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

Chenxi Guo, Jiayi Peng, Chaowei Wang, Junchen Han

1 Abstract

2 Introduction

- This section includes a summary of the topic, why it is important, why the reader should continue, what work has been done in the past by other research groups, what are the “different points of views”/interpretations in the literature, what are you exploring, what questions are you trying to address, what are your goals and hypothesis, etc
- For a data driven study, the Introduction is about the data science question and the topics you plan to explore. It helps the reader to understand what the data science question is, what the supporting topics and issues are, and what the overall research area is all about. An introduction allows the reader to “get to know” the data science question and related areas of interest. Ideally, an introduction should make the reader *care* about the topics and read more. The Introduction is NOT about the datasets, variables, methods or models. The Intro should not contain any information about the dataset or the data cleaning, prep, processing, etc. These things should go into the methods section. Introductions can and should include basis, background, history, the state-of-the-art, images, references, etc. An introduction will also help the reader to understand who the topics affect and why the topics matter.
- The following bullets show common components of an introduction section, they come from the following source Links to an external site..
  - Introduce your topic
    - The first job of the introduction is to tell the reader what your topic is and why it’s interesting or important. This is generally accomplished with a strong opening hook.The hook is a striking opening sentence that clearly conveys the

relevance of your topic. Think of an interesting fact or statistic, a strong statement, a question, or a brief anecdote that will get the reader wondering about your topic.

- Describe the background
  - Part of the introduction is a concise literature review to demonstrate that the writer is familiar with the relevant research. You can provide an overview of the most relevant research that has already been conducted. This is a sort of miniature literature review, a sketch of the current state of research into your topic, boiled down to a few paragraphs.
  - This should be informed by genuine engagement with the literature. Your search can be less extensive than in a full literature review, but a clear sense of the relevant research is crucial to inform your own work.
- Establish your research problem
  - In an empirical research paper, try to lead into the problem on the basis of your discussion of the literature. Think in terms of these questions:
    - What research gap is your work intended to fill?
    - What limitations in previous work does it address?
    - What contribution to knowledge does it make?
- Specify your objective(s)
  - The research question is the question you want to answer in an empirical research paper. Present your research question clearly and directly, with a minimum of discussion at this point. The rest of the paper will be taken up with discussing and investigating this question; here you just need to express it.
- Map out your paper
  - The final part of the introduction is often dedicated to a brief overview of the rest of the paper.
  - In a paper structured using the standard scientific "introduction, methods, results, discussion" format, this isn't always necessary. But if your paper is structured in a less predictable way, it's important to describe the shape of it for the reader. If included, the overview should be concise, direct, and written in the present tense.

3 Theory

4 Datasets & Data Preparation

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. For comparability, all datasets were standardized to identical training, validation, and test sizes (54,576 / 6,822 / 6,823).

4.1 Dataset Characteristics

4.1.1 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 1. SST-2 Label Distribution

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

4.1.2 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 2. 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

4.1.3 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 3. 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

4.2 Preprocessing Pipeline

A unified text preprocessing procedure was applied to all datasets to ensure consistency and reduce non-semantic noise before tokenization. The steps were as follows:

1. Normalization of text, including lowercasing and standardization of whitespace to remove redundant spacing or line breaks.

2. Removal of non-linguistic artifacts such as HTML tags and URL patterns commonly found in web-derived data.

3. Standardization of punctuation by collapsing repeated punctuation marks (e.g., sequences of exclamation points or ellipses) into single instances, preventing artificial inflation of sequence length.

This preprocessing strategy ensures that models operate over semantically meaningful content and reduces variability introduced by formatting irregularities or source-specific artifacts.

5 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.

5.1 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 4. Baseline Transformer Configuration

Component	Value
Number of layers	6

Component	Value
Hidden dimension	384
Attention heads	6
FFN intermediate size	1536 (4× hidden)
Total parameters	9.8M

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

$$\begin{aligned} h'_l &= h_{l-1} + \mathrm{MHA}(\mathrm{Norm}_1(h_{l-1})) \\ h_l &= h'_l + \mathrm{FFN}(\mathrm{Norm}_2(h'_l)) \end{aligned}$$

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

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

5.2 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 5. Recurrent Transformer Configuration

Component	Value
Physical layers	3
Recurrent iterations	2
Hidden dimension	256
Attention heads	4
FFN intermediate size	1024
Total parameters	3.4M

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.

5.3 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 65% 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.

5.4 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.

5.4.0.1 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).

6 Experiments & Results

6.1 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:

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.

6.1.1 Evaluation Methodology

We evaluate the trained models on the held-out test set to assess both predictive performance and computational efficiency. Our evaluation framework consists of three key components:

1. Classification Performance

To measure the model's ability to correctly categorize sequences, we report standard classification metrics: Accuracy, Precision, Recall, and F1-Score. These metrics provide a comprehensive view of the model's predictive power, balancing the trade-off between false positives and false negatives.

1. Parameter Efficiency

We quantify the spatial complexity of each architecture by calculating the total number of trainable parameters and the corresponding memory footprint (in MB, assuming 32-bit precision). This allows us to verify the hypothesis that the recurrent architecture significantly reduces model size while maintaining capacity.

