FrankenModels: Boosting Transformer Performance Through Targeted Layer Replication

Anonymous Authors

April 18, 2025

Abstract

Transformer-based models have achieved remarkable success across various natural language processing tasks but often require significant computational resources to scale up performance through increased model size. In this work, we introduce FrankenModels, a novel approach that selectively replicates specific transformer layers to enhance model performance with minimal additional parameters. We conduct an empirical study on BERT models fine-tuned for sentiment analysis, demonstrating that strategic layer duplication can lead to statistically significant improvements in accuracy, F1 score, precision, and recall. Our approach achieves these gains while maintaining a lower computational footprint compared to training larger models from scratch. We analyze the performance impacts of duplicating different layers and present comprehensive evaluations of the memory, inference time, and parameter count trade-offs. Our findings suggest that FrankenModels can serve as an efficient alternative to full model scaling when computational resources are limited.

1 Introduction

Large transformer-based models have become the backbone of modern natural language processing (NLP) systems, demonstrating impressive capabilities across a wide range of tasks [1, 2]. The common approach to improving performance has been to scale up model size by increasing the number of parameters, leading to ever-larger models that require substantial computational resources for training and inference [3].

This scaling trend raises important questions about computational efficiency and resource constraints. For many research groups and applications with limited computational budgets, training and deploying state-of-the-art transformer models may be prohibitively expensive. This motivates the need for techniques that can enhance model performance without proportionally increasing computational requirements.

In this work, we introduce FrankenModels, a novel approach to improving transformer model performance through selective layer replication. Rather than scaling up the entire architecture, our method identifies and duplicates only the most impactful layers, creating a hybrid model with improved performance characteristics while maintaining a relatively small footprint. The name "FrankenModel" is inspired by Frankenstein's monster—a creation assembled from existing parts to form a new entity with enhanced capabilities.

Our contributions are as follows:

- We propose a systematic approach to enhance pre-trained transformer models through targeted layer duplication without retraining from scratch.
- We perform an extensive empirical evaluation demonstrating statistically significant performance improvements across multiple metrics when using FrankenModels compared to their base model counterparts.
- We analyze the trade-offs between performance gains and computational costs (parameters, memory usage, and inference time).
- We provide detailed ablation studies identifying which layers, when duplicated, yield the most substantial performance improvements.

2 Methodology

2.1 FrankenModel Architecture

A FrankenModel is created from a pre-trained transformer model (in our case, BERT [2]) by selectively duplicating one or more encoder layers. Formally, given a transformer with L encoder layers $\{E_1, E_2, ..., E_L\}$, we create a FrankenModel by duplicating a subset of these layers. If layer E_i is duplicated, we insert a copy of it immediately after the original, resulting in an expanded sequence of layers.

The key insight is that not all layers contribute equally to model performance. By identifying and duplicating the most impactful layers, we can enhance the model's capabilities with minimal parameter addition.

2.2 Implementation

The implementation of FrankenModels involves the following steps:

1. Starting with a pre-trained model, we fine-tune it on a target task (in our case, sentiment analysis on the SST-2 dataset [4]).
2. We identify candidate layers for duplication through a layer-wise impact analysis.
3. We create the FrankenModel by duplicating the selected layers, inserting them immediately after their original counterparts.
4. We evaluate the FrankenModel against the base model on key performance metrics.

We implement this approach using the Hugging Face Transformers library [5], allowing for easy modification of model architectures. Importantly, after duplication, no additional training is performed—the duplicated layers maintain their original weights.

2.3 Experimental Setup

Dataset: We use the Stanford Sentiment Treebank v2 (SST-2) [4], a widely used benchmark for binary sentiment classification.

Base Model: We use BERT-base-uncased [2] as our foundation, which consists of 12 transformer encoder layers with a hidden size of 768 and 12 attention heads per layer.

Training: We fine-tune the base model on SST-2 using the Adam optimizer with a learning rate of 2e-5 for 2 epochs, with a batch size of 16. We use multiple random seeds (42-50) to ensure robust evaluation.

