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

Chenxi Guo, Jiayi Peng, Chaowei Wang, Juncheng Han

In recent years, transformer-based models dominate NLP, but critical questions remain:

How robust are they with scarce data?
Do they handle varying text lengths well?
How effectively do they generalize across domains?

These real-world challenges drive our investigation.

To systematically address these questions, we designed and implemented a **Recurrent Transformer** variant and conducted a comparative analysis against a strong **Baseline Transformer** model.

Our core objective is to understand how these architectures behave under various “stress tests” and identify their relative strengths and weaknesses.

Our research unfolds through experiments across four key dimensions:

Text Length Robustness:

Analyzing performance on short-only vs. long-only text subsets

Data Scale Robustness:

Testing sensitivity to data scarcity by varying dataset sizes

Cross-Dataset Sentiment Robustness:

Testing binary sentiment robustness on Yelp with SST-2–matched data size.

Broader Generalization:

Testing robustness on diverse classification benchmarks to verify performance on complex, multi-class scenarios.

Through these experiments, we aim to provide actionable insights for practitioners choosing between model complexity and efficiency.

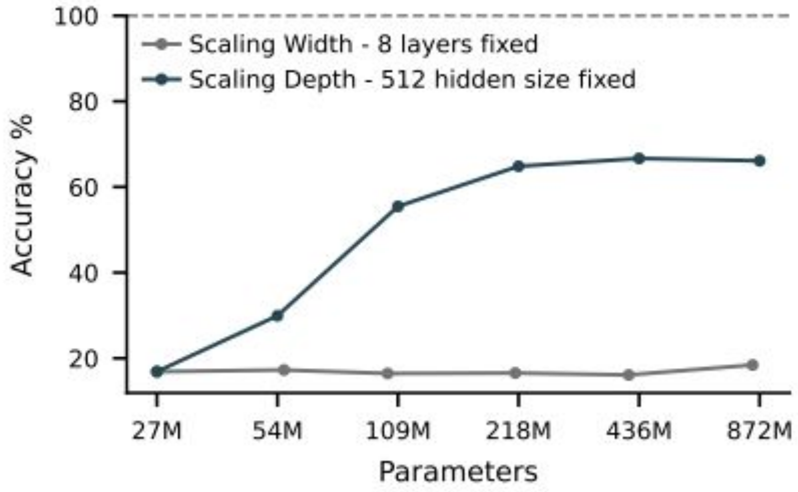
Methodology

Background & theory

Scaling behavior: depth-performance

- The deeper the model, the better the performance

- What about fixing parameter size?
    - Pure depth-recurrent operation!



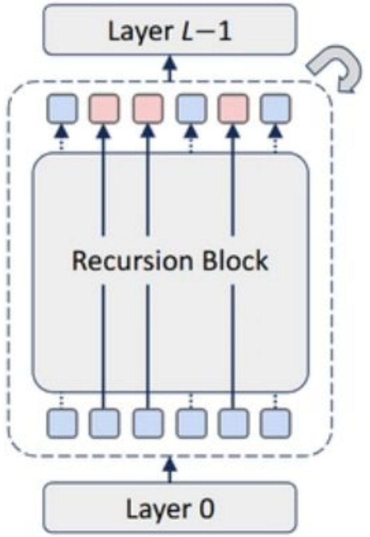
Gradient of depth-recurrent modeling

Standard Transformer: $\frac{\partial \mathcal{L}}{\partial \theta_l} = \frac{\partial \mathcal{L}}{\partial h^{(l)}} \cdot \frac{\partial h^{(l)}}{\partial \theta_l}$

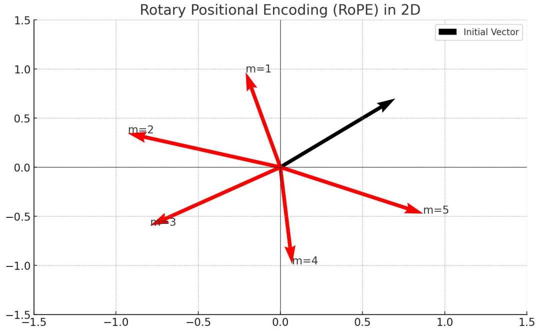
Depth-recurrent Transformer if recycled L times:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{t=0}^{L-1} \frac{\partial \mathcal{L}}{\partial H^{(t)}} \cdot \frac{\partial H^{(t)}}{\partial \theta}$$

Observation: Weight sharing amplifies quantization error.



