RealFormer: Transformer Likes Residual Attention

Ruining He, Anirudh Ravula, Bhargav Kanagal, Joshua Ainslie
Google Research
{ruininghe,braineater,bhargav,jainslie}@google.com

Abstract

Transformer is the backbone of modern NLP models. In this paper, we propose **RealFormer**, a simple and generic technique to create **Re**sidual **A**ttention **L**ayer Trans**former** networks that significantly outperform the canonical Transformer and its variants (BERT, ETC, etc.) on a wide spectrum of tasks including Masked Language Modeling, GLUE, SQuAD, Neural Machine Translation, WikiHop, HotpotQA, Natural Questions, and OpenKP. We also observe empirically that RealFormer stabilizes training and leads to models with *sparser* attention. Source code and pre-trained checkpoints for RealFormer can be found at https://github.com/google-research/google-research/tree/master/realformer.

1 Introduction

Transformer (Vaswani et al., 2017) architectures are the backbone of numerous state-of-the-art NLP models such as BERT (Devlin et al., 2019), GPT (Radford et al., 2019), and Meena (Adiwardana et al., 2020), and have seen wide success across both academia and industry. Typically, a Transformer network consists of a stack of residual layers. The original design follows a "Post-LN" structure which adds Layer Norm (LN) as a "post-processing" step for each sub-layer, as shown in Figure 1 (a). It has been adopted by various state-of-the-art models including BERT, XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), Transformer-XL (Dai et al., 2019), and ETC (Ainslie et al., 2020). Another notable design is to reorganize the order of modules to create a "direct"/clean path to propagate embeddings of tokens in the input sequence through the whole network, as shown in Figure 1 (b).[1] This design adds LN as a "pre-processing" step for each sub-layer, and is often referred to as "Pre-LN" and used by some well-known extra large models such as GPT-2 (Radford et al., 2019) and Megatron (Shoeybi et al., 2019). In some respect, Post-LN and Pre-LN are analogous to ResNet v1 (He et al., 2016a) and ResNet v2 (He et al., 2016b) respectively in the Computer Vision literature. Although ResNet v2 is usually preferable to v1 for Computer Vision, it does not appear to be the case for Pre-LN Transformer in the NLP literature. It is likely that the particularities of self-attention modules and Transformer architectures potentially favor (at least slightly) different designs compared to traditional convolutional neural networks.

[1]Note that a final LN module is usually added at the very top of the whole network.

In this paper, we propose a simple and generic technique to show that it is beneficial to create a "direct" path to propagate raw attention scores through Transformer-based networks. Our technique is called ***Re**sidual **A**ttention **L**ayer Trans**former***, or ***RealFormer*** in short. We also use *RealFormer* to denote the resulting Transformer networks whenever no confusion may arise. Without losing generality, taking the standard Transformer encoder as an example, each RealFormer layer takes the raw attention scores of all attention heads from the previous layer and adds "residual scores" (computed the same way as attention scores in regular Transformers) on top, as shown in Figure 1 (c). The sum of the two scores is then used to compute attention probabilities via softmax.

In other words, RealFormer can be seen as adding simple skip connections to a backbone Transformer. Since it does not add expensive multiplication ops, performance is expected to be comparable.[2] Note that our technique can also be applied straightforwardly for different Transformer varia-

[2]In certain settings, we find it helpful for RealFormer to use running *mean* of attention scores instead of running *sum*, though it adds some negligible amount of multiplications.

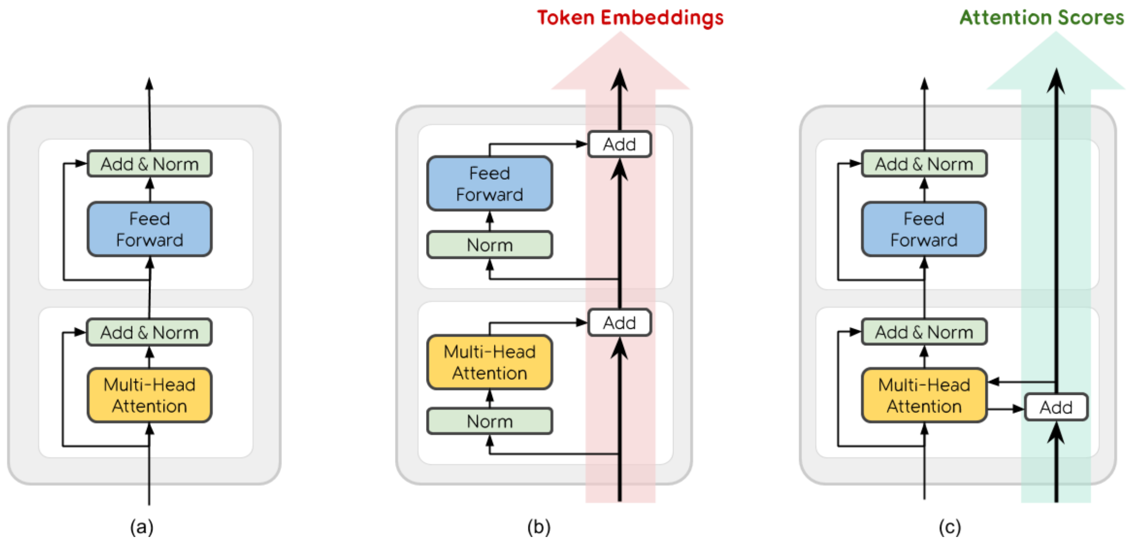


Figure 1: Comparison of different styles of Transformer layers: (a) The prevalent Post-LN layer used by (*e.g.*) BERT; (b) Pre-LN layer used by (*e.g.*) GPT-2 that creates a "direct" path to propagate token embeddings; (c) Our RealFormer layer that creates a "direct" path to propagate attention scores (by adding a simple skip edge on top of (a)). Note that here we are showing Transformer encoder for demonstration purposes only; RealFormer can be applied straightforwardly for different Transformer variations (*e.g.*, when decoders are involved).

tions and even when decoders are involved.

Specifically, our main contributions include:

- We present RealFormer, a simple, generic, and cheap technique to improve Transformer-based networks. It adds no parameters or hyper-parameters, and usually takes no more than a few lines of code changes to implement.

- We show that RealFormer can be used as a drop-in replacement of Transformer in BERT, outperforming both Post-LN and Pre-LN Transformers across a wide spectrum of model sizes for pre-training. In terms of fine-tuning, it even achieves competitive downstream results when pre-trained with only half the number of epochs of the baselines.

- We further demonstrate the genericity of RealFormer by using it as a drop-in replacement of two recent state-of-the-art Transformer variation models: ADMIN (Liu et al., 2020) from the Neural Machine Translation (NMT) domain, and ETC (Ainslie et al., 2020) that extends Transformer to handle long and structured inputs. We show that RealFormer can improve these models significantly on various tasks and lead to new state-of-the-art results.

- Qualitatively, we observe that attention in RealFormer tends to be *sparser* and more correlated across layers compared to baselines, which we believe may have some regularization effects that could stabilize training and benefit fine-tuning.

2 Related Work

Vaswani et al. (2017) proposed Transformer initially for NMT and it has profoundly changed the NLP field ever since.