1. Inference Latency

To assess temporal efficiency, we measure the real-world inference speed. We calculate the average inference time per sample (in milliseconds) over the entire test set. This metric captures the computational cost of the recurrent unrolling process compared to the standard stacked layers of the baseline.

6.2 Experimental Design & Results

6.2.1 SST-2 Benchmark Evaluation (Full Dataset)

We evaluate the baseline transformer and the recurrent transformer on the full SST-2 dataset to examine their predictive performance, parameter efficiency, and inference characteristics. Table 1 summarizes the quantitative results, while Figure 1 visualizes the accuracy–latency trade-off using bubble size to represent overall model storage cost.

Across all key metrics, the recurrent model provides a favorable balance between compactness and predictive quality. Although it contains less than half the parameters of the baseline model (11.0M vs. 25.9M; 41.9 MB vs. 98.8 MB), it achieves higher accuracy (0.7993 vs. 0.7901) and an improved F1 score (0.8031 vs. 0.7946). Similar gains are observed in precision and recall, suggesting that recurrent depth-sharing does not compromise representational capacity on short-sequence sentiment classification tasks such as SST-2.

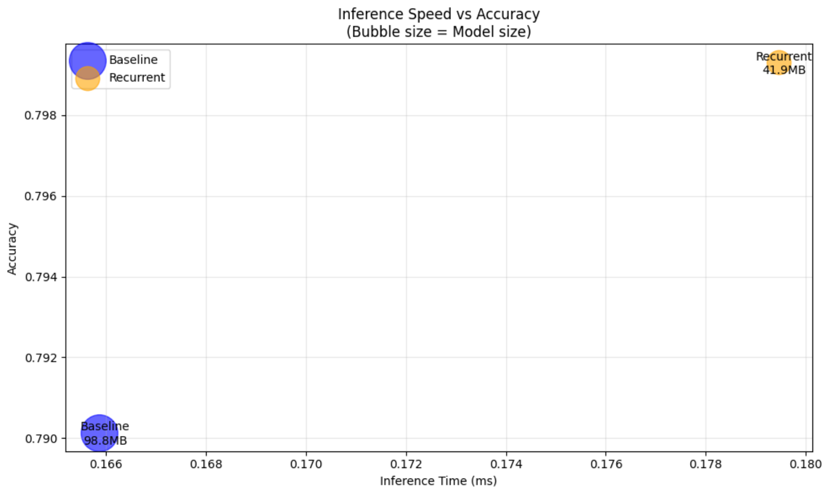
As illustrated in Figure 1, the recurrent model occupies a more desirable position along the accuracy–efficiency frontier, delivering better accuracy at less than half the model size.

Overall, these results indicate that the recurrent transformer provides a more parameter-efficient alternative to the conventional transformer for SST-2 sentiment analysis, achieving superior or comparable performance while maintaining a significantly smaller computational footprint.

Table 6. SST-2 Benchmark Performance

Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25,912,706	98.85	0.7901	0.7946	0.7919	0.7973	0.1659
Recurrent	10,972,162	41.86	0.7993	0.8031	0.8022	0.8041	0.1795

Figure 1. Inference Speed vs. Accuracy (Bubble size = model storage cost)

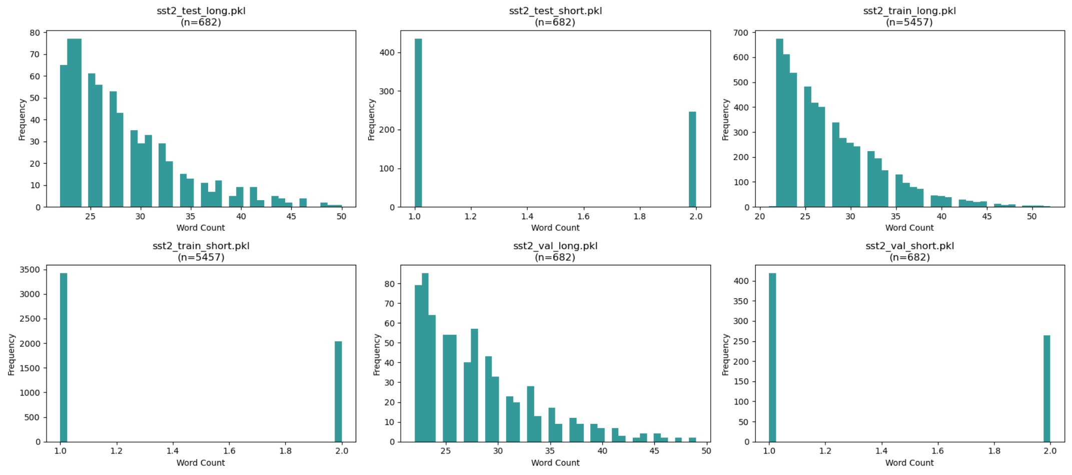


6.2.2 Data Size Sensitivity (50% / 10% SST-2)

6.2.3 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 3. 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

6.2.4 Cross-Domain Architectural Consistency Analysis on Yelp

6.2.5 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

7 Conclusions

7.1 References