Improving Transformer Models by Reordering their Sublayers

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

[◇]Paul G. Allen School of Computer Science & Engineering, 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 pattern that adheres to this property, the *sandwich transformer*, and show that it improves perplexity on multiple word-level and character-level language modeling benchmarks, at no cost in parameters, memory, or training time. However, the sandwich reordering pattern does not guarantee performance gains across every task, as we demonstrate on machine translation models. Instead, we suggest that further exploration of task-specific sublayer reorderings is needed in order to unlock additional gains.[1]

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$\cdots$) throughout a multilayer transformer model. To the best of our knowledge, there is no 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.

(a) Interleaved Transformer

(b) Sandwich Transformer

Figure 1: A transformer model (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. Input flows from left to right.

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 word-level language modeling benchmark (Merity et al., 2016), and observe that some of these random models outperform the original interleaved transformer model, 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). 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.

Finally, we demonstrate that even though the

[1]Our code is available at https://github.com/ofirpress/sandwich_transformer.

Model	PPL
fsfsfffsffsfsssffsfssfsssssffsffs	20.74
sfssffsffffssssfsfffsfsffsfsssssf	20.64
fsffssffsssssffssssssffsfssfsfffff	20.33
fsffffffsssfssffsfssffsfssffsfsssffsss	20.27
fssffffffsfsssfffsssssfffsssssffss	19.98
sssfssfsfffffssfsfsfsssffsfsfffsf	19.92
fffsfsssfsffsfsffsffsssssffssffs	19.69
fffsffssffsssfssfsssfffffsfsssfs	19.54
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**19.13**
fsffssfssfffsssssfffsssfffffsfssfs	19.08
sfsffsssssffssffffsssffsssfsffsff	18.90
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.83**
sssssssffsffsfsfsffffsfffsfsssffs	18.83
sffsfsffsfsssffssfsssssssffffffs	18.77
sssfssffsfssfsffsfffssffsfsffssf	18.68
fffsssssfffsfsssssffsfsfsfssffsff	18.64
sfffsssfsfssfsssssfssfffffsfffsf	18.61
ssffssfsssssffffffssffsssfsffssff	18.60
fsfsssssfsfsffffffsfffsffssffssss	18.55
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.54**
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.49**
fsfsssssfsffffssfsffsfsfsfsfffffss	18.38
sfssffsfsfsffsssssffffsssffffsffsf	18.28
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.25**
sfsfssfsssffsfsfsfsffffssffsfssf	18.19

Table 1: Randomly generated models with 16 self-attention (s) sublayers 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.

sandwich transformer is motivated by random search experiments on WikiText-103, it can improve performance on additional domains and tasks. Sandwich transformers achieve state-of-the-art results on the enwik8 character-level language modeling dataset and on an additional word-level corpus, but have no significant effect on machine translation. We conjecture that tuning transformer reorderings to specific tasks could yield even larger gains, and that further exploration of the ordering space may provide universally beneficial patterns.

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:[2]

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

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 $(\mathtt{s}|\mathtt{f})^*$ 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.

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

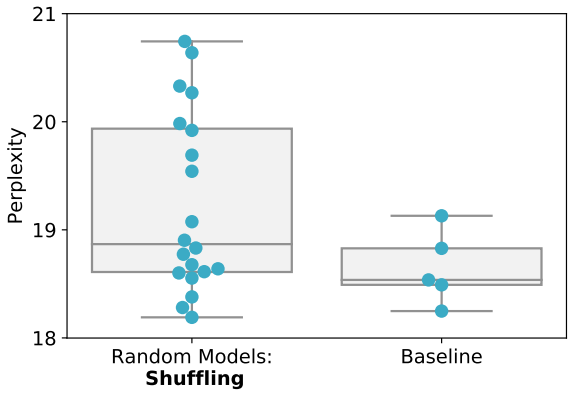


Figure 2: The perplexities on the WikiText-103 development set of 20 randomly generated models with 16 self-attention and 16 feedforward sublayers and of the 5 baselines (the standard transformer trained with different random seeds).

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 the number of parameters constant. 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).