Radford et al. (2018) demonstrated that generative pre-training of a Transformer-based language model (GPT) on a diverse corpus of unlabeled text can give large gains to downstream NLP tasks that suffer from scarce labeled data. Following this thread, Devlin et al. (2019) proposed to pre-train a *bidirectional* Transformer encoder (BERT) with a novel Masked Language Modeling as the main optimization objective. Since then, advances on many NLP tasks have been dominated by the self-supervised general-purpose pre-training, task-specific fine-tuning paradigm. Following BERT, there has been a large stream of work that explores better self-supervision objectives (*e.g.*, Yang et al. (2019); Clark et al. (2020)), larger pre-training data and better hyper-parameters (*e.g.*, Liu et al. (2019)), model parameter sharing (*e.g.*, Lan et al. (2019)), multi-task pre-training (*e.g.*, Sun et al. (2020); Raffel et al. (2020)). These efforts typically employ a Post-LN Transformer at their core. In this paper we adopt BERT to test different Transformer architectures because it is widely used and representative of this body of work.

Another notable thread of work focuses on improving the efficiency/scalability of Transformer. Typically, they try to reduce the *quadratic* complexity of the self-attention mechanism with respect to sequence length via low-rank methods (*e.g.*, Wang et al. (2020)), fixed strided attention patterns (*e.g.*, Child et al. (2019)), learnable attention patterns (*e.g.*, Kitaev et al. (2020); Roy et al. (2020)), memory-based global & local attention (*e.g.*, Ainslie et al. (2020); Beltagy et al. (2020); Zaheer et al. (2020)), and so on. These methods are particularly useful when dealing with long documents that go beyond the capacity of standard Transformer models. We would refer the reader to Tay et al. (2020) for a detailed survey. RealFormer is orthogonal to these methods as it focuses on improving various Transformer networks with an universal technique which can apply to these models as well. In this paper, we will use RealFormer to improve a state-of-the-art model, ETC (Ainslie et al., 2020), from this line of work to demonstrate the universality of RealFormer.

Some recent work (*e.g.*, Wang et al. (2019b); Xiong et al. (2020); Zhang et al. (2018); Huang et al. (2020); Zhang et al. (2019)) has studied normalization and parameter initialization schemes for Transformers, though most evaluations focus only on NMT to the best of our knowledge. In this strand, Liu et al. (2020) recently proposed ADMIN, which achieved state-of-the-art results on multiple popular NMT benchmarks. In this paper, we will take ADMIN as an example to (1) evaluate RealFormer in settings involving decoders, and (2) show that it is possible to apply RealFormer on top of this line of work.

3 RealFormer

3.1 Standard Transformer

There is an encoder and a decoder in Transformer (Vaswani et al., 2017). Since they work in a similar way, here we only introduce the encoder and refer the reader to the original paper for complete details.

There are two sub-layers inside each layer of a Transformer encoder. The first sub-layer contains a Multi-Head Attention module that computes output embeddings of a set of queries (Q) by aggregating the embeddings (V) of a set of keys (K):

$$\text{MultiHead}\,(Q, K, V) = \\ \text{Concat}\,(head_1, ..., head_h)\,W^O,$$

where $head_i = \text{Attention}\,(QW_i^Q, KW_i^K, VW_i^V)$. Q and K are matrices with dimension d_k and V is a matrix with dimension d_v. W_i^Q, W_i^K, and W_i^V are matrices that linearly project queries, keys, and values into the "attention space" of the i-th head. W^O is a matrix that linearly transforms the concatenation of the outputs of all heads.

The attention function is typically implemented with a Scaled Dot-Product Attention module (Vaswani et al., 2017) which computes a weighted sum of the values:

$$\text{Attention}\,(Q', K', V') = \text{Softmax}\,(\frac{Q'K'^T}{\sqrt{d_k}})\,V',$$

where matrix $\frac{Q'K'^T}{\sqrt{d_k}}$ contains the raw attention scores for each (query, key) pair. These scores are normalized via the Softmax function for each query and then act as weights for the corresponding vectors in V'.

The second sub-layer contains a fully-connected Feed-Forward Network (FFN) module with one hidden layer:

$$\text{FFN}\,(x) = \sigma\,(x\,W_1 + b_1)\,W_2 + b_2,$$

where σ is an activation function usually implemented with ReLU or GELU (*e.g.*, Devlin et al. (2019)). FFN is applied to each position in the sequence separately and identically. Finally, there are Layer Norm (LN) modules inserted into the above two sub-layers to stabilize training.

As shown in Figure 1, there are two canonical designs of the Transformer network which only differ in the ways they organize the modules. Post-LN is the original architecture proposed by Vaswani et al. (2017) which normalizes the outputs at the end of each sub-layer. In contrast, Pre-LN normalizes sub-layer inputs instead and creates a direct path (without LN in the way) to propagate embeddings of the tokens in the sequence.

3.2 Residual Attention Layer Transformer

RealFormer uses a Post-LN style Transformer[3] as backbone and adds skip edges to connect

[3]Potentially we could also use Pre-LN, but Post-LN tends to outperform Pre-LN, as we will show in Section 4.

Multi-Head Attention modules in adjacent layers, as shown in Figure 1 (c). More formally, it adds $Prev$, the pre-softmax attention scores from the previous layer with shape $(\#heads, from\_seq\_len, to\_seq\_len)$,[4] as one additional input to the Multi-Head Attention module in the current layer:

$$\text{ResidualMultiHead}\,(Q, K, V, Prev) = \text{Concat}\,(head_1, ..., head_h)\,W^O,$$

where $head_i = \text{ResidualAttention}\,(QW_i^Q, KW_i^K, VW_i^V, Prev_i)$ and $Prev_i$ is the slice of $Prev$ with shape $(from\_seq\_len, to\_seq\_len)$ corresponding to $head_i$. ResidualAttention adds "residual scores" on top of $Prev_i$ and then computes the weighted sum as usual:

$$\text{ResidualAttention}\,(Q', K', V', Prev') = \text{Softmax}\,(\frac{Q'K'^T}{\sqrt{d_k}} + Prev')\,V'. \quad (1)$$

Finally, new attention scores $\frac{Q'K'^T}{\sqrt{d_k}} + Prev'$ are passed over to the next layer.

Implementing RealFormer takes no more than adding a few lines of code to the backbone Transformer. Note that the RealFormer technique can be straightforwardly applied for Transformer variations and even when there are more than one type of attention modules in the network. For example, there are encoder self-attention, encoder-decoder attention, and decoder self-attention modules for machine translation. In such cases, RealFormer simply adds skip edges to create *multiple* direct paths, one for each type of attention module.

Discussion. Adding skip edges is equivalent to using a softmax over the running sum of the attention scores (to get attention probabilities). This might be sub-optimal for very deep networks due to the linear scaling nature of sum. Empirically, we find it helpful to use running *mean* instead in such cases, which can be viewed as adding a temperature (*i.e.*, #traversed layers) to the softmax function in Eq. 1 of each RealFormer layer.

4 Experiments

To demonstrate that RealFormer is general-purpose, we conduct comprehensive empirical studies on a variety of tasks including (masked) language modeling, machine translation, and long document modeling, based on corresponding state-of-the-art models: BERT, ADMIN, and ETC. To evaluate its robustness, we only do minimal (if at all) hyper-parameter tuning for RealFormer and initialize all parameters the same way as the backbone Transformers. More aggressive hyper-parameter tuning or better initialization might further improve RealFormer, though we leave them for future work. Details of our experiments are included in Appendix.

4.1 BERT

BERT (Devlin et al., 2019) has been the standard way of transferring knowledge from large unlabeled text corpora by pre-training a bidirectional Transformer encoder. Numerous downstream NLP tasks suffering from scarcity of supervised data have benefited considerably by fine-tuning a pre-trained BERT model. This drives us to adopt BERT as the main evaluation setup for RealFormer.