RoPE (Rotary Positional Embedding)



Intuition: rotations on complex plane.

- Given a token's **q** (or **k**) vector with only 2 dimensions as (x_0, x_1) with interpretation $z = x_0 + ix_1$
- Given position ***m***, we rotate by angle θ : $z' = z \cdot e^{im\theta} = (x_0 + ix_1)(\cos m\theta + i \sin m\theta)$
- Or by rotation matrix:

$$\mathbf{x}' = R_m \mathbf{x} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

The attention score to another token in ***n***-th position will be: $(R_m \mathbf{q})^T (R_n \mathbf{k}) = \mathbf{q}^T R_{n-m} \mathbf{k}$

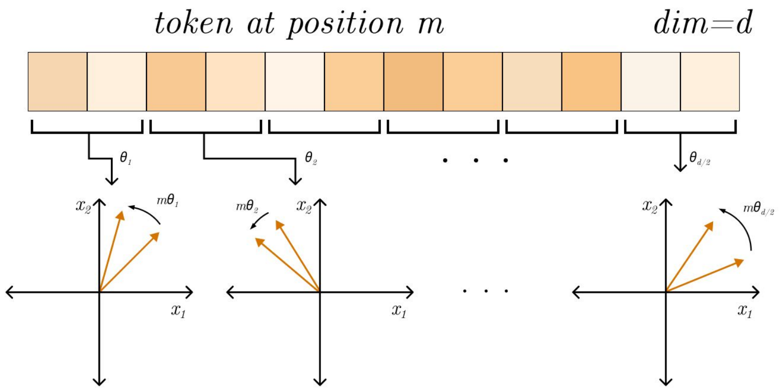
RoPE in practice

Intuition: We decompose the ***d***-dimension space into ***d/2*** independent rotational pairs, with each pair assigned a distinct frequency: $\theta_i = 10000^{-2i/d}, \quad i \in \{0, 1, \ldots, \frac{d}{2} - 1\}$

$$\mathrm{RoPE}(x, m) = \begin{bmatrix} x_0 \cos(m\theta_0) - x_1 \sin(m\theta_0) \\ x_0 \sin(m\theta_0) + x_1 \cos(m\theta_0) \\ x_2 \cos(m\theta_1) - x_3 \sin(m\theta_1) \\ \vdots \end{bmatrix}$$

Observation: Higher dimensions ***d***

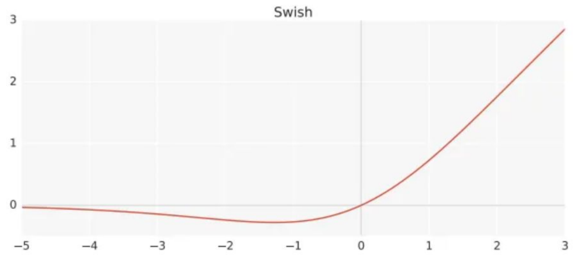
prevent positional aliasing (collisions).