Experimental Setup Our baseline is the strong transformer language model of Baevski and Auli (2019), trained on WikiText-103 (Merity et al., 2016). WikiText-103 contains roughly 103 million tokens from English Wikipedia, split into train, development, and test sets by article. The Baevski

Model	PPL
sfffssfsfsfssffffsfsffsffffff	22.80
sffssfsssssssssssssfsfsssfsffsssfsssfs	21.02
ssssssffsffffssfffffsssfsfsssssssss	20.98
fffffffffsffssffsffsssfsfsssf	20.75
fssfsssffffffssfsssfsfffsssfsfss	20.43
sffsffffffsfsfssfsssfsfsfssfssfs	20.28
sffssffsfffsfsfsssssffffffsssssff	20.02
fsffsfssffffsfsfffsfffssfffsss	19.93
sffsffssffsfsffsssfsssssfsssfffsss	19.85
ssffffffffssfffssfssffsfsfsffsf	19.82
sfsfsfffsfffssfsfffsffssfsfsfss	19.77
sfsffsssffsffsssfssfffffsssssfsssf	19.55
sffsfssffffsffsfsssssfsfsfffffsfsss	19.49
sfffffsffsssssfsssfssfffsssfsssssfsfs	19.47
fsssffsssssssfsfsfsffsfffffssfsfsssss	19.25
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**19.13**
fsssssssfsfsfsfffsfsssfssffsssssfsff	18.86
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.83**
ssfsfsssfsssssffsfsfsssfssfsfsssssssf	18.62
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.54**
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.49**
sssfsffsfssfsssffsfffffffssfsfff	18.34
sssfsfsffsssfsfffffsfsffffsssff	18.31
sfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsf	**18.25**
ssssssfsssffffsfsfffffffffffsf	18.12

Table 2: 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.

and Auli 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) with five different random seeds, achieving 18.65 ± 0.24 perplexity on the development set.

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. Table 1 shows the entire sample, while Figure 2 plots the perplexity distributions of the shuffled transformers and the baseline side by side.

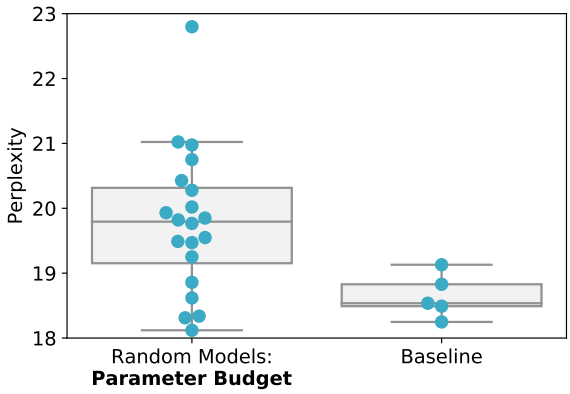


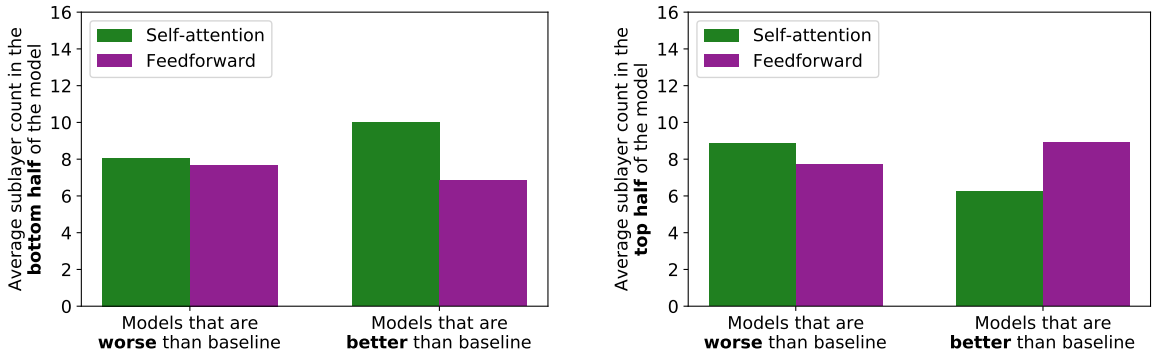
Figure 3: The perplexities on the WikiText-103 development set of 20 randomly generated models with the same number of parameters as the baseline, and of the 5 baselines (the standard transformer trained with different random seeds).