Experiment setup. Our experiments are based on the official BERT repository[5]. We follow the standard pre-training setup (dataset: Wikipedia + BookCorpus, vocab: uncased 30K, max sequence length: 512[6], dropout: 10%, learning rate: 1e-4, learning rate schedule: warm up and then linearly decay to 0, weight decay: 0.01, optimizer: AdamW, objective: Masked Language Modeling + Next Sentence Prediction, etc.) to compare three Transformer models: Post-LN, Pre-LN, and RealFormer. We experiment with Transformer architectures with a wide spectrum of sizes as detailed in Table 1. For simplicity, all models are pre-trained 1M steps with a mini-batch size of 512 (except that xLarge uses 256 to avoid TPU OOM). Note that we use a larger mini-batch size than Devlin et al. (2019), *i.e.*, doubling the amount of pre-training epochs, to show more complete behavior of different models.

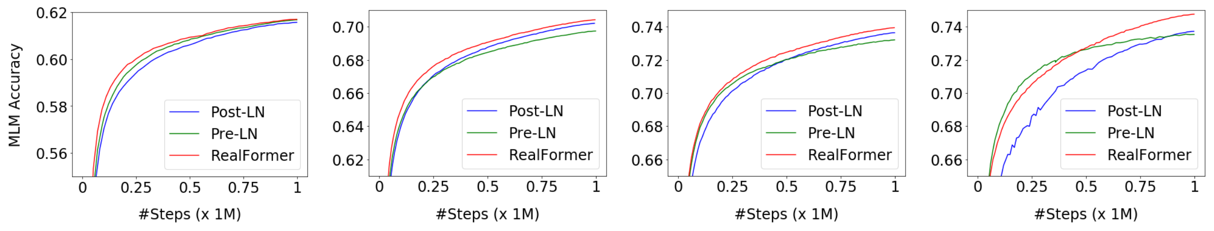
We use exactly the same setup for all three Transformer architectures except that for the Pre-LN Transformer we follow the initialization strategy suggested by Radford et al. (2019) and Child et al. (2019).[7] Note that for simplicity RealFormer reuses all hyper-parameter setups from Post-LN Transformer unless otherwise specified. We use running sum of attention scores for all RealFormer

[4]Batch dimension is omitted for ease of discussion.

[5]`https://github.com/google-research/bert`

[6]Unlike BERT which uses a reduced sequence length for the first 90% of steps, we always use 512 for simplicity.

[7]We also tried BERT-style initialization in our pilot experiments but without success.



(a) BERT-Small (b) BERT-Base (c) BERT-Large (d) BERT-xLarge

Figure 2: Development set MLM accuracy (best viewed in color). Improvement gap of RealFormer over the best baseline tends to increase with model size. Note that these are without hyper-parameter tuning for RealFormer. (As we will show later, RealFormer can benefit from larger learning rates and even double the gap size over Post-LN.)

Model	L	H	A	I	P
BERT-Small	4	512	8	2,048	30M
BERT-Base	12	768	12	3,072	110M
BERT-Large	24	1,024	16	4,096	340M
BERT-xLarge	36	1,536	24	6,144	1B

Table 1: Model architectures for BERT evaluation. L: #layers, H: hidden size, A: #heads, I: intermediate size, P: approximate #parameters.

Model	Post-LN	Pre-LN	RealFormer
BERT-Small	61.57%	61.67%	**61.70%**
BERT-Base	70.20%	69.74%	**70.42%**
BERT-Large	73.64%	73.21%	**73.94%**
BERT-xLarge	73.72%	73.53%	**74.76%**

Table 2: Development set MLM accuracy after pre-training 1M steps. RealFormer outperforms baselines more as model size increases.

models except xLarge (for which we use running mean for reasons discussed in Section 3.2).

All experiments are performed on 128 or 256 TPU v3 cores depending on model sizes (see Appendix A.1 for details).

4.1.1 Pre-training Results

To evaluate pre-trained models, we report Masked Language Modeling (MLM) accuracy[8] on a randomly held-out development set. As shown in Table 2, RealFormer outperforms the two baseline Transformers considerably with the gap increasing with model size. Our hypothesis is that larger models are inherently harder to train (*e.g.*, we observe that BERT with Post-LN is unstable and sometimes even diverges for xLarge) and RealFormer can help regularize the model and stabilize training.

We also report the pre-training curves in Figure 2. One interesting finding is that the Pre-LN Transformer seems to favor the combination of extra large models and a small number of steps, though it is consistently outperformed by the other two in “regular-sized” settings or given enough pre-training budget.

4.1.2 Downstream Results

To evaluate downstream performance, we fine-tune the above pre-trained BERT-Large models on both sentence-level (*i.e.*, GLUE) and token-level (*i.e.*, SQuAD) NLP tasks.

GLUE. General Language Understanding Evaluation (GLUE) is a canonical benchmark proposed by Wang et al. (2019a) for evaluating models across a diverse set of NLU tasks. Following the fine-tuning recipe in Devlin et al. (2019), we use a mini-batch size of 32 for all models on all tasks. For each (task, model) pair, we select number of fine-tuning epochs in {2, 3, 4} and learning rate in {6e-6, 8e-6, 1e-5, 2e-5, 3e-5, 4e-5, 5e-5}.[9] For each setup, we run the experiment five times and report the best median performance and the corresponding standard deviation on the development set.

Results are tabulated in Table 3. We exclude the problematic WNLI task following Devlin et al. (2019). For each task, we report metric(s) suggested by Wang et al. (2019a). RealFormer achieves the best overall performance and outperforms both baselines on most tasks, testifying its strength at tackling sentence-level tasks.

SQuAD. The Stanford Question Answering Dataset (SQuAD v1.1) is a reading comprehension dataset consisting of 100K crowd-sourced question-answer pairs, where the answer to each question is

[8] All methods achieved similar (and great) results on Next Sentence Prediction presumably because it is much easier.

[9] We use a slightly wider range than Devlin et al. (2019) to better accommodate all three models.

Task	Post-LN	Pre-LN	RealFormer
MNLI-m	85.96±0.11	85.03±0.12	**86.28**±0.14
MNLI-nm	85.98±0.14	85.05±0.19	**86.34**±0.30
QQP	91.29±0.10	91.29±0.16	**91.34**±0.03
QQP (F1)	**88.34**±0.15	88.33±0.26	88.28±0.08
QNLI	92.26±0.15	**92.35**±0.26	91.89±0.17
SST-2	92.89±0.17	93.81±0.13	**94.04**±0.24
CoLA (MC)	58.85±1.31	58.04±1.50	**59.83**±1.06
STS-B (PC)	90.08±0.27	90.06±0.33	**90.11**±0.56
STS-B (SC)	89.77±0.26	89.62±0.28	**89.88**±0.54
MRPC	**87.50**±0.67	86.76±5.64	87.01±0.91
MRPC (F1)	**91.16**±0.45	90.69±3.16	90.91±0.65
RTE	71.12±2.52	68.59±1.52	**73.65**±0.90
Overall	84.01	83.47	**84.53**

Table 3: GLUE development set results of fine-tuning BERT-Large models in Table 2. Default metric: accuracy, MC: Matthews correlation, PC: Pearson correlation, SC: Spearman correlation. Overall: first average metrics within each task (if there are 1+) and then across tasks. Numbers in smaller font are standard deviations. All numbers are scaled by 100.

