parameters. Instead of stacking 6 distinct layers, the model uses 3 shared layers unrolled for 2 iterations.

Table 2. Recurrent Transformer Configuration

Component	Value
Physical layers	3
Recurrent iterations	2
Hidden dimension	256
Attention heads	4
FFN intermediate size	1024
Total parameters	10,972,162

Let $h^{(r)}$ denote the hidden representation at iteration r. Each iteration applies the same 3-layer block:

$$h^{(r+1)} = F(h^{(r)}) + \alpha\, h^{(r)}, \quad \alpha = 0.5$$

Although only 3 layers are instantiated, the effective depth equals:

$$D_{\text{eff}} = N_{\text{layers}} \times N_{\text{iterations}} = 6$$

This design follows prior iterative-refinement encoder models and enables parameter-efficient depth scaling.

4.2.3 Rationale for Fixed Configurations

To ensure a meaningful and controlled comparison, architectural hyperparameters are held constant across both models:

1. Effective depth equivalence: Both architectures achieve a depth of 6 transformer layers.
2. Controlled parameter budget: The recurrent model reduces parameters by approximately 58% while maintaining comparable representational depth.
3. Consistent scaling principles: Both models use a 4× FFN expansion ratio and standard head dimensionality.

This controlled setup isolates the architectural contribution of recurrence, allowing us to analyze performance differences independent of model capacity.

4.2.4 Shared Architectural Components

1. Flash Attention for memory-efficient attention computation
2. SwiGLU feed-forward networks
3. Rotary Positional Embeddings (RoPE)
4. RMSNorm for stable pre-norm training dynamics

By controlling for these components, any observed performance differences can be attributed primarily to the structural distinction between stacked depth and recurrent iterative depth. While effective depth is matched numerically, we acknowledge that recurrent unrolling differs from independently parameterized layers in optimization dynamics; our goal is to control representational depth rather than enforce identical training trajectories.

4.3 Datasets Overview

To assess the generalization capability and computational efficiency of the recurrent transformer architecture, three sentiment classification datasets were employed: the Stanford Sentiment Treebank (SST-2), Yelp Reviews, and a composite Multi-Domain corpus. These datasets collectively span short-sequence, long-sequence, and cross-domain linguistic settings.

To enable a controlled and fair comparison, we deliberately subsampled each dataset to the same number of training, validation, and test instances (54,576 / 6,822 / 6,823), resulting in 68,221 samples per dataset. By matching dataset size across domains, we eliminate confounding effects arising from data scale and ensure that observed performance differences primarily reflect architectural efficiency rather than variations in training data volume.

4.3.1 Dataset Characteristics

4.3.1.1 SST-2 (Short-Sequence Domain)

SST-2 consists of concise movie review excerpts, with sequence lengths concentrated between 0 and 50 tokens. This dataset emphasizes sentiment cues embedded in short syntactic patterns. Label distributions across all splits exhibit a mild positive skew.

Table 3. SST-2 Label Distribution

Split	Label 0 (%)	Label 1 (%)	n

Split	Label 0 (%)	Label 1 (%)	n
Train	44.11	55.89	54,576
Validation	44.91	55.09	6,822
Test	45.04	54.96	6,823

4.3.1.2 Yelp Reviews (Long-Sequence Domain)

Yelp reviews include substantially longer paragraphs, with some sequences reaching approximately 1,000 tokens. This dataset enables evaluating the model's ability to capture long-range dependencies. The class distribution is nearly symmetric, minimizing confounding effects arising from label imbalance.

Table 4. Yelp Label Distribution

Split	Label 0 (%)	Label 1 (%)	n
Train	50.08	49.92	54,576
Validation	51.28	48.72	6,822
Test	51.08	48.92	6,823

4.3.1.3 Multi-Domain Dataset (Composite Setting)

This dataset integrates samples drawn from local business reviews, movie reviews, and online shopping reviews. Category proportions were strictly controlled to maintain balanced domain representation across splits. Text lengths span a wide range, including many long sequences.