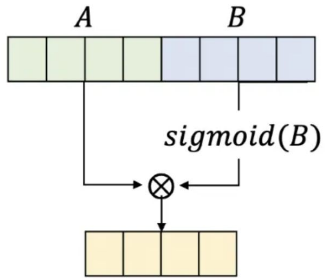
SwiGLU

$$\text{swish}(x) = x\,\text{sigmoid}(\beta x) = \frac{x}{1+e^{-\beta x}}$$

$$GLU(x, W, V, b, c) = sigmoid(xW + b) \otimes (xV + c)$$



Swish Graph

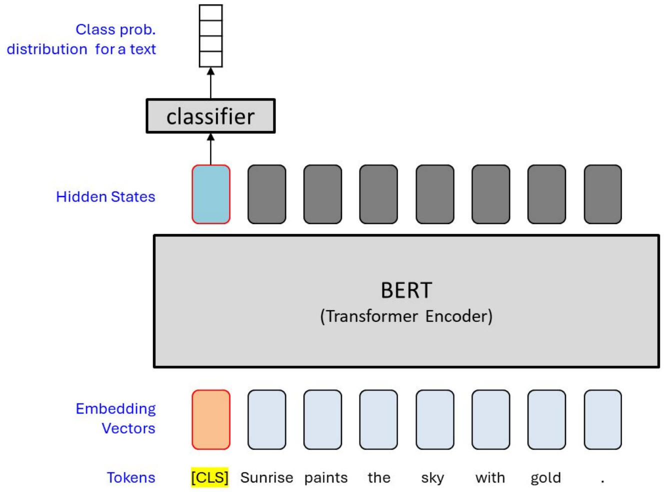


Gated Linear Unit

$$SwiGLU(x, W, V, b, c\beta) = Swish_{\beta}(xW + b) \otimes (xV + c)$$

[Link to Demo]

Implementation for classification



- [CLS] token at position 0
    - Example: [CLS] this movie was great ! [SEP]

During bidirectional self-attention: $\alpha_{0,j} = \text{Softmax}\left(\frac{q_0 \cdot k_j^T}{\sqrt{d_k}}\right)$

Aggregation: $h_{cls}^{(\ell+1)} = \sum_{j=0}^{T-1} \alpha_{0,j} \cdot v_j$ where ℓ is current layer index.

- Extract at final layer: $h_{cls} = \text{hidden\_states}[:,0]$
    - Apply Dropout regularization for training: $\tilde{h}_{cls} = \text{Dropout}(h_{cls})$
    - Compute logits: $\text{Logits} = W_{cls} \cdot \tilde{h}_{cls} + b_{cls}$
    - Map to a probability distribution of classes: $p_i = \text{Softmax}(\text{Logits})_i, \quad i \in \{1, \ldots, C\}$

EXPERIMENTS

Data Distribution

SST-2 Label Distribution

- **Train (n = 54,576)**
  - Label 0: 44.11%
  - Label 1: 55.89%
- **Val (n = 6,822)**
  - Label 0: 44.91%
  - Label 1: 55.09%
- **Test (n = 6,823)**
  - Label 0: 45.04%
  - Label 1: 54.96%

Yelp Label Distribution

- **Train (n = 54,576)**
  - Label 0: 50.08%
  - Label 1: 49.92%
- **Val (n = 6,822)**
  - Label 0: 51.28%
  - Label 1: 48.72%
- **Test (n = 6,823)**
  - Label 0: 51.08%
  - Label 1: 48.92%

Multi-Domain Category Distribution

- **Train (n = 54,576)**
  - local_business_review: 33.33%
  - movie_review: 33.33%
  - online_shopping: 33.33%
- **Val (n = 6,822)**
  - local_business_review: 33.33%
  - movie_review: 33.33%
  - online_shopping: 33.33%
- **Test (n = 6,822)**
  - local_business_review: 33.33%
  - movie_review: 33.33%
  - online_shopping: 33.33%

Baseline vs. Recurrent Transformers: Standard Benchmark Evaluation

Setting	Baseline Model	Recurrent Model
Hidden Size	384	256
Num Hidden Layers	6	3 (recurrent depth 2)
Attention Heads	6	4
Intermediate Size	1536	1024
Dropout	0.1	0.1
Recurrent Depth	—	2
Residual Scale	—	0.5

Both models include the same modern enhancements—Flash Attention, SwiGLU, RoPE, and RMSNorm. We train both models under identical settings—5 epochs, batch size 16, learning rate 3e-5, warmup 100 steps, and evaluation every 50 steps—and use the checkpoint with the lowest validation loss for final test evaluation to ensure a fair and controlled comparison.

