Improving Transformer Models by Reordering their Sublayers

Ofir Press[◇] **Noah A. Smith**[◇♠] **Omer Levy**[♣]

[◇] University of Washington
[♠] Allen Institute for AI
[♣] Facebook AI Research

Abstract

Multilayer transformer networks consist of interleaved self-attention and feedforward sublayers. Could ordering the sublayers in a different pattern achieve better performance? We generate randomly ordered transformers and train them with the language modeling objective. We observe that some of these models are able to achieve better performance than the interleaved baseline, and that those successful variants tend to have more self-attention at the bottom and more feedforward sublayers at the top. We propose a new transformer design pattern that adheres to this property, the *sandwich transformer*, and show that it improves perplexity on the WikiText-103 language modeling benchmark, at no cost in parameters, memory, or training time.

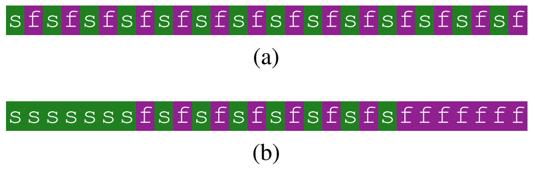


Figure 1: A transformer network (a) is composed of interleaved self-attention (green) and feedforward (purple) sublayers. Our sandwich transformer (b), a reordering of the transformer sublayers, performs better on language modeling.

1 Introduction

The transformer layer (Vaswani et al., 2017) is currently the primary modeling component in natural language processing, playing a lead role in recent innovations such as BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019). Each transformer layer consists of a *self-attention* sublayer (s) followed by a *feedforward* sublayer (f), creating an interleaving pattern of self-attention and feedforward sublayers (sfsfsf · · ·) throughout a multilayer transformer network. To the best of our knowledge, there is no a priori reason to expect this particular pattern to be optimal. We conduct a series of explorations to obtain insights about the nature of transformer orderings that work well, and based on this, we design a new transformer ordering pattern that improves upon the baseline.

First, we generate random transformer models, varying the number of each type of sublayer, and their ordering, while keeping the number of parameters constant. We train these models on the standard WikiText-103 language modeling benchmark (Merity et al., 2016), and observe that some of these random models outperform the original interleaved transformer network, even when the number of self-attention and feedforward layers is not equal. Our analysis shows that models with more self-attention toward the bottom and more feedforward sublayers toward the top tend to perform better in general.

Based on this insight, we design a new family of transformer models that follow a distinct sublayer ordering pattern: *sandwich transformers* (Figure 1). Our experiments demonstrate that a sandwich transformer outperforms the baseline (of Baevski and Auli, 2019) by 0.44 perplexity. This result is made more interesting by the fact that our sandwich transformer is simply a reordering of the sublayers in the baseline model, and does not require more parameters, memory, or training time.

2 Notation

Each transformer layer consists of a self-attention sublayer followed by a feedforward sublayer, modifying a sequence of vectors $\mathbf{X}_0$ as follows:[1]

$$\mathbf{X}_1 = \text{self-attention}(\mathbf{X}_0) + \mathbf{X}_0$$
$$\mathbf{X}_2 = \text{feedforward}(\mathbf{X}_1) + \mathbf{X}_1$$

Stacking multiple transformer layers creates an interleaved network of sublayers. We denote these models as strings, with s and f representing self-attention and feedforward sublayers, respectively. A three-layer transformer network, for example, would be denoted sfsfsf, with the flow of computation moving from input on the left to output on the right. Thus, any string in the regular language (s|f)* defines a valid network that uses the same building blocks as the original transformer. For simplicity, we refer to these alternatives as transformers as well.

3 Random Search

We conduct a series of experiments to understand which transformer networks work well and whether particular architectural patterns can improve performance. First, we generate random transformer models while keeping constant the number of parameters. We then train these random models to determine whether the interleaving pattern (sfsfsf$\cdots$) is optimal (Section 3.1), and whether balancing the number of self-attention and feedforward sublayers is desirable (Section 3.2). Finally, we analyze additional properties of these random models, and find that those with more self-attention at the beginning and more feedforward sublayers near the end tend to outperform the standard interleaved model (Section 3.3).