Table 5. Multi-Domain Category Distribution

Split	Local business (%)	Movie review (%)	Online shopping (%)	n
Train	33.33	33.33	33.33	54,576
Validation	33.33	33.33	33.33	6,822
Test	33.33	33.33	33.33	6,823

5 Experiments and Results

5.1 Training Protocol

We implemented all models using PyTorch and trained them under identical conditions to ensure a fair comparison. The specific protocol is as follows, and the same training strategy is applied consistently across all experiments and data subsets.

1. Optimization Configuration

We utilize the AdamW optimizer with an initial learning rate of 3×10^{-5} and a batch size of 16. To maintain training stability and prevent gradient explosion, particularly in the recurrent layers, we apply gradient clipping with a maximum norm of 1.0.

1. Adaptive Scheduling

To facilitate convergence, we employ a ReduceLROnPlateau scheduler. The learning rate is dynamically decayed by a factor of 0.5 whenever the validation loss fails to improve for 2 consecutive epochs.

1. Early Stopping and Model Selection

We implement early stopping to prevent overfitting, terminating training if validation loss does not improve by a margin of 10^{-3} for 3 consecutive epochs. The final evaluation uses the model checkpoint that achieved the lowest validation loss, rather than the final training state.

5.1.1 Evaluation Methodology

We evaluate all models on the held-out test set to assess both predictive performance and efficiency. Evaluation focuses on three aspects: classification performance, parameter efficiency, and inference latency. Predictive performance is measured using Accuracy, Precision, Recall, and F1-score. Parameter efficiency is quantified by the total number of trainable parameters and model size (MB, assuming 32-bit precision). Inference efficiency is assessed by the average per-sample inference time (in milliseconds) over the full test set, capturing the runtime impact of recurrent unrolling relative to standard depth stacking.

5.2 Experimental Design & Results

5.2.1 SST-2 Benchmark Evaluation (Full Dataset)

Table 6 presents the evaluation results on the SST-2 dataset. While the baseline model achieves marginally higher accuracy (0.9061), the recurrent transformer demonstrates superior efficiency with minimal performance loss. Specifically, the recurrent model reduces the parameter count

and storage requirements by over 55% (11.0M vs. 25.9M parameters) while maintaining comparable inference latency. Although it exhibits a slightly more conservative decision boundary (higher precision, lower recall), the results confirm that recurrent depth-sharing offers a favorable trade-off, significantly saving memory with only negligible degradation in predictive quality.

Table 6. SST-2 Benchmark Performance

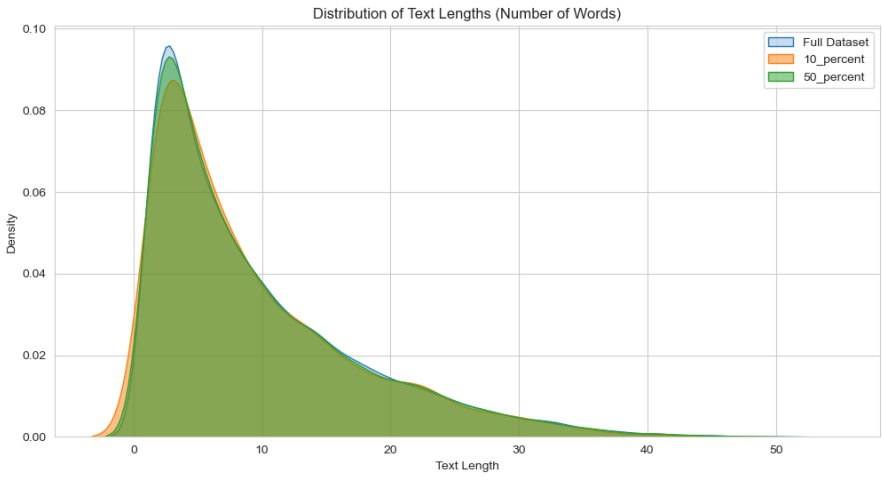
Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25,912,706	98.85	0.9061	0.9148	0.9174	0.9122	0.3706
Recurrent	10,972,162	41.86	0.9021	0.9100	0.9250	0.8955	0.3595

5.2.2 Data Size Sensitivity (50% / 10% SST-2)

To assess the impact of training data scale, we conducted controlled experiments on two stratified SST-2 subsets containing 10% and 50% of the original training set, with class distributions preserved and the validation/test sets unchanged.