Baseline vs. Recurrent Transformers: Standard Benchmark Evaluation

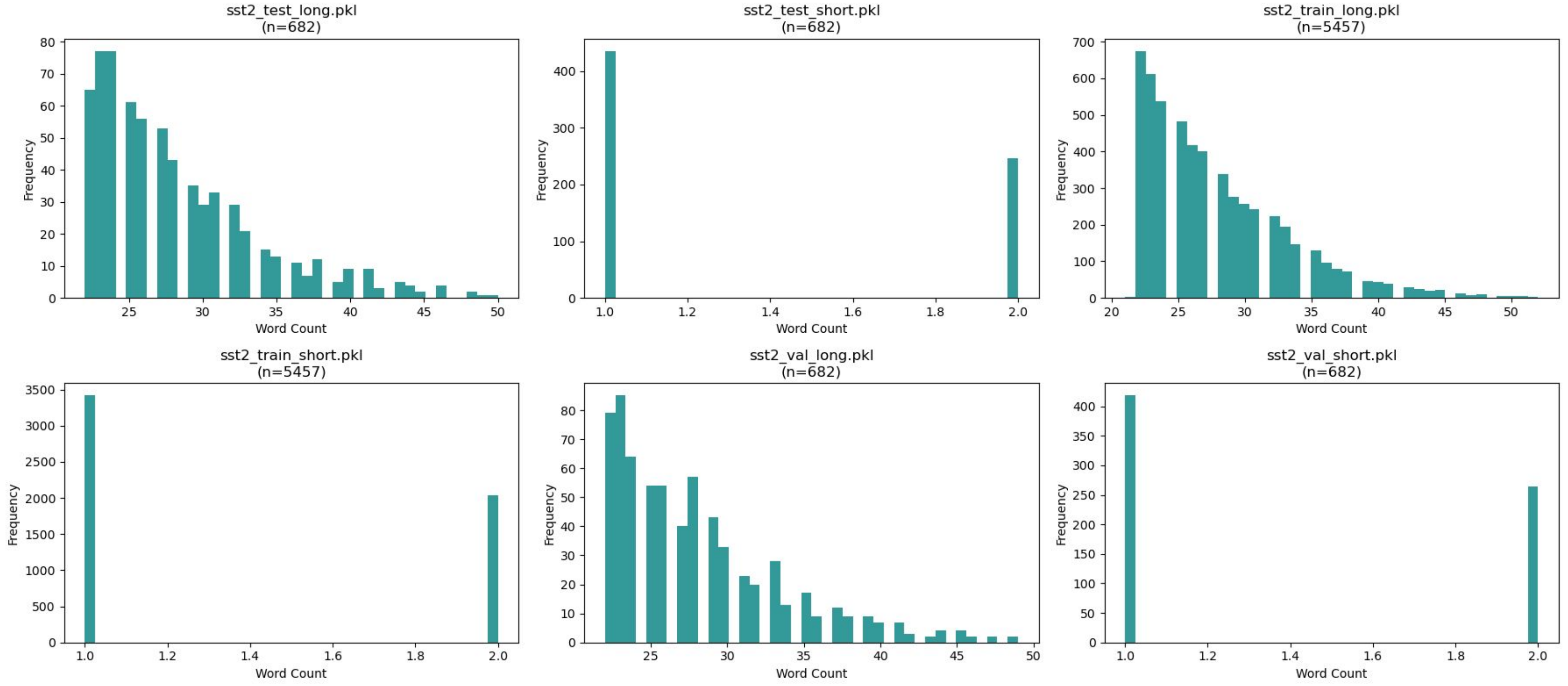
Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25,912,706	98.85	0.7901	0.7946	0.7919	0.7973	0.1659
Recurrent	10,972,162	41.86	0.7993	0.8031	0.8022	0.8041	0.1795

A controlled comparison of Baseline vs. Recurrent Transformers under short vs. long input conditions

We compare Baseline and Recurrent Transformers under short-text and long-text conditions to evaluate how input length influences model performance.

We construct the shortest 30% and longest 30% SST-2 subsets and train each model separately on short-only and long-only data to isolate the effect of sequence length.