[1] We omit dropout (Srivastava et al., 2014) and layer normalization (Ba et al., 2016) to simplify the notation.

Experimental Setup Our baseline is the strong transformer language model of Baevski and Auli (2019), trained on WikiText-103 (Merity et al., 2016).[2] This model contains 16 transformer layers of $d = 1024$ dimensions, with 16 heads in each self-attention sublayer, and feedforward sublayers with an inner dimension of 4096. In this setting, each self-attention sublayer contains $4d^2$ parameters, while each feedforward sublayer contains $8d^2$ parameters (excluding bias terms, which have a marginal contribution). Thus, each f sublayer contains twice the parameters of a s sublayer, following the parameter ratio between self-attention and feedforward sublayers described in Vaswani et al. (2017).

All of our experiments use the same hyperparameters as Baevski and Auli's original model. To set an accurate baseline, we train the baseline model (the standard interleaved transformer stack) with five different random seeds, achieving 18.65 ± 0.24 perplexity on the development set. Unless otherwise mentioned, we do not modify the random seed in the other experiments.

[2] WikiText-103 contains roughly 103 million tokens from English Wikipedia, split into train, development, and test sets by article.

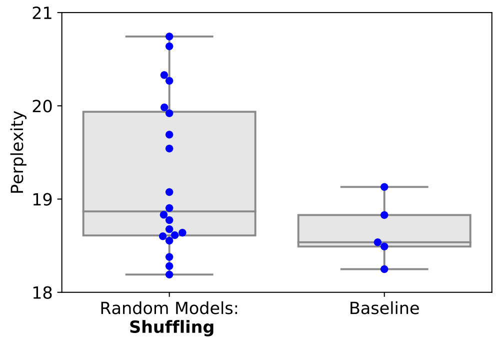


Figure 2: The perplexities on the WikiText-103 development set of 20 randomly generated models with 16 self-attention and 16 feedforward sublayers and of the 5 baselines (the standard transformer trained with different random seeds). The best random model (`sfsfssfsssffsfsfsfsffffssffsfssf`) obtains 18.19 perplexity.

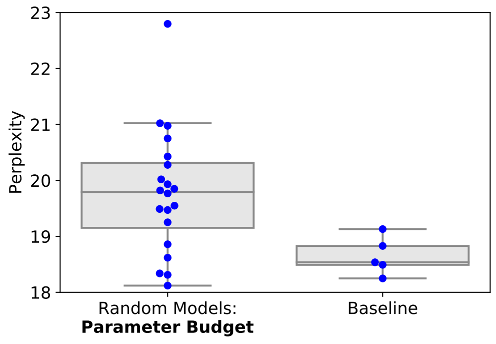


Figure 3: The perplexities on the WikiText-103 development set of 20 randomly generated models with the same number of parameters as the baseline, and of the 5 baselines (the standard transformer trained with different random seeds). The best random model (`ssssssfsssffffsfsfffffffffffsf`) obtains 18.12 perplexity.

3.1 Is Interleaving Optimal?

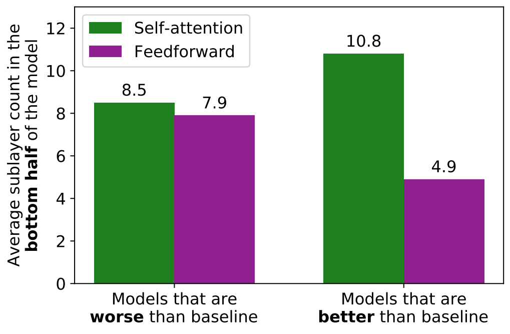
In the baseline 16-layer transformer model, 16 sublayers of each type are interleaved. Can we improve model performance by simply rearranging them? We thus generate 20 random transformer models with 16 self-attention sublayers and 16 feedforward sublayers, randomly permuted, and train these models from scratch, without modifying any of the hyperparameters.

Figure 2 shows that 7 of the 20 randomly-permuted models perform at least as well as the interleaved baseline's average performance, with the best model achieving 18.19 perplexity (full results are in Table 2 in the appendix). While the average performance of the baseline model beats the average performance of these random models, the fact that a third of our random models outperformed the average baseline suggests that a better ordering than interleaving probably exists.