We observe 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. 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 Architectures 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). Table 2 shows the entire sample, while Figure 3 plots the perplexity distributions of the randomly-generated transformers and the baseline side by side.

We see that four of the generated unbalanced models outperform the average baseline transformer. The best performing random model reaches



(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.

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 (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 no guarantees – given the small sample sizes and fixed hyperparameters – we conclude that a balanced number of self-attention and feedforward sublayers seems 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., sssfff 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 *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 ($s^{16}f^{16}$). 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^n_k transformer consists of $2n$ sublayers in total (n of each type), conforming to the regular expression $s^k(sf)^{n-k}f^k$. 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 model, and when $k = n - 1$ (its maximal value) we get the previously mentioned s^nf^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 $k \in \{0, \ldots, 15\}$. 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

Model	Test
Baseline (Baevski and Auli, 2019)	18.70
Transformer XL (Dai et al., 2019)	18.30
kNN-LM (Khandelwal et al., 2019)	15.79
Baseline (5 Runs)	18.63 ± 0.26
Sandwich$_6^{16}$	17.96

Table 3: Performance on the WikiText-103 test set. We compare the best sandwich transformer to the unmodified, interleaved transformer baseline (Baevski and Auli, 2019) trained over 5 random seeds and to other previously reported results.

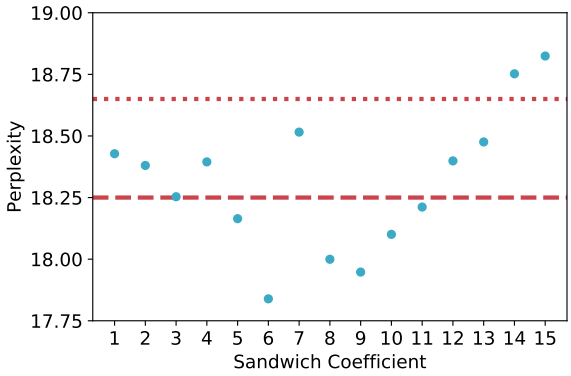


Figure 5: The transformer's sandwich coefficient (k) and validation perplexity, for $k \in \{1, \ldots, 15\}$. The dotted line is the average baseline model's perplexity (trained with different random seeds), whereas the dashed line represents the best baseline model.

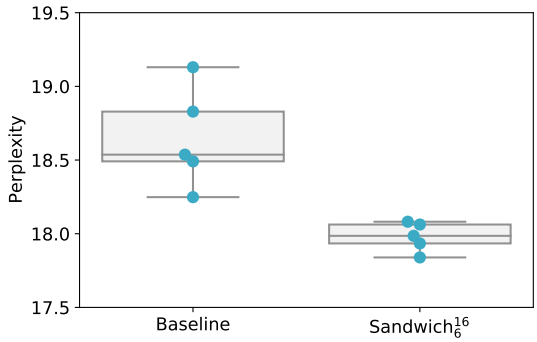


Figure 6: Performance on the WikiText-103 development set of the Sandwich$_6^{16}$ transformer and the baseline. Each model is trained with 5 different random seeds to assess the perplexity distribution.

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 compare this model to the baseline on WikiText-103's test set.

Table 3 shows that, despite its simple design, the sandwich transformer outperforms the original transformer baseline by roughly double the gap between the baseline (Baevski and Auli, 2019) and Transformer XL (Dai et al., 2019). This improvement comes at no extra cost in parameters, data, memory, or computation; we did not even change any of the original hyperparameters, including the number of training epochs.

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 evaluating our model more than once on the test set. This is the only experiment in which we modify our model's random seed. Figure 6 shows that we obtain a mean perplexity value of 17.98 with a standard deviation of 0.10, while the baseline achieves 18.65 mean perplexity, with a larger standard deviation of 0.34 (these values reflect *development* set performance, not test set performance as in Table 3).

In very recent work, kNN-LM (Khandelwal et al., 2019) set a new state of the art on WikiText-103, surpassing other recent models by a wide margin. The model achieves this result by storing the entire training set in an auxiliary memory component. Since this approach appears orthogonal to ours, it is quite possible that kNN-LM could benefit from sublayer reordering as well.