A controlled comparison of Baseline vs. Recurrent Transformers under short vs. long input conditions



Performance of Baseline vs. Recurrent Transformers on Long Inputs

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

This table shows that the Recurrent Transformer achieves slightly higher accuracy and F1 than the Baseline model despite using less than half the parameters, though with a small increase in inference time.

Performance of Baseline vs. Recurrent Transformers on Short Inputs

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

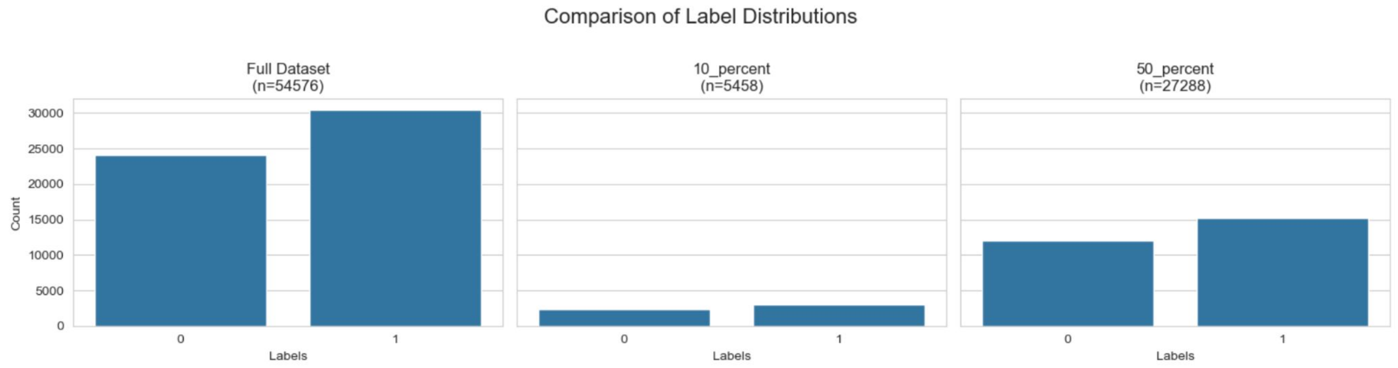
On short texts, both models perform also similarly, with the Baseline slightly higher in precision and the Recurrent slightly higher in recall. Recurrent's higher recall means it catches more true positives on short texts, missing fewer positive samples.

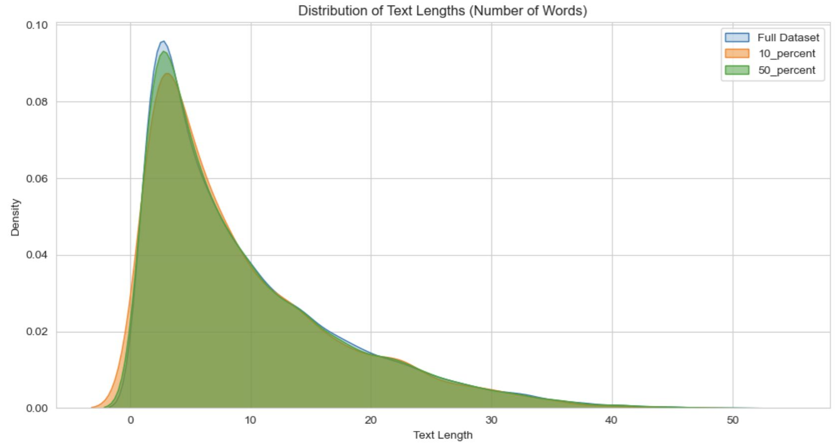
A controlled comparison of Baseline vs. Recurrent Transformers under 10% & 50% data size conditions

We also investigated how varying training data scales affects the performance of Baseline vs. Recurrent Transformers.

We trained each model on 10% and 50% subsets of the SST-2 dataset to measure their robustness and quantify the performance difference caused by data volume.