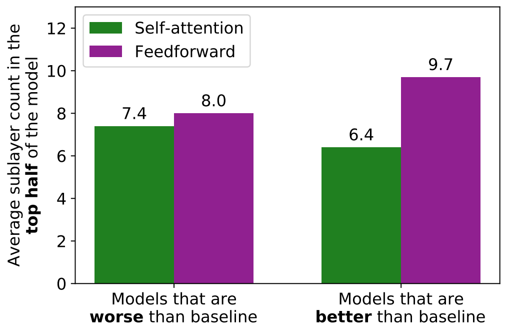
3.2 Are Balanced Stacks Better?

Is it necessary to have an identical number of sublayers of each type, or could models with more self-attention (or more feedforward) sublayers yield better results? To find out, we generate 20 unbalanced transformer models by randomly selecting one sublayer at a time (either `s` or `f` with equal probability) until the parameter budget is exhausted. Since a feedforward sublayer contains double the parameters of a self-attention sublayer, the networks' depth is not necessarily 32 sublayers as before and can range from 24 (all `f`) to 48 (all `s`).

Figure 3 shows that four of the generated unbalanced models outperform the average baseline transformer (full results are in Table 3 in the appendix). The best performing random model reaches a perplexity of 18.12 and has 12 self-attention and 18 feedforward sublayers. Both the average and the median perplexities of this sample of unbalanced models are worse than those of the balanced permuted models in Section 3.1. We do not observe any preference for more sublayers of one type over the other; there are self-attention-heavy and feedforward-heavy models in both the top five and the bottom five of the results table. While offering



(a)



(b)

Figure 4: Analysis of sublayer distribution in models that do better or worse than the average baseline, split across bottom (a) and top (b) halves of the model.

no guarantees – given the small sample sizes and fixed hyperparameters – we take from the above explorations that balancing the number of self-attention and feedforward sublayers appears to be a desirable property, though not a necessary one.

3.3 Attention First, Feedforward Later

So far, it is not clear which characteristics make one transformer model more successful than another; for example, measuring the number of times each sublayer type appears in the network does not reveal any strong correlation with performance. However, analyzing the bottom (or top) half of the network in isolation reveals an interesting property.

We first split the models to those that perform better than the average baseline and those that do not. We then slice each one of the previously-generated random models in half by parameter count (e.g., `sssssff` would be split to `ssss` and `ff`, since every `f` contains twice as many parameters as an `s`), and count how many sublayers of each type appear in each slice.

Figure 4 shows that models that outperform the average baseline tend to have more self-attention `s` in the first (bottom) half of the network and more `f` in the second (top) half. While we do not have a good hypothesis to explain this phenomenon, we can, however, exploit it to improve transformers (Section 4).

4 Designing a Better Transformer

Our analysis in the previous section motivates designing a transformer model that is heavy on self-attention at the bottom and feedforward sublayers at the top, while at the same time containing a more-or-less balanced amount of both sublayer types. As a first attempt to manually design a better transformer, we take this hypothesis to the extreme, and train a transformer model of 16 self-attention sublayers followed by 16 feedforward sublayers ($\texttt{s}^{16}\texttt{f}^{16}$). This model achieves 18.82 perplexity, which is comparable to the performance of the baseline with the same number of parameters.

We next generalize this model and the original interleaved transformer, creating the family of *sandwich transformers*. A sandwich$_k^n$ transformer consists of $2n$ sublayers in total (n of each type), conforming to the regular expression $\texttt{s}^k(\texttt{sf})^{n-k}\texttt{f}^k$. The first k sublayers are purely self-attention (`s`), while the last k are feedforward sublayers (`f`). In between, we use the original interleaving pattern (sf) to fill the remaining $2(n-k)$ sublayers. When $k=0$, we get the original transformer stack, and when $k=n-1$ (its maximal value) we get the previously mentioned $s^n f^n$ model. We refer to k as the transformer's *sandwich coefficient*.