SQuAD	Public	Post-LN	Pre-LN	RealFormer
v1.1 (F1)	90.9	91.68±0.12	91.06±0.09	**91.93**±0.12
v1.1 (EM)	84.1	85.15±0.13	83.98±0.24	**85.58**±0.15
v2.0 (F1)	81.9	82.51±0.12	80.30±0.12	**82.93**±0.05
v2.0 (EM)	78.7	79.57±0.12	77.35±0.16	**79.95**±0.08

Table 4: SQuAD development set results of fine-tuning BERT-Large models in Table 2. EM: exact match. Public: Post-LN results from Devlin et al. (2019). Numbers in smaller font are standard deviations. All numbers are scaled by 100.

a segment of text from the corresponding reading passage (Rajpurkar et al., 2016). SQuAD v2.0, a later version, further extends with over 50K unanswerable questions written adversarially by crowd-workers to look similar to answerable ones.

We follow the fine-tuning recipe in Devlin et al. (2019) for all three Transformer models on these two datasets without using any additional data such as TriviaQA (Joshi et al., 2017). For both v1.1 and v2.0, we select mini-batch size in {32, 48}, number of fine-tuning epochs in {2, 3, 4}, and learning rate in {2e-5, 3e-5, 4e-5, 5e-5}. For each setup, we run the experiment five times and report the best median performance and the corresponding standard deviation on the development set. As we can see from Table 4, RealFormer outperforms the two baselines considerably, attesting its strength at tackling token-level tasks.

Task	Post-LN (500K)	Post-LN (1M)	RealFormer (500K)
GLUE	83.84	84.01	84.34
v1.1 (F1)	91.46±0.18	91.68±0.12	91.56±0.09
v1.1 (EM)	84.87±0.24	85.15±0.13	85.06±0.12
v2.0 (F1)	81.44±0.50	82.51±0.12	82.52±0.55
v2.0 (EM)	78.64±0.48	79.57±0.12	79.54±0.54
Overall	83.97	84.37	**84.51**

Table 5: Downstream development set results of fine-tuning BERT-Large with Post-LN and RealFormer pre-trained with different number of steps. v*: SQuAD version, EM: exact match. Overall: First average across SQuAD and then GLUE. Numbers in smaller font are standard deviations. All numbers are scaled by 100.

4.1.3 Research Questions

How well does RealFormer perform with half the pre-training budget? Although RealFormer has outperformed both Post-LN and Pre-LN considerably when pre-training 1M steps, we are also interested in investigating its potential when the pre-training budget is more limited. For this purpose, we experiment with BERT-Large models. In particular, we take the 500K step checkpoint of the pre-trained RealFormer in Table 2 and fine-tune it on GLUE and SQuAD datasets using exactly the same procedure as described above. Comparison results against the strongest baseline, Post-LN Transformer pre-trained 500K (checkpoint) and 1M steps respectively, are collected in Table 5. We can see that RealFormer with merely half the amount of pre-training epochs can beat Post-LN (1M) on GLUE with a significant margin, and almost match its performance on SQuAD.

Does a larger learning rate help? As suggested by some recent work (*e.g.*, Xiong et al. (2020)), Pre-LN Transformer may benefit from using larger learning rates. To this end, we follow the pre-training procedure detailed earlier and switch to a larger learning rate, 2e-4, to pre-train BERT-Large with the three Transformer models. Development set MLM accuracy with training steps can be found in Figure 3. We find that both Pre-LN and RealFormer can reap some benefits of using larger learning rates with RealFormer seeming to benefit slightly more in this case (73.94% $\rightarrow$ 74.31%) compared to Pre-LN (73.21% $\rightarrow$ 73.46%). Post-LN diverges with the learning rate of 2e-4. Note that it also means that RealFormer can outperform Post-LN, the strongest baseline, actually with a

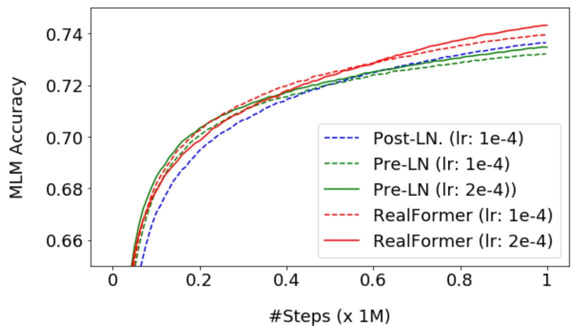


Figure 3: Development set MLM accuracy of BERT-Large with different learning rates (best viewed in color). RealFormer seems to benefit slightly more from using a larger, non-default learning rate compared to Pre-LN, while Post-LN diverges with 2e-4.

prominent gap, 0.67% (*i.e.*, 74.31% - 73.64%), for pre-training, though with only minimal learning rate tuning.

Is attention sparser in RealFormer? We conduct one empirical study to observe the qualitative differences between RealFormer and Post-/Pre-LN Transformers. We randomly sample 8,192 examples from the held-out development set and visualize the distribution of attention probabilities of each token in these examples across heads in all layers. In particular, for each (token, layer, head) triplet, we compute the entropy of the attention probabilities as the "sparsity measure" of attention. Intuitively, as entropy gets lower, the attention weight distribution becomes more skewed and therefore attention is sparser.

In a similar fashion to Ramsauer et al. (2020), we use violin plots to show the entropy distributions of the pre-trained BERT-Base model with RealFormer from Table 2 (see Figure 4). Plots for the two baseline Transformers in Table 2 are included in Appendix A.4. Each row is a layer in BERT-Base and each column is an attention head.

We find that attention tends to get sparser for later (upper) layers for all three Transformers. However, RealFormer differs from the two baselines in the following ways:

- RealFormer has *significantly sparser* attention for top layers (layer 9-11);
- RealFormer tends to have lower variance across all layers, which means that attention density is less input-dependent.

We hypothesize that the above two properties might be a sign of stableness and benefit fine-tuning.

Dropout	Post-LN	Pre-LN	RealFormer
0%[10]	71.16%	69.80%	**71.30%**
10%	73.64%	73.21%	**73.94%**
20%	73.21%	72.97%	**73.66%**

Table 6: Development set MLM accuracy of BERT-Large with different dropout rates.

Do attention heads in layer L resemble those in layer $L-1$? Since RealFormer uses a residual attention scheme, it is interesting to show to what extent an attention head is "relying on" the corresponding head in the previous layer. To this end, we take each of the three pre-trained BERT-Base models in Table 2 and compute the Jensen-Shannon Divergence (JSD) between attention probabilities in each pair of *vertically* adjacent heads, *i.e.*, JSD $(\text{head}_i^L, \text{head}_i^{L-1})$, for $1 \leq L < 12$ and $0 \leq i < 12$.

Appendix A.5 demonstrates detailed JSD distributions of Post-LN and RealFormer respectively based on 8,192 held-out examples. We observe that RealFormer tends to have *significantly lower* JSD values (*i.e.*, indicating more "similar" attention across layers), especially for heads in middle layers. This might mean that RealFormer has some regularization advantages and provides one hypothesis for why it tends to outperform Post-LN more for larger models. Note that head_i^L can still be useful even if it has exactly the same attention probabilities with head_i^{L-1} because of the existence of the FFN sublayer and the potential differences in value matrices (*i.e.*, V' in Eq. 1).

Is residual attention really necessary? One may wonder whether increasing dropout rate can already regularize large models well so that residual attention is redundant. To this end, we experiment with different dropout rates for pre-training BERT-Large with different Transformers (following the procedures in Section 4.1.1). Results are collected in Table 6, from which we can see that (1) RealFormer outperforms the two baselines across all dropout settings, and (2) simply increasing dropout rate can not regularize Transformer models as well as what residual attention appears to be doing.