5 One Reordering to Rule Them All?

The sandwich transformer is a manually-crafted pattern motivated by the performance of random sublayer reorderings of the Baevski and Auli (2019) model, trained on the WikiText-103 word-level language modeling benchmark (Merity et al., 2016).

Does this particular pattern improve performance in other settings as well? To find out, we apply sandwich transformers to three other tasks: word-level language modeling on a different domain (Section 5.1), character-level language modeling (Section 5.2), and machine translation (Section 5.3).

Results show that as we drift away from our original setting, sandwich transformers provide diminishing gains, but always perform at least as well as the baseline transformers (provided that the sandwich coefficient is properly tuned). This finding suggests that different settings may benefit from different sublayer reordering patterns.

Model	PPL
Baseline (5 runs)	11.89 ± 0.35
kNN-LM (Khandelwal et al., 2019)	10.89
Sandwich$^{16}_{7}$	10.83

Table 4: Performance on the Toronto Books Corpus language modeling test set. The baseline model (Baevski and Auli, 2019) is trained over 5 random seeds. The sandwich coefficient is tuned on the validation set and we run our model on the test set only once.

5.1 Books-Domain Language Modeling

We first apply sandwich transformers to a different domain, while retaining the other architectural aspects and hyperparameter settings from Baevski and Auli (2019). Specifically, we use the Toronto Books Corpus (Zhu et al., 2015), which has previously been used to train GPT (Radford et al., 2018) and also BERT (Devlin et al., 2019) (combined with Wikipedia). The corpus contains roughly 700M tokens.

We use the same train/validation/test split as Khandelwal et al. (2019), as well as their tokenization, which uses BERT's vocabulary of 29K byte-pair encodings. Since the vocabulary is much smaller than WikiText-103's, we replace the adaptive word embedding and softmax of Baevski and Auli (2019) with a tied word embedding and softmax matrix (Press and Wolf, 2017; Inan et al., 2017). Finally, we tune the sandwich coefficient on the development set for $k \in \{4, \ldots, 8\}$, i.e., a neighborhood of 2 around the best value we found for WikiText-103 ($k = 6$).

Table 4 shows that the sandwich transformer transfers well to the books domain, improving performance by 1.06 perplexity, achieving similar performance to the datastore-augmented kNN-LM (Khandelwal et al., 2019), which is the state of the art on WikiText-103 (see Section 4).

5.2 Character-level Language Modeling

Modeling text as a stream of characters, rather than word or subword tokens, presents a different modeling challenge: long-range dependencies become critical, and the vocabulary takes on a more uniform distribution. We apply our sandwich reordering to the adaptive span model of Sukhbaatar et al. (2019), which is state of the art on the popular English-language benchmark text8 and is currently a close second on enwik8.[3] The adaptive span model learns to control each attention head's maximal attention span, freeing up memory in the bottom layers (which typically need very short attention spans) and applying it to the top layers, allowing the top-level attention heads to reach significantly longer distances. The adaptive span model's efficient use of attention also results in a significant speed boost.

We tune the sandwich coefficient on the development set for $k \in \{1, \ldots, 8\}$ (the baseline model has 24 transformer layers). We do not modify any hyperparameters, including the number of training epochs. Table 5 compares the baseline model's performance with the sandwich transformer's. On text8, the sandwich transformer performs within the baseline's random seed variance. On enwik8, the sandwich transformer gains an improvement of about 0.007 bits-per-character, matching the state of the art results obtained by the Transformer-XL-based compressive transformer of Rae et al. (2020).

However, our approach is able to achieve this result without applying the Transformer-XL's recurrent attention, which is much slower (Sukhbaatar et al., 2019), and without adding additional parameters (the compressive transformer uses 277M parameters, while our baseline and sandwich models use only 209M).

5.3 Machine Translation

Sandwich Decoders Tranformer-based translation models (Vaswani et al., 2017) consist of an encoder and decoder, where the encoder has interleaved self-attention and feedforward sublayers (just as in language models), while the decoder includes an additional sublayer, cross-attention (c), between every pair of self-attention and feedforward sublayers. Cross-attention sublayers attend to the encoder's representations of the input sentence's tokens.