The stratified sampling ensures both label proportions and sequence length distributions are closely matched to the original data (see Figure 1), enabling a fair comparison of model robustness under restricted resource regimes. Both architectures were trained and evaluated following the same protocol as for the full dataset.

Figure 1. Text length distributions for 10% and 50% subsets



Both models exhibit a clear reduction in accuracy and F1 as the available training data is restricted, which is expected under low-resource conditions. Notably, the performance gap between recurrent and baseline models narrows, with the recurrent model showing relatively less degradation at the smallest data scale.

Tables 7 and 8 summarize the primary results. These findings reinforce that the recurrent model preserves strong parameter and memory efficiency with only moderate performance loss, even when training data is severely limited. This extends and supports the overall robustness-efficiency trade-off observed throughout our study.

Table 7. Test results on 10% SST-2 subset

Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25,912,706	98.85	0.7661	0.7665	0.7798	0.7614	0.372
Recurrent	10,972,162	41.86	0.7453	0.7451	0.7696	0.7467	0.364

With 10% of the training data, the recurrent model maintains competitive performance in the extremely data-limited setting relative to the baseline while requiring less than half the parameters and memory.

Table 8. Test results on 50% SST-2 subset

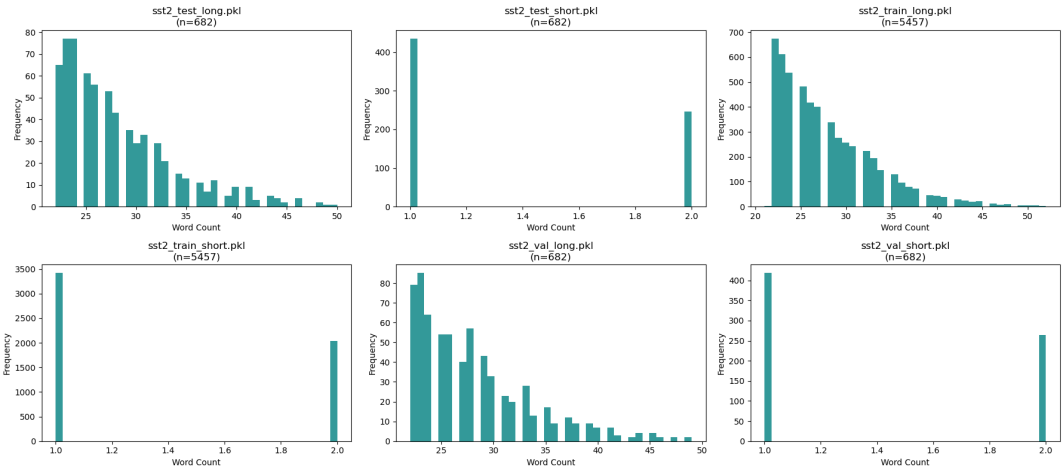
Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25,912,706	98.85	0.8903	0.8965	0.9001	0.8922	0.360
Recurrent	10,972,162	41.86	0.8791	0.8823	0.8960	0.8714	0.355

With 50% of the training data, both models recover much of their accuracy, and the efficiency advantage of the recurrent architecture persists with only a modest absolute gap in test performance.

5.2.3 Length-Based Sensitivity (Short vs Long on SST-2)

To study how input length influences model performance, we extract the shortest 30% and longest 30% of SST-2 samples and train each model separately on short-only and long-only subsets. The distributions shown in Figure 3 illustrate a clear separation between the two regimes: short subsets contain predominantly 1–2 token sequences, whereas long subsets span 20–50 tokens and exhibit substantially higher lexical diversity.

Figure 2. Word Count Distributions for Short and Long SST-2 Subsets



Model performance on the long-text subset is summarized in Table 9. Despite using fewer than half the parameters, the recurrent transformer slightly outperforms the baseline in accuracy and yields noticeably higher F1 and recall. This suggests that recurrent depth-sharing provides an advantage when modeling extended contextual dependencies. Although its inference latency is marginally higher, the improvement in predictive performance combined with a significantly smaller model footprint indicates a favorable efficiency–performance trade-off.