4.2 ADMIN

To evaluate the genericity of RealFormer, here we try it on top of ADMIN (Liu et al., 2020), a state-of-

[10]When dropout rate is 0%, we use early stopping for all models due to overfitting.

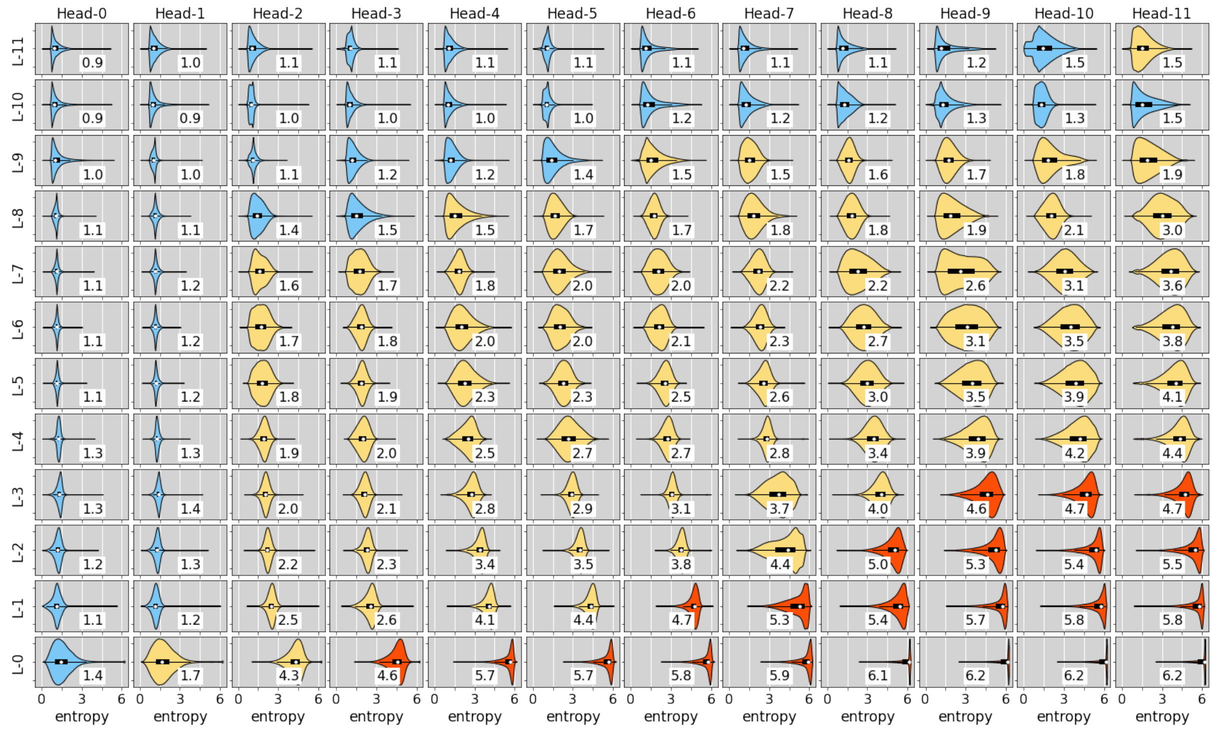


Figure 4: Distribution of entropies of the attention probabilities of the tokens of 8,192 held-out examples using the pre-trained BERT-Base with **RealFormer** (see Section 4.1.1). For better legibility, (1) attention heads in each layer are ordered by their medians of entropies, and (2) distributions are color-coded based on the median of entropies: RED (median $>$ 4.5), YELLOW (1.5 $\leq$ median $\leq$ 4.5), BLUE (median $<$ 1.5), *i.e.*, colder colors mean sparser attention. There is a clear trend that higher layers tend to have *sparser* attention.

the-art NMT model without using either additional data or data augmentation. ADMIN adopts Post-LN as the backbone, which we simply replace with RealFormer. In particular, we add three types of skip edges for encoder-encoder, encoder-decoder, and decoder-decoder attention respectively to the Post-LN Transformer. Empirically, RealFormer with running mean of attention scores tends to outperform running sum for our experiments, therefore here we use the former exclusively for brevity.

We use two popular NMT benchmarks, WMT'14 En-De and WMT'14 En-Fr, and follow Liu et al. (2020) for all training setups on both benchmarks except that in all cases (1) we select the peak learning rate from {5e-4, 1e-3, 1.2e-3} and use a linear learning rate decay schedule (instead of inverse sqrt);[11] (2) we train RealFormer only 50 epochs (in contrast, ADMIN trains 100 epochs on En-De and 50 epochs on En-Fr); and (3) we average across the last 25 checkpoints (while ADMIN uses the last 10). More checkpoints are helpful for us (especially for large models) presumably because the last few are not "diverse" enough as learning rate decays to 0.

Our experiments are performed on NVIDIA A100 GPUs, based on the official ADMIN repository[12]. We follow Liu et al. (2020) to configure the amount of GPUs to use for different setups.

BLEU scores on test sets are collected in Table 7. For fair comparisons, we also run ADMIN using our above setups and report results in the same table. Following Liu et al. (2020), all networks (including both encoders and decoders) share the same width setup (hidden size 512, intermediate size 2048, 8 heads) and only vary in depth. RealFormer outperforms all baselines across all depths considerably with a new state-of-the-art BLEU score (43.97) on En-Fr for models not using additional data or data augmentation to the best of our knowledge. One interesting observation here is that RealFormer does not always lead to larger improvement gaps for larger models, which might be due to the checkpoint averaging mechanism (which potentially regularizes large models reasonably well).

[11]With inverse sqrt decay, we find that RealFormer tends to favor larger peak learning rates than what Liu et al. (2020) uses, and we have also seen improvements in most cases.

[12]`https://github.com/LiyuanLucasLiu/Transformer-Clinic`

Model	En-De 6L-6L	En-De 12L-12L	En-De 18L-18L	En-Fr 6L-6L	En-Fr 60L-12L
Post-LN	27.80	failed	failed	41.29	failed
Pre-LN	27.27	28.26	28.38	40.74	43.10
ADMIN	27.90	28.58	29.03	41.47	43.80
ADMIN†	28.06	28.85	29.11	41.65	43.72
Ours	**28.17**	**29.06**	**29.35**	**41.92**	**43.97**

Table 7: Test set BLEU scores on two WMT'14 benchmarks using different sizes of models. xL-yL: #Encoder layers-#Decoder layers. First three rows are from Liu et al. (2020). Ours is switching the backbone of ADMIN from Post-LN to RealFormer. †Our run of ADMIN using the same setups as RealFormer.

Task	Metric	ETC	Ours
WikiHop	Accuracy	78.92$\pm$0.14	**79.21**$\pm$0.38
HotpotQA	Ans. F1	80.38$\pm$0.13	**80.86**$\pm$0.16
	Sup. F1	89.07$\pm$0.06	**89.21**$\pm$0.12
	Joint F1	73.12$\pm$0.19	**73.57**$\pm$0.19
Natural Questions	Long Ans. F1	77.70$\pm$0.15	**77.93**$\pm$0.31
	Short Ans. F1	58.54$\pm$0.41	**59.10**$\pm$0.81
	Average F1	68.07$\pm$0.17	**68.51**$\pm$0.56
OpenKP	F1@3	44.06$\pm$0.08	**44.27**$\pm$0.08

Table 8: Development set results of ETC-Large. Ours is adding residual attention edges to ETC. Numbers in smaller font are standard deviations. All numbers are scaled by 100.