A controlled comparison of Baseline vs. Recurrent Transformers under 10% & 50% data size conditions





Consistent Class Distribution & Similar Sentence Length Distribution

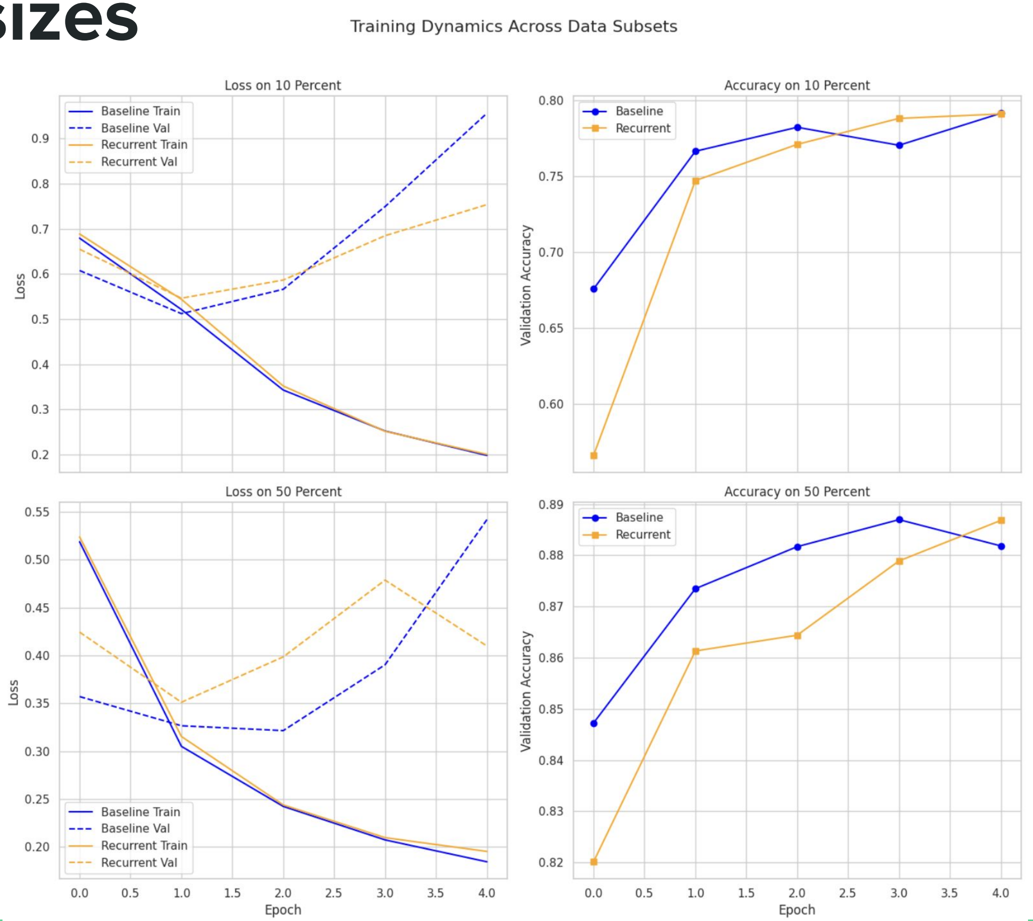
Data subsets are balanced and unbiased, providing a solid foundation for the upcoming robustness experiments.

Performance of Baseline vs. Recurrent Transformers under 10% & 50% data sizes

Recurrent models require more iterations to reach baseline performance on 10% data (starting at 57% vs. 78% accuracy at epoch 1)

Baseline overfits severely on 10% data (validation loss: 0.55➔0.95), while recurrent shows better regularization with stable validation curves

Performance converges at 50% data (~88% accuracy), suggesting architecture differences diminish with scale



Performance of Baseline vs. Recurrent Transformers under 10% & 50% data sizes

Model	Subset	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	10_percent	25,912,706	98.85	0.7661	0.7665	0.7765	0.7661	0.3198
Recurrent	10_percent	10,972,162	41.86	0.7453	0.7451	0.7620	0.7453	0.3084
Baseline	50_percent	25,912,706	98.85	0.8852	0.8854	0.8862	0.8852	0.3299
Recurrent	50_percent	10,972,162	41.86	0.8655	0.8658	0.8708	0.8655	0.3075

Efficiency-Performance Tradeoff: Recurrent achieves 97% of baseline accuracy (0.866 vs 0.885 at 50%) with only 42% of parameters (11M vs 26M) and 58% smaller model size (42 MB vs 99 MB)
Consistent Speed Advantage: Recurrent inference is faster across all conditions (0.31ms vs 0.32ms), demonstrating that parameter reduction translates to real computational savings
Balanced Metrics: Both models maintain healthy precision-recall balance (difference <0.01), with recurrent showing no metric degradation despite fewer parameters

Cross-Dataset Sentiment Robustness: Testing binary sentiment robustness on Yelp with SST-2–matched data size.