Following our notation from Section 2, a transformer decoder layer modifies the sequence of tokens in the target language $\mathbf{Y_0}$, using the encoded source tokens $\mathbf{X}$, as follows:

$$\mathbf{Y}_1 = \text{self-attention}(\mathbf{Y}_0) + \mathbf{Y}_0$$
$$\mathbf{Y}_2 = \text{cross-attention}(\mathbf{Y}_1, \mathbf{X}) + \mathbf{Y}_1$$
$$\mathbf{Y}_3 = \text{feedforward}(\mathbf{Y}_2) + \mathbf{Y}_2$$

Applying the sandwich pattern to the encoder follows the same methodology as our previous experiments. However, for the decoder, we group the

[3] Both datasets are taken from http://mattmahoney.net/dc/textdata.html

Model	text8 (BPC)	enwik8 (BPC)
Transformer-XL (Dai et al., 2019)	1.08	0.99
Adaptive Span (Sukhbaatar et al., 2019)	1.07	0.98
Compressive (Rae et al., 2020)	—	0.97
Baseline (Adaptive Span; 5 Runs)	1.0802 ± 0.0103	0.9752 ± 0.0008
Sandwich$^{24}_{3}$	1.076	—
Sandwich$^{24}_{5}$	—	0.968

Table 5: Performance on character-level language modeling, evaluated on the enwik8 and text8 test sets. The baseline model (Sukhbaatar et al., 2019) is trained over 5 random seeds. The sandwich coefficient is tuned on each benchmark's validation set, and we run our model on the test only once.

self-attention (s) and cross-attention (c) sublayers, and treat them as a single unit for reordering purposes (sc). For example, a three layer decoder (scfscfscf) with a sandwiching coefficient of $k = 1$ would be: scscfscff. We apply the sandwich pattern to either the encoder or decoder separately, while keeping the other stack in its original interleaved pattern.

Experiment Setting As a baseline, we use the large transformer model (6 encoder/decoder layers, embedding size of 1024, feedforward inner dimension of 4096, and 16 attention heads) with the hyperparameters of Ott et al. (2018). We also follow their setup for training and evaluation: we train on the WMT 2014 En-De dataset which contains 4.5M sentence pairs; we validate on newstest13 and test on newstest14. We use a vocabulary of 32K symbols based on a joint source and target byte pair encoding (Sennrich et al., 2016). For inference we use beam search with a beam width of 4 and length penalty of 0.6, following Vaswani et al. (2017) and Ott et al. (2018). As before, we do not modify our model's hyperparameters or training procedure.

Results Table 6 shows that reordering of either the encoder or decoder does not have a significant impact on performance, across the board. We also find that using the most extreme sandwich decoder $(\mathtt{sc})^6\mathtt{f}^6$ performs almost exactly the same as the average baseline; this result is consistent with our observation from Section 4, where we show that the extreme sandwich language model ($\mathtt{s}^{16}\mathtt{f}^{16}$) performs as well as the baseline.

Discussion This experiment indicates that a reordering pattern that benefits one particular task (language modeling) might not carry the same performance gains to another (machine translation). However, it also demonstrates the general robustness of transformer architectures to sublayer reordering, as we did not observe any major performance degradation. Since the sandwich pattern naively groups self- and cross-attention sublayers together, it is also possible that a reordering pattern that takes all three sublayer types into account could potentially improve performance.

Sandwich Coefficient	Encoder Sandwich	Decoder Sandwich
0 (Baseline)	28.74 ± 0.15	
1	28.71	28.64
2	28.71	28.56
3	28.81	28.67
4	28.48	28.66
5	28.45	28.76

Table 6: BLEU on newstest2014 En-De. Our encoder (decoder) sandwich model keeps the decoder (encoder) unmodified. We train the baseline model (Transformer-large with the hyperparameters of Ott et al., 2018) 5 times with different random seeds.

6 Analysis

At the time of writing, we do not have an explanation for why sublayer reordering improves performance on language modeling. However, we are able to determine that sandwich transformers spread their attention in a *different* fashion than interleaved models.

We analyze two baseline models and two sandwich$^{16}_{6}$ models trained with different seeds on the WikiText-103 dataset, by first recording the attention values that each token's heads assign to all other tokens during inference on the validation set. Given the attention outputs of two models, we then compute the models' *attention distance* for each token, and for each self-attention sublayer. This metric compares the attention distribution in the ith self-attention sublayer of the first model to that of the ith self-attention sublayer of the second model, for a specific token.