4.3 ETC

Extended Transformer Construction (ETC) is a recent sparse attention mechanism proposed by Ainslie et al. (2020) and Zaheer et al. (2020) to handle long context. It has achieved state-of-the-art results on four natural language benchmarks requiring long and/or structured inputs. Here we evaluate RealFormer on top of ETC models on these benchmarks including WikiHop (Welbl et al., 2018), HotpotQA (Yang et al., 2018), Natural Questions (Kwiatkowski et al., 2019), and OpenKP (Xiong et al., 2019). They vary significantly in terms of dataset size, context length, and structure in text inputs. Please refer to Ainslie et al. (2020) for more details.

Our experiments are based on the official ETC repository[13]. We take the ETC-Large model (24 layers, 1024 hidden size, 16 heads), add residual attention edges (*i.e.*, using running sum), and follow all the pre-training and fine-tuning recipes as well as hardware setups detailed in Ainslie et al. (2020). For each fine-tuning setup, we run the experiment five times and report the best median performance and the corresponding standard deviation on the development set in Table 8. RealFormer can improve ETC consistently across all four benchmarks.

As of June 2021, we are ranked the first on the WikiHop leaderboard[14] with a test accuracy of 84.4% (2.1% absolute improvement over the previous best result).

[13] https://github.com/google-research/google-research/tree/master/etcmodel
[14] http://qangaroo.cs.ucl.ac.uk/leaderboard.html

5 Conclusions

We propose RealFormer, a simple, generic, and cheap technique based on the novel idea of residual attention to improve Transformer-based networks. Quantitatively, we show that RealFormer can improve a diverse set of state-of-the-art Transformer-based models considerably for tasks like Masked Language Modeling, Neural Machine Translation, and long document modeling. Qualitatively, we show that RealFormer tends to have comparatively *sparser* attention, both within heads and across heads in adjacent layers.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283.

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977.*

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 268–284.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150.*

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In *European Cconference on Computer Vision*, pages 630–645.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. 2020. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1601–1611.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5747–5763.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, et al. 2020. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8968–8975.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019b. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822.

Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open domain web keyphrase extraction beyond language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5178–5187.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pages 5753–5763.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big Bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. Improving deep transformer with depth-scaled initialization and merged attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 897–908.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. 2018. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*.

Model	Post-LN	Pre-LN	RealFormer
BERT-Small	5.4 hrs	5.3 hrs	5.9 hrs
BERT-Base	20 hrs	20 hrs	23 hrs
BERT-Large	58 hrs	58 hrs	66 hrs
BERT-xLarge	136 hrs	137 hrs	137 hrs

Table 9: Pre-training time of different BERT models in Table 2. We use 128 TPU v3 cores and mini-batch size 512 for BERT-Small/Base/Large, and 256 TPU v3 cores and mini-batch size 256 for BERT-xLarge.

A Appendices

A.1 Training Details: BERT

All our experiments are conducted on TPUs based on `https://github.com/google-research/bert`, the official BERT repository in TensorFlow (Abadi et al., 2016).

Pre-training. We use 128 TPU v3 cores (*i.e.*, 64 chips) for BERT-Small/Base/Large and 256 TPU v3 cores (*i.e.*, 128 chips) for BERT-xLarge. Table 9 demonstrates the time used to pre-train each model 1M steps. We can see that there is 10%-15% performance drop when adding residual attention edges for all sizes except xLarge. Our suspicion is that additions are not as optimized as other ops like matrix multiplications on TPU v3 cores. There is a much smaller performance drop for xLarge though, which might indicate that addition scales nicely compared to other ops on TPU v3 cores. As we will show later in Appendix A.2, performance drop on GPUs is almost negligible across different Transformer sizes, suggesting that it is hardware-dependent.

Fine-tuning. We use 8 TPU v2 cores (*i.e.*, 4 chips) to fine-tune each model. Best hyper-parameter configurations for BERT-Large with RealFormer on GLUE and SQuAD are collected in Table 10. We include RealFormer pre-trained both 1M and 500K steps, corresponding to the results in Table 3, 4, and 5.

A.2 Training Details: ADMIN

All our NMT experiments are conducted on NVIDIA A100 GPUs based on `https://github.com/LiyuanLucasLiu/Transformer-Clinic`, the official ADMIN repository implemented via fairseq (Ott et al., 2019). We use the same scripts to collect and process data and evaluate all models.

Following Liu et al. (2020), we use different number of GPUs for different setups, as detailed in

Task	500K-step			1M-step		
	BS	LR	EP	BS	LR	EP
MNLI	32	2e-5	2	32	1e-5	4
QQP	32	3e-5	4	32	2e-5	4
QNLI	32	3e-5	4	32	2e-5	2
SST-2	32	3e-5	2	32	1e-5	4
CoLA	32	2e-5	4	32	1e-5	3
STS-B	32	2e-5	3	32	2e-5	4
MRPC	32	2e-5	4	32	1e-5	4
RTE	32	1e-5	4	32	1e-5	4
SQuAD v1.1	48	3e-5	2	48	3e-5	2
SQuAD v2.0	32	5e-5	2	48	5e-5	2

Table 10: Hyper-parameter configurations on GLUE and SQuAD for best-performing BERT-Large with RealFormer (pre-trained 500K steps and 1M steps respectively). BS: mini-batch size, LR: learning rate, EP: #fine-tuning epochs.

Model	En-De			En-Fr	
	6L-6L	12L-12L	18L-18L	6L-6L	60L-12L
ADMIN	9.3 hrs	16 hrs	23 hrs	28 hrs	70 hrs
Ours	9.4 hrs	16 hrs	23 hrs	28 hrs	72 hrs
#GPUs	4	4	4	8	16

Table 11: Number of GPUs and training time used for each model. Ours is switching the backbone of ADMIN from Post-LN to RealFormer (using running mean of attention scores).

Table 11. For our runs of ADMIN and RealFormer (*i.e.*, the last two rows in Table 7), learning rate is set to 5e-4 on WMT'14 En-De and 1.2e-3 on WMT'14 En-Fr across different model sizes. We train all models 50 epochs across the two benchmarks and average across the last 25 checkpoints (corresponding to the last 25 epochs).

Training time comparison of ADMIN and our model using the same setups is shown in Table 11. Adding residual attention edges and using running mean of attention scores do not incur significant performance drop on GPUs across different model sizes.

A.3 Training Details: ETC

All our experiments are conducted on TPU v3 cores based on the official ETC repository in TensorFlow: `https://github.com/google-research/google-research/tree/master/etcmodel`.

Pre-training. As is the case with ETC-Large (Ainslie et al., 2020), we find that pre-training ETC-Large with RealFormer can also benefit sig-

Task	**Instance Statistics**				**Hyper-parameter**		
	#Training	Median Length	95%tile Length	Max Length	BS	LR	EP
WikiHop	43,738	1,541	3,994	20,337	32	4e-5	15
HotpotQA	90,447	1,227	1,810	3,560	32	4e-5	5
Natural Questions	307,373	4,004	17,137	156,551	64	3e-5	2
OpenKP	133,724	761	4,546	89,183	64	3e-5	2

Table 12: Statistics of benchmarks and the hyper-parameter configurations for best-performing ETC-Large with RealFormer. BS: mini-batch size, LR: learning rate, EP: #fine-tuning epochs.

nificantly from lifting weights from RoBERTa (Liu et al., 2019). Note however that we lift from the same RoBERTa checkpoint as our ETC-Large baseline, which could be disadvantageous to our model since RoBERTa is pre-trained *without* residual attention.