- SST-2 consists of movie reviews and mainly reflects more abstract sentiment expressions.
- Yelp reviews focus more on concrete opinions about products, services, and user experience.

We test these two datasets with matched data sizes under the same binary sentiment classification setting to evaluate how models generalize across different domain characteristics.

This allows us to directly compare the performance of the Baseline and Recurrent models under domain shift and assess their robustness.

Comparing Baseline and Recurrent Transformers on Yelp Subsets

Model	Parameters	Size (MB)	Accuracy	F1	Precision	Recall	Inference (ms)
Baseline	25912706	98.85	0.8796	0.8721	0.8818	0.8627	0.1261
Recurrent	10972162	41.86	0.8956	0.8902	0.8913	0.8892	0.1346

For the Yelp dataset, we observe the same pattern as with SST-2: the Recurrent model achieves higher accuracy and F1 scores than the baseline model.

Extension Experiment Overview: Multi-Domain Review Classification

We add a **three-class** domain classification task (Movie, Yelp, Amazon) to test whether models can distinguish stylistic differences across review sources, beyond simple sentiment polarity.

By matching each domain’s data size to SST-2, this extension provides **a balanced and practical multi-class setting** to compare Baseline and Recurrent Transformers, providing a harder test of architectural differences beyond sentiment classification.

Multi-Domain Review 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

For the multi-domain classification task, both models achieve extremely high performance, with the Recurrent Transformer slightly outperforming the Baseline across all metrics while using less than half the parameters and maintaining similar inference speed.

Our Demo

DEMO_LINK

CONCLUSION

Despite using less than half the parameters and model size, the Recurrent model delivers comparable performance to the Standard Transformer, making it a highly efficient alternative.

Conclusion

Across SST-2 and Yelp (binary sentiment classification) and the multi-domain three-class task, the Recurrent Transformer performs slightly better overall.

On long texts, Recurrent shows an advantage. On short texts, the Baseline performs slightly better, especially in precision. Recurrent is more recall-oriented, capturing more true positives.

Under limited data scales, Recurrent requires more training iterations but provides better regularization. As data volume grows, both architectures achieve comparable results

Using less than half the parameters, the Standard Transformer performs slightly better overall, but the Recurrent model remains competitive

Overall differences are small, indicating both architectures are competitive; given that the Recurrent model uses less than half the parameters and model size, achieving similar performance already demonstrates strong efficiency.

Thank you!

Reference

1. J. Hao, X. Wang, B. Yang, L. Wang, J. Zhang, and Z. Tu, “Modeling Recurrence for Transformer,” in Proc. Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Minneapolis, MN, USA, 2019, pp. 1198–1207.
2. G. Wang, J. Li, Y. Sun, X. Chen, C. Liu, Y. Wu, M. Lu, S. Song, and Y. A. Yadkori, “Hierarchical Reasoning Model,” arXiv preprint arXiv:2506.21734, 2025.
3. Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations,” in Proc. Int. Conf. Learning Representations (ICLR), 2020.
4. J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, “RoFormer: Enhanced Transformer with Rotary Position Embedding,” arXiv preprint arXiv:2104.09864, 2021.
5. N. Shazeer, “GLU Variants Improve Transformer,” arXiv preprint arXiv:2002.05202, 2020.
6. Selssabil, “Exploring SwiGLU: The Activation Function Powering Modern LLMs,” *Medium*, Oct. 4, 2024. [Online]. Available: https://medium.com/@s_boudefel/exploring-swiglu-the-activation-function-powering-modern-llms-9697f88221e7