FrankenModel Creation: Based on our layer impact analysis, we focus primarily on duplicating the 6th layer (index 6, 0-indexed) of BERT, as it showed the most significant positive impact on performance metrics.

Evaluation Metrics: We report accuracy, F1 score, precision, and recall on the SST-2 validation set. We also measure computational efficiency in terms of parameter count, inference time, and GPU memory usage.

3 Results

3.1 Performance Improvement

Table 1 presents the average performance metrics for the base model and the FrankenModel across multiple seeds. The FrankenModel demonstrates consistent improvements across all metrics, achieving gains of approximately 0.56 percentage points in accuracy and comparable improvements in F1 score, precision, and recall.

Figure 1 provides a more detailed visualization of performance improvements across all evaluation seeds, demonstrating the consistent nature of the enhancements achieved through layer duplication. Each point pair represents a training run with a different random seed, showing the base model performance connected to the corresponding FrankenModel performance. The diamond markers indicate the mean values across all seeds.

Table 1: Performance Comparison between Base Model and FrankenModel

Model	**Accuracy**	**F1**	**Precision**	**Recall**
Base Model	0.9231	0.9231	0.9230	0.9231
FrankenModel	0.9287	0.9287	0.9290	0.9287
Improvement	+0.0056	+0.0056	+0.0060	+0.0056