Given a token and a self-attention sublayer,

Model Pair	Average Attention Distance
Baseline – Baseline	$1.081 \cdot 10^{-3}$
Sandwich – Sandwich	$1.067 \cdot 10^{-3}$
Baseline – Sandwich	$1.289 \cdot 10^{-3} \pm 0.049 \cdot 10^{-3}$

Table 7: The average attention distance, on the WikiText-103 validation dataset, of each model pair. Since there are two baselines and two sandwich transformers (initialized with different random seeds), the distance between the baseline and sandwich models is averaged over all four baseline-sandwich combinations.

we use the Hungarian algorithm (Kuhn, 1955) to find a matching of heads in the first model to heads in the second model $[a_1, b_1], \ldots, [a_8, b_8]$ such that $\sum_{i=1}^{8} \text{EMD}(a_i, b_i)$ is minimized, where $\text{EMD}(a_i, b_i)$ is the earth mover's (Wasserstein) distance between the attention distributions of head a_i in the first model and head b_i in the second model. That minimal value is the attention distance for that token, in that layer. We then average the attention distances across all tokens and layers.

Table 7 shows the average attention distances between every pair of models. We observe that models of the same architecture have significantly lower attention distances than models with different sublayer orderings. This indicates that sublayer reordering has a strong effect on the attention function that the model learns in each head. Future investigations of what this difference is, in a qualitative sense, could potentially provide important insights for designing better reordering patterns.

7 Related Work

7.1 Neural Architecture Search

In this paper, we manually search through a constrained transformer architecture space, after analyzing the results of two small-scale random searches. This human-in-the-loop method for architecture search 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 of instances, 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.

7.2 Transformer Modifications

Much recent work has been devoted to improving transformers by modifying their sublayers. This includes sparsifying their attention patterns, either in an input-based manner (as in Correia et al., 2019), or in a static manner (as in Guo et al., 2019). So et al. (2019) proposed modifying the transformer by adding convolutions and changing the activation function, while others have demonstrated that different initialization schemes (Zhang et al., 2019) and repositioning the layer normalization (Nguyen and Salazar, 2019) can also have a positive effect on performance.

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.

Recently, Lu et al. (2019) introduced a new transformer ordering, where instead of stacking layers of the form **sf** (as in the vanilla interleaved transformer), they stack layers of the form **fsf**. In order keep the total parameter count unchanged, Lu et al. cut the hidden dimension of their feedforward sublayers by half. However, the overall depth of the network is increased by 50%, which causes a similar increase in the model's inference time (Sanh, 2019).

8 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 significantly improves performance over the baseline at no cost in parameters, memory, or runtime.

We then show that the sandwich ordering also improves language modeling performance on a different word-level language modeling bechmark, and that the sandwich pattern can be used to achieve state of the art results on character-level language

modeling. Although sandwich ordering does not improve translation models, we show that they are robust to layer order changes, and that even extreme reorderings (all attention sublayers at the bottom, and all the feedforward sublayers at the top) perform as well as the baseline.

Sublayer reordering can improve the performance of transformer models, but an ordering that improves models on one group of tasks (word/character-level language modeling) might not improve the performance on another task. By showing that sublayer ordering can improve models at no extra cost, we hope that future research continues this line of work by looking into optimal sublayer ordering for other tasks, such as translation, question answering, and classification.

Acknowledgments

We thank Tim Dettmers, Jungo Kasai, Sainbayar Sukhbaatar, and the anonymous reviewers for their valuable feedback.

References

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

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *ICLR*.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. arXiv:1909.00015.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *NAACL*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *ICLR*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *ICLR*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. arXiv:1911.00172.

Harold W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Understanding and improving transformer from a multi-particle dynamic system point of view. arXiv:1906.02762.

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

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. arXiv:1910.05895.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *CMT*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *EACL*.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning.

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

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *ICLR*.

Victor Sanh. 2019. Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *ICML*.

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.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *ACL*.

Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*.

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 *NeurIPS*.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. Improving deep transformer with depth-scaled initialization and merged attention. arXiv:1908.11365.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. arXiv:1506.06724.

Barret Zoph and Quoc V. Le. 2016. Neural architecture search with reinforcement learning. arXiv:1611.01578.