Fine-tuning. Statistics of the four benchmarks and the corresponding best hyper-parameter configurations for ETC-Large with RealFormer are collected in Table 12. On Natural Questions and OpenKP, we simply reuse the best configurations for our ETC-Large baselines as reported in Ainslie et al. (2020). On WikiHop and HotpotQA, we follow the hyper-parameters search space specified in Ainslie et al. (2020) for ETC-Large.[15] In addition, on WikiHop we found it to be slightly better (development set accuracy 79.21 vs 78.96) to turn off RealFormer during fine-tuning (*i.e.*, adding no residual attention but still loading from our pre-trained RealFormer checkpoint); therefore we adopted this setup for WikiHop in Table 8 and our leaderboard submission.

A.4 Entropy Distribution of Pre-trained Baseline Transformer Models

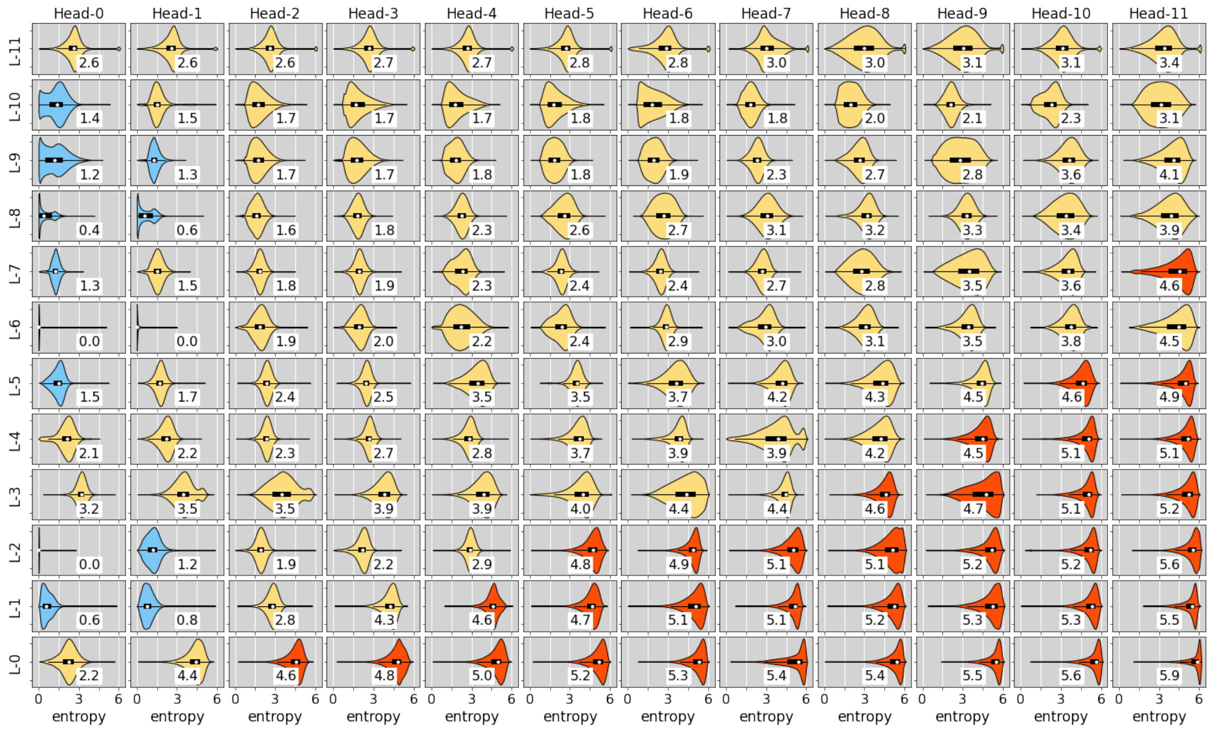
Violin plots demonstrating the entropy distributions of the pre-trained BERT-Base models with Post-LN and Pre-LN Transformers from Table 2 are included in Figure 5.

A.5 Jensen-Shannon Divergence of Different Pre-trained Transformers

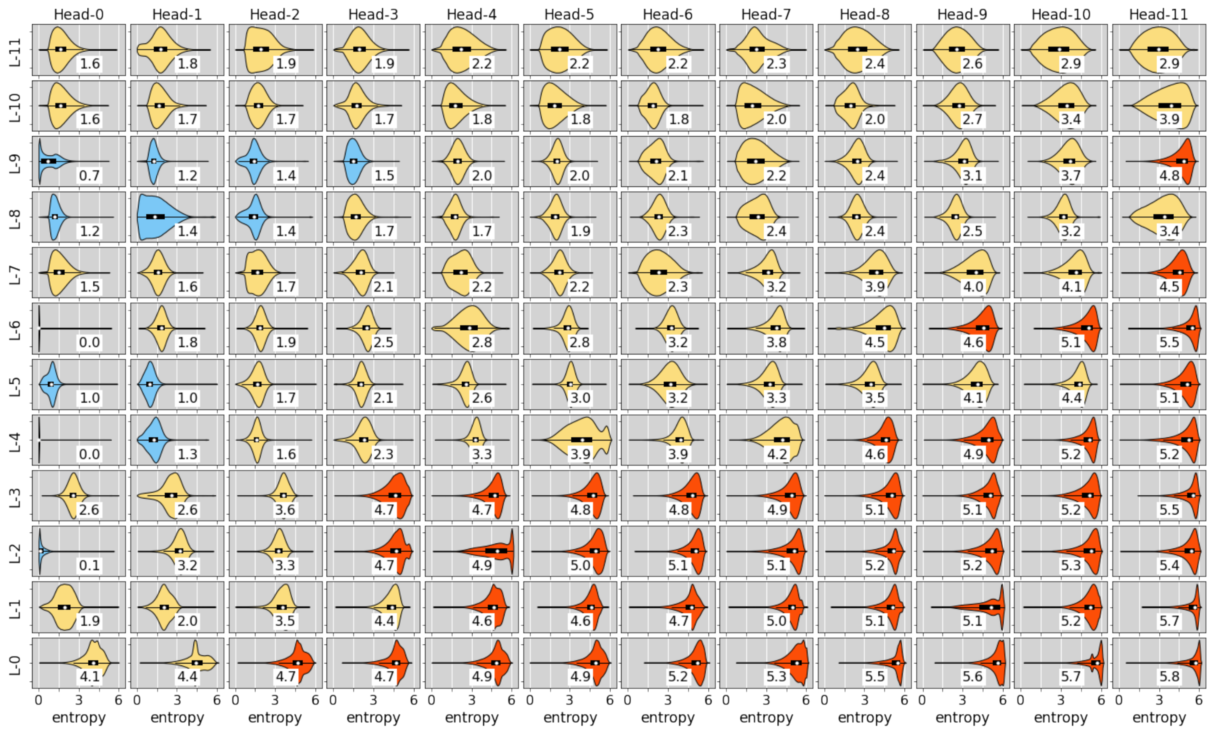
We use violin plots to show the Jensen-Shannon Divergence distributions of the pre-trained BERT-Base models with Post-LN and RealFormer from Table 2 respectively (see Figure 6). Each row is a pair of adjacent layers in BERT-Base and each column is an attention head. Instead of computing one scalar value for each head pair, we show the full distribution based on the tokens in 8,192 held-out examples, *i.e.*, each data point is the JSD between the attention probabilities of a token at these two heads. For better legibility, we color code these plots to help distinguish head pairs with relatively "similar" attention (BLUE: median < 0.25) and relatively "distinct" attention (RED: median > 0.75) from the rest (YELLOW: $0.25 \leq$ median ≤ 0.75).