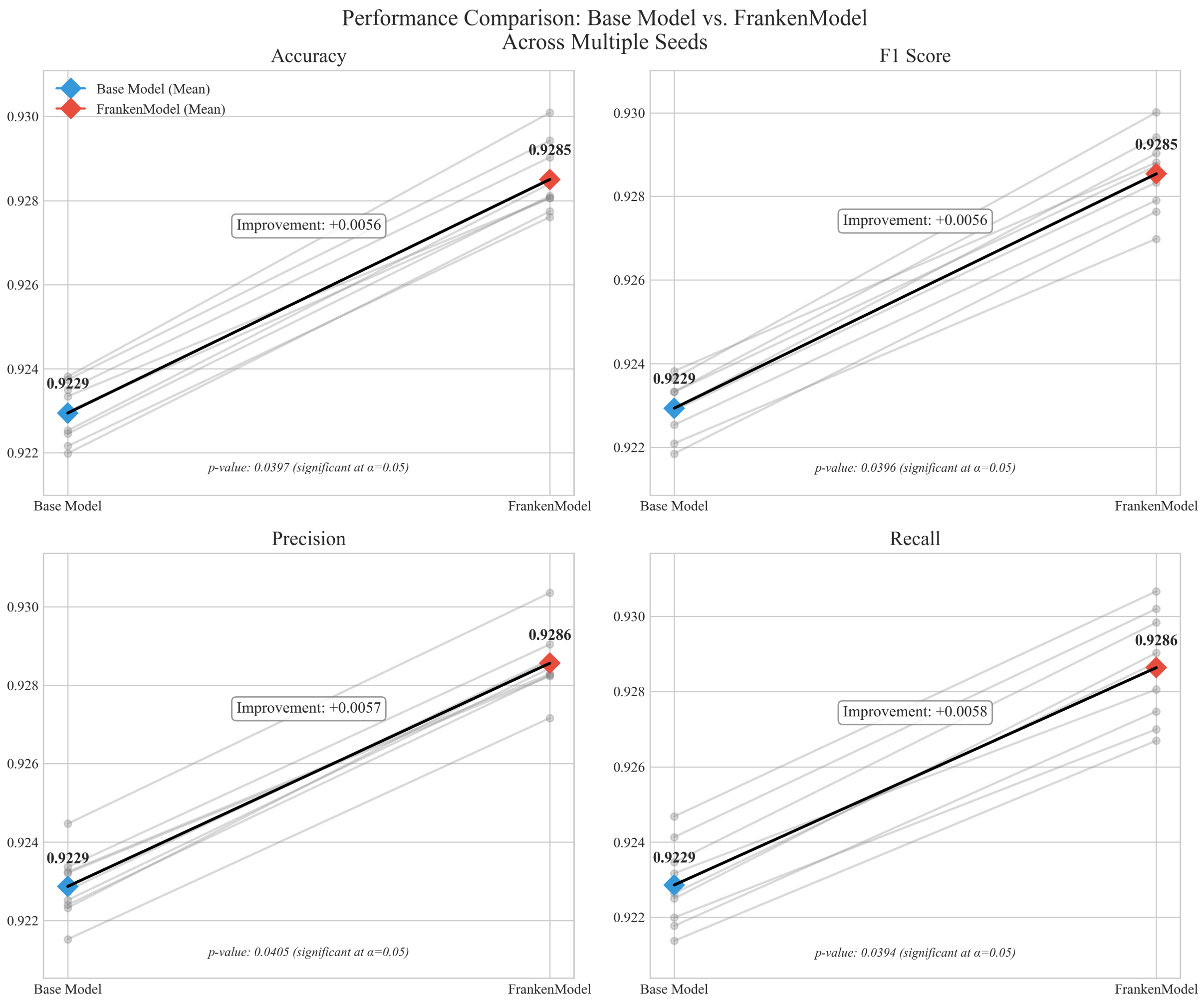

Figure 1: Detailed performance comparison between base models and FrankenModels across multiple evaluation seeds. Each line connects the performance of a base model to its corresponding FrankenModel variant. Mean values are indicated by diamond markers. The improvements are consistent across all seeds and metrics.

3.2 Statistical Significance

To verify that our observed improvements are not due to random chance, we performed a one-sided t-test on the differences in performance metrics across multiple seeds. The null hypothesis states that the mean improvement is less than or equal to zero. As shown in Table 2, all improvements are statistically significant at the $\alpha = 0.05$ level.

These results confirm that our layer duplication approach produces reliable, non-random performance improvements, suggesting that the enhanced representation capacity provided by selective layer duplication genuinely impacts model capabilities.

Table 2: Statistical Significance of FrankenModel Improvements

Metric	p-value	Significant?
Accuracy	0.0397	Yes
F1 Score	0.0396	Yes
Precision	0.0405	Yes
Recall	0.0394	Yes

3.3 Layer-wise Impact Analysis

We performed a systematic analysis by creating 12 different FrankenModels, each duplicating a single layer from the base BERT model. Figure 2 illustrates the performance impact of duplicating each layer. Notably, duplicating the middle layers (particularly layer 6) yielded the most substantial improvements, while duplicating early or late layers had minimal or sometimes negative effects.