Table 9. Long-Sequence SST-2 Subset Performance

Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25,912,706	98.85	0.8666	0.8679	0.9144	0.8260	0.3115
Recurrent	10,972,162	41.86	0.8710	0.8795	0.8723	0.8867	0.3920

On the short-only subset (1–2 token inputs), both models show reduced performance due to the absence of contextual structure. The baseline attains slightly higher accuracy, while the recurrent model achieves higher recall with nearly identical F1 scores, indicating that under minimal context, the two architectures behave similarly and differ mainly in precision–recall trade-offs.

Table 10. Short-Sequence SST-2 Subset Performance

Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25,912,706	98.85	0.7610	0.7739	0.7971	0.7520	0.2985
Recurrent	10,972,162	41.86	0.7522	0.7732	0.7701	0.7763	0.2959

5.2.4 Cross-Domain Architectural Consistency Analysis on Yelp

SST-2 and Yelp are both sentiment analysis datasets, but they differ in how sentiment is expressed. SST-2 contains movie reviews, where opinions are usually short and more abstract. Yelp reviews, in contrast, describe specific experiences with businesses, such as food quality, service, or pricing. In addition to these conceptual differences, the Yelp dataset also contains much longer texts compared to SST-2.

To address this, we also train and evaluate the models on the Yelp dataset. The Yelp data is matched to SST-2 in size, and the same binary sentiment classification task is used. By keeping the two datasets the same size, we can reduce other sources of variation. This allows us to directly compare the performance of the Baseline and Recurrent models under domain shift and assess their robustness.

Table 11. Yelp dataset under same size as SST-2 Performance

Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25,912,706	98.85	0.8796	0.8721	0.8818	0.8627	0.1261
Recurrent	10,972,162	41.86	0.8956	0.8902	0.8913	0.8892	0.1346

The Baseline model has an accuracy of 0.8796 and an F1 score of 0.8721. The Recurrent model achieves a higher accuracy of 0.8956 and a higher F1 score of 0.8902, even though it uses less than half the number of parameters.

These results suggest that the relative performance of the two architectures does not change significantly when moving from movie reviews to Yelp reviews. The difference between these models remains small, but the recurrent transformer continues to achieve better results with a smaller model size, which shows its potential to be a powerful alternative.

5.2.5 Multi-Domain Review Classification (3-class)

We extend our evaluation to a three-class domain classification task (Movie, Yelp, Amazon) to assess whether the models can distinguish stylistic and distributional differences across review sources, beyond simple sentiment polarity. By matching each domain's data size to SST-2, this setting provides a balanced and more challenging multi-class benchmark for comparing the Baseline and Recurrent Transformers, offering clearer insight into architectural differences.

Both models achieve near-perfect performance on the multi-domain three-class task, with the recurrent transformer showing a slight but consistent improvement in accuracy and F1, as summarized in Table 12.

Table 12. Multi-Domain 3-Class Classification Results

Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25,913,091	98.85	0.9840	0.9840	0.9841	0.9840	0.3304
Recurrent	10,972,419	41.86	0.9865	0.9865	0.9865	0.9865	0.3228

5.2.6 Supplementary Deployment Analysis: Precision and Quantization Effects

To evaluate deployment characteristics, we compare model architectures and numerical precision with an emphasis on memory footprint, inference latency, and robustness to FP16 quantization. The Recurrent-Light (Benchmark, hidden size = 256) model serves as the primary reference, as it matches the recurrent configuration used in the main experiments. The Baseline (Benchmark) employs a standard 6-layer transformer with matched effective depth, providing a consistent non-recurrent comparison.

In addition, we include a Recurrent (Wider, hidden size = 384) variant as an auxiliary experiment. This configuration is not part of the benchmark comparison and is introduced solely to isolate the effect of recurrent weight sharing under FP16 quantization while controlling for representational width. Together, these settings disentangle architectural form, capacity, and numerical precision in a deployment-focused evaluation.

5.2.6.1 Experimental Configurations

We evaluate three model families, each under FP32 and FP16 precision:

1. Baseline Transformer (Benchmark) A standard 6-layer transformer with hidden size = 384, serving as the non-recurrent reference architecture.

2. Recurrent Transformer (Wider Variant) A recurrent transformer with depth sharing (3 layers iterated twice, effective depth = 6) and hidden size = 384, introduced to assess quantization robustness under matched capacity.

3. Light-Recurrent Transformer (Benchmark) A parameter-efficient recurrent transformer with depth sharing and reduced width (hidden size = 256), representing the main experimental configuration and deployment benchmark.