We train sandwich transformers for $n=16$ (to remain within the same parameter budget as our baseline language model) and all values of $k \in \{0, \dots, 15\}$. Figure 5 shows the transformer's performance as a function of the sandwich coefficient k. With the exception of $k=14, 15$, all sandwich transformers achieve lower perplexities than the average baseline transformer. Of those, 6 models outperform the best baseline transformer ($k=5, 6, 8, 9, 10, 11$). The best performance of 17.84 perplexity is obtained when $k=6$.

We then take our best model and compare it to the best baseline (selected via development set perplexity) on WikiText-103's test set and find that our sandwich transformer indeed outperforms the original transformer by 0.44 perplexity (Table 1). To check whether this advantage is consistent, we train 4 more sandwich$_6^{16}$ models with different random seeds (5 in total) and evaluate them on the development set (to avoid running more than once on the test set). Figure 6 compares the distribution of sandwich transformer perplexities to the baseline's; we obtain a mean perplexity value of 17.98 with a standard deviation of 0.10, while the baseline achieves 18.65 ± 0.24 perplexity.

Despite its simple and even heuristic design, the sandwich transformer consistently outperforms the standard interleaved transformer. This improvement comes at no extra cost in parameters, data, memory, or computation.

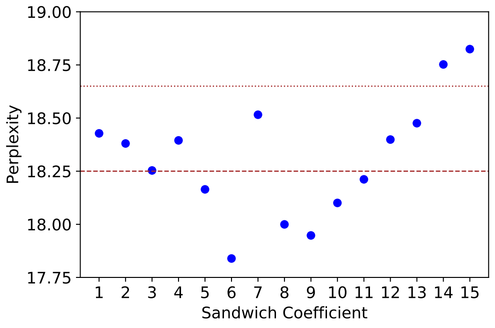


Figure 5: Sandwich model's sandwich coefficient (k) and perplexity, for $k \in \{1, \dots, 15\}$. The dotted line is the average baseline model's perplexity (trained with different random seeds), and the dashed line is the best baseline model's perplexity.

Model	**Test**
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	18.40
sssssssfsfsfsfsfsfsfsfsfsfffffff	17.96

Table 1: The sandwich$_6^{16}$ transformer achieves better language modeling perplexity than the unmodified, interleaved transformer (sandwich$_0^{16}$) on the WikiText-103 test set.

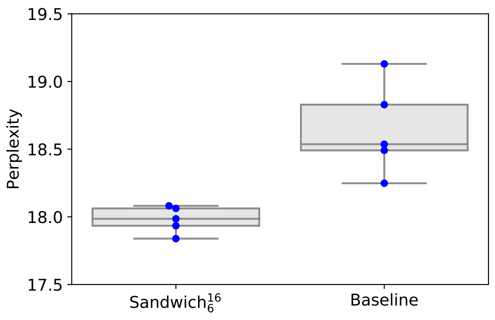


Figure 6: The perplexities on the WikiText-103 development set of the sandwich$_6^{16}$ transformer and the baseline. Each model is trained with 5 different random seeds to assess the perplexity distribution.

5 Related Work

Neural Architecture Search In this paper, we manually searched through a constrained transformer architecture space, after analyzing the results of small-scale random searches. This human-in-the-loop for architecture method has advantages over previous methods (Jozefowicz et al., 2015; Zoph and Le, 2016; Tan and Le, 2019) since it requires that only a few dozen models be trained, unlike typical architecture search methods that require training thousands, consuming massive computational resources.

While we do find a better performing transformer, our goal is not only to do so, but to better understand how sublayer ordering affects transformer models. Future work could apply methods from the architecture space literature to the sublayer ordering problem. Furthermore, a better understanding of the inner workings of transformers could inspire more efficient, constrained architecture search.

Transformer Modifications Unlike recent papers that tried to improve the transformer by modifying the sublayers such as So et al. (2019); Guo et al. (2019); Zhang et al. (2019); Correia et al. (2019), in this paper we do not modify the sublayers at all, but simply rearrange their order. The performance gains from sublayer reordering are orthogonal to improving the sublayers themselves and could be combined to achieve even better performance.