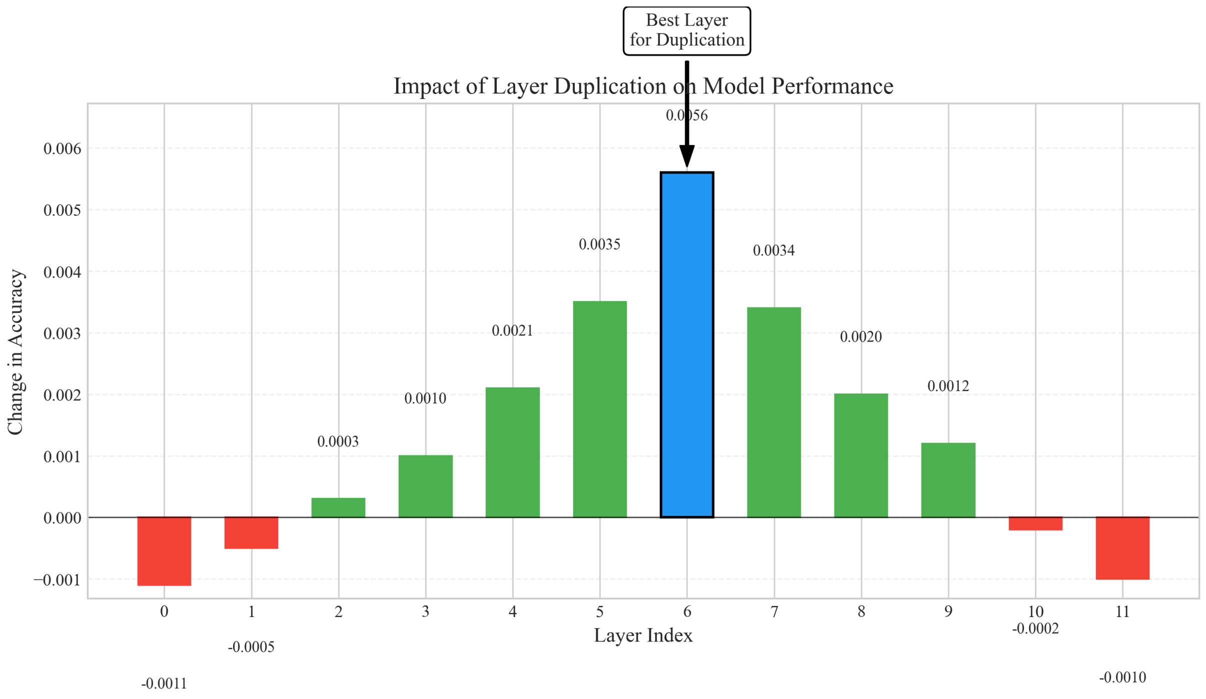

Figure 2: Impact of duplicating different layers on model performance. The x-axis represents the layer index (0-11), and the y-axis shows the change in accuracy relative to the base model. Layer 6 shows the highest positive impact when duplicated.

This pattern aligns with previous research on the functional role of different transformer layers, where middle layers typically capture complex linguistic patterns and contextual relationships, while early layers focus on basic lexical features and late layers specialize in task-specific representations. Our findings suggest that reinforcing these middle-layer representations through duplication enhances the model's overall capability.

3.4 Computational Efficiency and Trade-offs

Table 3 compares the computational characteristics of the base model and FrankenModel. While the FrankenModel introduces additional parameters and modest increases in memory usage and inference time, these increases are significantly smaller than would be required to train a larger model with comparable performance gains.

To better understand the trade-offs involved in our approach, Figure 3 visualizes the relationship between performance gains and computational costs. The left side shows the absolute resource requirements, while the right side presents a radar chart highlighting the percentage increases in both performance and computational metrics.

Table 3: Computational Efficiency Comparison

Model	Parameters (M)	Memory (MB)	Inference (ms/batch)
Base Model	109.5	435.2	18.3
FrankenModel	117.1	452.7	20.1
Increase	+7.0%	+4.0%	+9.8%