5.2.6.2 Quantitative Results

Table 13. Deployment Performance Comparison under FP32 and FP16 Precision.

Model (Configuration)	Precision	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline (Benchmark)	FP32	25.9M	98.85	0.9022	0.9125	0.9036	0.9215	0.411
Baseline (Benchmark)	FP16	25.9M	49.42	0.9021	0.9128	0.9034	0.9213	0.246
Recurrent (Wider)	FP32	18.8M	71.78	0.9100	0.9180	0.9226	0.9138	0.405
Recurrent (Wider)	FP16	18.8M	35.89	0.9100	0.9180	0.9221	0.9134	0.241
Recurrent-Light (Benchmark)	FP32	11.0M	41.86	0.8977	0.9082	0.9017	0.9146	0.403
Recurrent-Light (Benchmark)	FP16	11.0M	20.93	0.8977	0.9082	0.9017	0.9146	0.230

Under the benchmark configuration, the Recurrent-Light (Benchmark) model exhibits a modest reduction in accuracy compared to the Baseline (Benchmark) transformer (0.8977 vs. 0.9022), reflecting an intentional trade-off between capacity and efficiency. This performance difference is small relative to the deployment benefits: the Recurrent-Light (Benchmark) model uses less than half the parameters (11.0M vs. 25.9M) and achieves nearly an 80% reduction in storage under FP16 precision (20.9 MB vs. 98.9 MB), while maintaining comparable inference latency. Across all architectures, FP16 quantization proves highly stable, preserving accuracy and F1 while reducing model size by approximately 50% and improving inference speed by 40–45%. Moreover, the Recurrent (Wider, hidden size = 384) model outperforms the Baseline (Benchmark) under matched width, confirming that recurrent weight sharing is not inherently limiting. Overall, these results indicate that

recurrent transformers, particularly the Recurrent-Light (Benchmark) configuration with FP16 precision, offer an attractive deployment-efficient alternative with minimal impact on predictive performance. This analysis is not intended as a primary performance comparison, but as evidence that recurrent weight sharing does not impede quantization robustness under matched capacity.

6 Conclusion

6.1 Key Findings

We conducted a controlled comparison between standard depth-stacked Transformers and recurrent Transformers with shared weights for text classification. On short-text benchmarks such as SST-2, the standard Transformer achieves marginally higher accuracy, but at more than twice the parameter count and memory footprint. In contrast, the recurrent architecture preserves competitive performance while reducing parameters by approximately 58%, demonstrating that iterative refinement can effectively substitute explicit depth stacking in low-context settings.

On longer and more complex inputs, including Yelp reviews and multi-domain classification tasks, recurrent Transformers consistently match or outperform the baseline despite their smaller size, suggesting particular advantages for modeling long-range dependencies. Deployment-oriented experiments further show that FP16 quantization is stable across architectures, reducing memory usage by about 50% and improving inference latency with negligible performance loss. Overall, recurrent Transformers offer a strong efficiency–performance trade-off, especially under long-context and cross-domain conditions.

6.2 Limitations

This study focuses on encoder-only classification tasks, and the observed benefits may not directly transfer to generative or sequence-to-sequence settings. Moreover, effective depth matching is approximate, as recurrent unrolling differs from independently parameterized layers in optimization dynamics, and sequential recurrence may limit parallelism on some hardware.

6.3 Future Directions

Future work may investigate adaptive recurrence mechanisms that vary refinement depth by input complexity, as well as extensions to large-scale pretraining and combinations with other parameter-efficient techniques.

7 References

[1] Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Kaiser, Ł. (2018). Universal Transformers. *arXiv preprint arXiv:1807.03819*.

[2] Anonymous. (2025). Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-level Computation. *arXiv preprint arXiv:2507.10524*.

[3] Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2021). RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv preprint arXiv:2104.09864*.

[4] Shazeer, N. (2020). GLU Variants Improve Transformer. *arXiv preprint arXiv:2002.05202*.

[5] Jolicoeur-Martineau, A., et al. (2025). Less is More: Recursive Reasoning with Tiny Networks. *arXiv preprint arXiv:2510.04871*.

[6] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT 2019*, 4171-4186.