6 Conclusion

We train random transformer models with reordered sublayers, and find that some perform better than the baseline interleaved transformer in language modeling. We observe that, on average, better models contain more self-attention sublayers at the bottom and more feedforward sublayer at the top. This leads us to design a new transformer stack, the sandwich transformer, which consistently improves performance over the baseline at no cost.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*.

Gonalo M. Correia, Vlad Niculae, and Andr F. T. Martins. 2019. Adaptively sparse transformers.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. 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)*, pages 1315–1325, Minneapolis, Minnesota. Association for Computational Linguistics.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI*.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5877–5886, Long Beach, California, USA. PMLR.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114, Long Beach, California, USA. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. Improving deep transformer with depth-scaled initialization and merged attention.

Barret Zoph and Quoc V. Le. 2016. Neural architecture search with reinforcement learning.

A Appendix

This section contains detailed results of the experiments described in Section 3. Table 2 shows the performance of permuted but balanced transformers (Section 3.1). Table 3 shows the performance of unbalanced transformers (Section 3.2).

Model	PPL
fsfsfffsffsfsssffsfssfssssffsffs	20.74
sfssffsffffssssfsfffsfsffsfssssf	20.64
fsffssffssssffsssssffsfssfsfffff	20.33
fsffffffsssfssffsfssffsfsssffsss	20.27
fssffffffsfsssfffssssfffssssffss	19.98
sssfssfsffffssfsfsfsssffsfsfffsf	19.92
fffsfsssfsffsfsffsffsssssffssffs	19.69
fffsffssffsssfssfsssfffffsfsssfs	19.54
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**19.13**
fsffssfssfffssssfffsssffffsfssfs	19.08
sfsffssssffssffffsssffsssfsffsff	18.90
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.83**
sssssssffsffsfsfsffffsfffsfssffs	18.83
sffsfsffsfsssffssfssssssfffffffs	18.77
sssfssffsfssfsffsfffssffsfsffssf	18.68
fffsssssfffsfssssffsfsfsfssffsff	18.64
sfffsssfsfssfsssssfssfffffsfffsf	18.61
ssffssfssssffffffssffsssfsffssff	18.60
fsfsssssfsfsfffffsfffsffssffssss	18.55
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.54**
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.49**
fsfsssssfsfffssfsffsfsfsfsffffss	18.38
sfssffsfsfsffsssssfffsssfffsffsf	18.28
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.25**
sfsfssfsssffsfsfsfsffffssffsfssf	18.19

Table 2: Randomly generated models with 16 self-attention (s) and 16 feedforward (f) sublayers, and their perplexity on the WikiText-103 development set. The baselines (the standard transformer trained with different random seeds) are in bold.

Model	PPL
sfffssfsfsfssffffsfsffsffffff	22.80
sffssfsssssssssssssfsfsssfsffsssfsssfs	21.02
ssssssffsffffssfffffsssfsfsssssssss	20.98
fffffffffsffssffsffssssfsfsssf	20.75
fssfsssffffffssfsssfsfffssssfsfss	20.43
sffsffffffsfsfssfsssfsfsfssfssfs	20.28
sffssffsfffsfsfssssffffffssssff	20.02
fsffsfssffffsfsfffsfffssfffsss	19.93
sffsffssffsfsffsssfsssssfsssfffsss	19.85
ssfffffffssfffssfssffsfsfsffsf	19.82
sfsfsfffsfffssfsfffsffssfsfsfss	19.77
sfsffsssffsffsssfssfffffssssfsssf	19.55
sffsfssfffsffsfssssfsfsffffsfsss	19.49
sffffsffssssfsssfssfffsssfssssfsfs	19.47
fsssffssssssfsfsfsffsffffssfsfssss	19.25
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**19.13**
fssssssfsfsfsfffsfsssfssffssssfsff	18.86
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.83**
ssfsfsssfsssssffsfsfsssfssfsfsssssssf	18.62
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.54**
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.49**
sssfsffsfssfsssffsffffffssfsfff	18.34
sssfsfsffsssfsfffffsfsffffsssff	18.31
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.25**
ssssssfsssffffsfsfffffffffffsf	18.12

Table 3: Randomly generated models with the same number of parameters as the baseline, and their perplexity on the WikiText-103 development set. The baselines (the standard transformer trained with different random seeds) are in bold.