Note that JSD results from Post-LN are used only as a reference; we expect them to be "random" because there is no correspondence between heads in adjacent layers for Post-/Pre-LN. Proof: An equivalent Post-/Pre-LN can be constructed by permuting the order of attention heads in a layer (and the corresponding variables).

[15] On WikiHop, number of fine-tuning epochs is selected from {5, 10, 15} instead of {5, 10} for both ETC-Large and our model. We added 15 here following the official ETC repository.

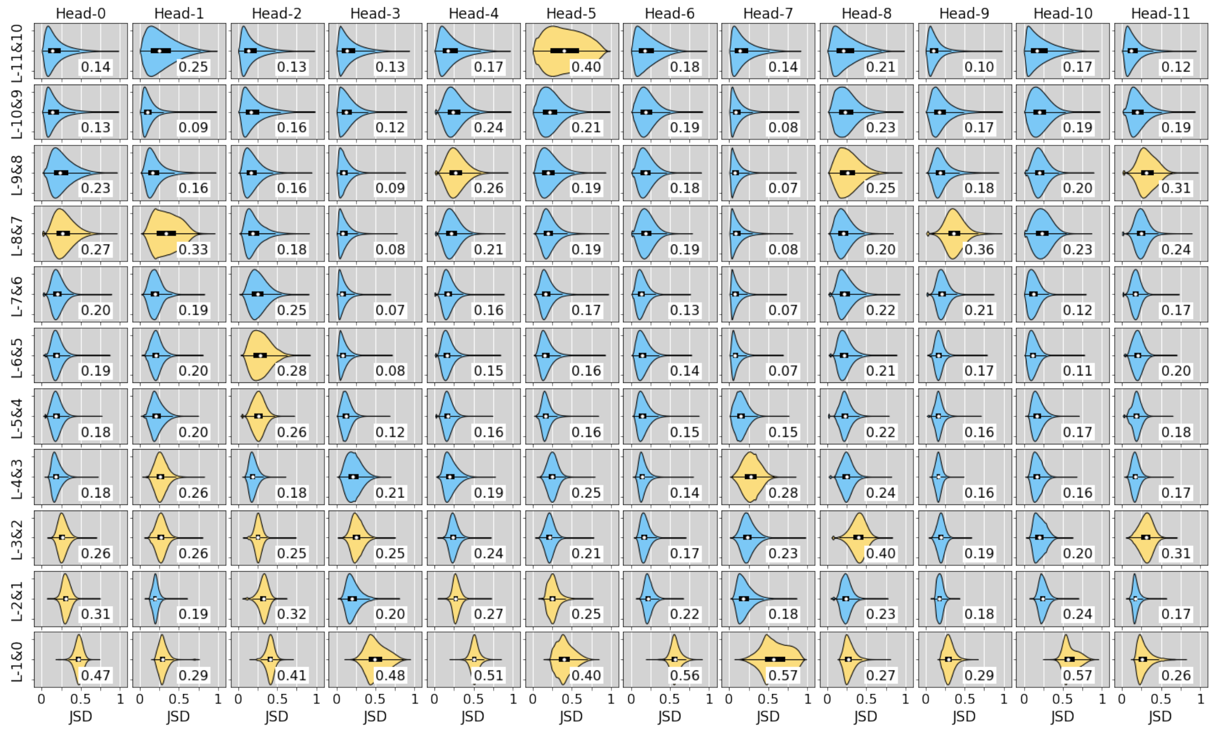


(a) Post-LN

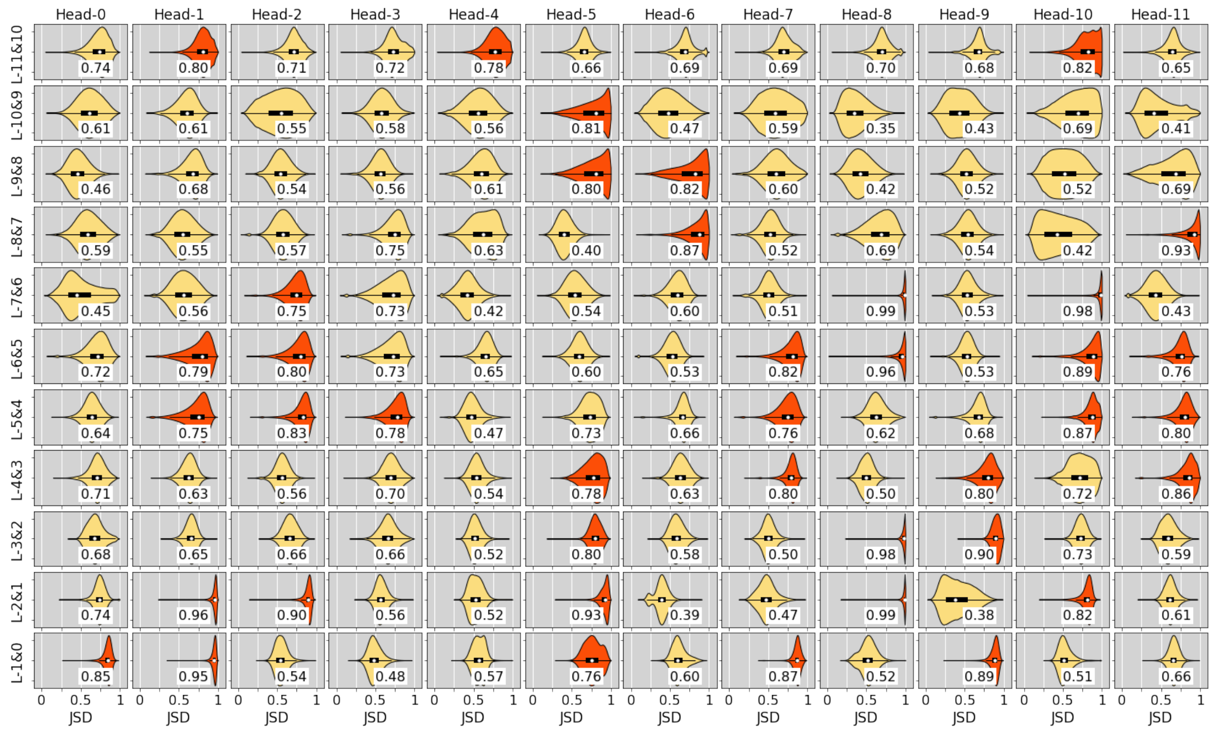


(b) Pre-LN

Figure 5: Distribution of entropies of the attention probabilities of the tokens of 8,192 held-out examples using the pre-trained BERT-Base with **Post-LN** and **Pre-LN** Transformer respectively (see Section 4.1.1). For better legibility, (1) attention heads in each layer are ordered by their medians of entropies, and (2) distributions are color-coded based on the median of entropies: RED (median > 4.5), YELLOW ($1.5 \leq$ median ≤ 4.5), BLUE (median < 1.5), *i.e.*, colder colors mean sparser attention. Note that here top layers (layer 9-11) tend to have larger entropies compared to RealFormer, which means that attention is relatively *denser*.



(a) RealFormer



(b) Post-LN

Figure 6: Distribution of Jensen-Shannon Divergence (JSD) of attention probabilities in (vertically) adjacent attention heads, *i.e.*, JSD($\text{head}_i^L, \text{head}_i^{L-1}$). Based on 8,192 held-out examples using the pre-trained BERT-Base with **RealFormer** and **Post-LN** Transformer respectively (see Section 4.1.1). Distributions are color-coded based on the median of JSDs: RED (median > 0.75), YELLOW ($0.25 \leq$ median ≤ 0.75), BLUE (median < 0.25). *I.e.*, colder color means more "similar" attention heads across adjacent layers.