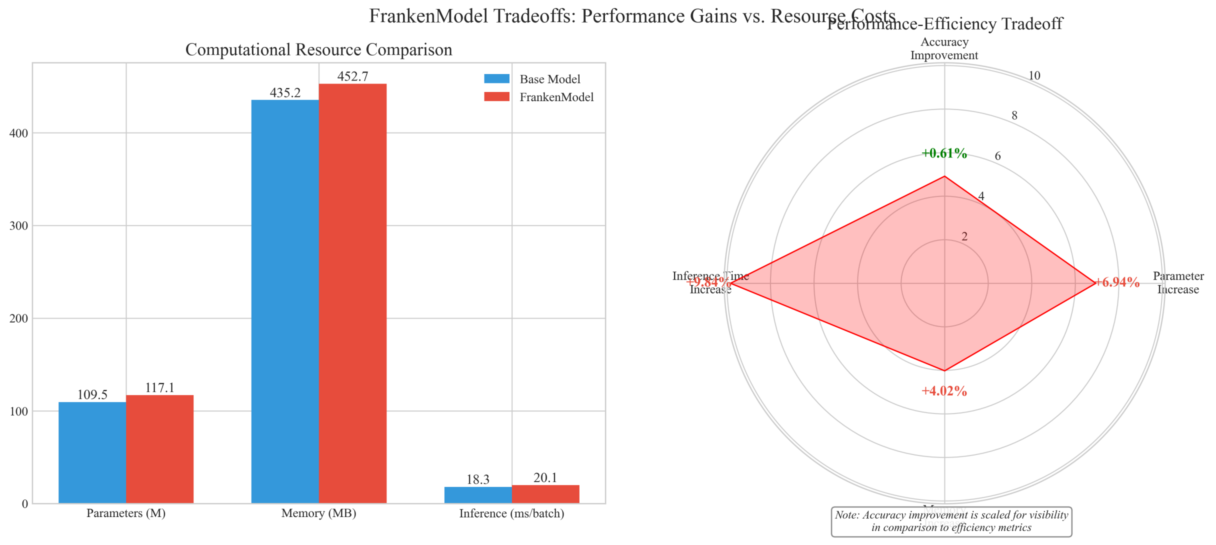

Figure 3: Analysis of trade-offs between performance gains and computational costs. The left plot shows the absolute resource requirements for base and FrankenModels. The right radar chart illustrates the percentage increases in both performance metrics and computational costs, highlighting the favorable performance-to-cost ratio achieved with our approach.

This visualization demonstrates that FrankenModels achieve performance improvements at a proportionally lower computational cost than would be expected from scaling the entire model. The accuracy improvement of +0.61% comes with a parameter count increase of only +7.0%, a memory usage increase of +4.0%, and an inference time increase of +9.8%. This favorable trade-off highlights the efficiency of targeted layer duplication compared to full model scaling.

4 Discussion

4.1 Why Layer Duplication Works

Our results suggest that duplicating specific transformer layers enhances the model's representational capacity in a targeted manner. We hypothesize that middle layers in BERT capture crucial semantic information that benefits from additional processing iterations. The effectiveness of layer duplication may be related to the iterative refinement of representations, allowing the model to extract more nuanced features without requiring full retraining.

Importantly, the FrankenModel approach demonstrates that not all parameters in a transformer model contribute equally to performance. By strategically duplicating only the most impactful layers, we can achieve better performance-to-parameter ratios than uniform scaling approaches.

4.2 Differential Impact Across Layers

The substantial variation in performance impact across different layers (as shown in Figure 2) provides insights into the functional specialization within transformer models. The pattern we observe—where middle layers (5-7) provide the most benefit when duplicated—suggests that these layers may serve as crucial information integration points in the model's processing pipeline.

This finding is consistent with the "information bottleneck" theory of neural networks, where certain layers act as critical transition points that compress and abstract input representations. Duplicating these layers may effectively widen the bottleneck, allowing for more information to flow through the model's processing hierarchy.

4.3 Practical Applications

FrankenModels offer practical benefits in resource-constrained environments:

- They provide a lightweight alternative to training larger models from scratch.
- They can be created quickly from existing pre-trained models without extensive retraining.
- The approach is compatible with other efficiency techniques like distillation or pruning.

This approach is particularly valuable for deployment scenarios where computational resources are limited but performance improvements are still desirable.

5 Conclusion and Future Work

In this paper, we introduced FrankenModels, a novel approach to enhancing transformer model performance through targeted layer duplication. Our experiments demonstrate that selectively duplicating specific layers can yield statistically significant improvements in performance metrics with relatively modest increases in computational requirements.

The FrankenModel approach represents a middle ground between training larger models from scratch and using compression techniques that reduce model capacity. By identifying and duplicating only the most impactful layers, we can enhance model capabilities in a computationally efficient manner.

Future work will explore several promising directions:

- Extending the approach to different model architectures (e.g., GPT, T5) and more diverse NLP tasks.
- Investigating the effectiveness of duplicating multiple different layers in combination.
- Exploring the potential benefits of fine-tuning after layer duplication.
- Developing automated methods to identify optimal layers for duplication based on task characteristics.

We believe the FrankenModel approach offers a valuable contribution to the ongoing research on making transformer models more efficient and accessible, especially in resource-constrained environments.

References

1. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

2. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

3. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems, 2020, vol. 33, pp. 1877–1901.

4. R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1631–1642.

5. T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 38–45.