Multi-head or Single-head?
An Empirical Comparison for Transformer Training

Anonymous ACL submission

Abstract

Multi-head attention plays a crucial role in the recent success of Transformer, which leads to consistent performance improvements over conventional attention in various applications. The popular belief is that its effectiveness stems from the ability to attend multiple positions jointly. In this paper, we first demonstrate that jointly attending multiple positions is not a unique feature of *multi-head* attention, as *multi-layer single-head* attention also attends multiple positions. Then, we suggest the main advantage of the multi-head attention is the training stability, since it has fewer layers than the single-head attention when attending the same number of positions. Meanwhile, we show that, with recent advances in deep learning, we can successfully stabilize the training of the deep single-head Transformer. As the training difficulty is no longer a bottleneck, substantially deeper single-head Transformers achieve consistent performance improvements.

1 Introduction

Transformers (Vaswani et al., 2017) have led to a series of breakthroughs in various deep learning tasks (Devlin et al., 2019; Velickovic et al., 2018b). One distinguishing characteristic of Transformer is that it does not contain any recurrent connections and can parallelize all computations in the same layer, thus leads to better effectiveness, efficiency, and scalability. Without using recurrent connections, Transformer purely relies on the attention mechanism to capture the dependency among input tokens. Specifically, a multi-head attention module was proposed and used in Transformer to better capture the dependency among input tokens.

This multi-head attention module has been observed to be one major reason behind the success of the Transformer. For example, on machine translation benchmarks, Recurrent Neural Networks (RNNs) can outperform Transformers when both are using the multi-head encoder-decoder attention and would underperform without using the multi-head attention (Chen et al., 2018). Besides Transformer, multi-head attention has also been incorporated into other models (Chen et al., 2018; Velickovic et al., 2018a; Fang et al., 2019). More discussions on related work is available at Appendix A.1.

Multi-head attention was proposed to jointly attend multiple positions, while conventional attention can only attend one position in one layer. Specifically, multi-head attention projects the inputs into multiple different subspaces and conducts multiple attention computations in parallel.

Our Contributions. Our point of start is demonstrating that attending multiple positions is not a unique feature of multi-head attention. In fact, stacking multiple conventional attention modules can also attend multiple positions.

Specifically, as in Figure 1, a multi-head attention module can be viewed as an ensemble model, which combines multiple single-head attention modules by calculating their average. Thus, by integrating these modules differently, we can reconstruct a Transformer to be single-head[1] and substantially deeper. These two networks can attend the same number of places (i.e., have the same total number of attention heads), have roughly the same number of parameters and inference computation complexity, while the multi-head one is shallower and the single-head one is deeper.

In our experiments, we observe that, compared to the shallower multi-head Transformer, the deeper single-head Transformer performs better but is harder to train, which matches the common wisdom that model depth can increase model capacity at the cost of training difficulty. Also, we observe that, benefited from the recent advance of deep learning (Liu et al., 2020b), the training difficulty is no longer an obstacle.

[1]We use single-head/multi-head Transformer to refer Transformer with single-head/multi-head Attention.

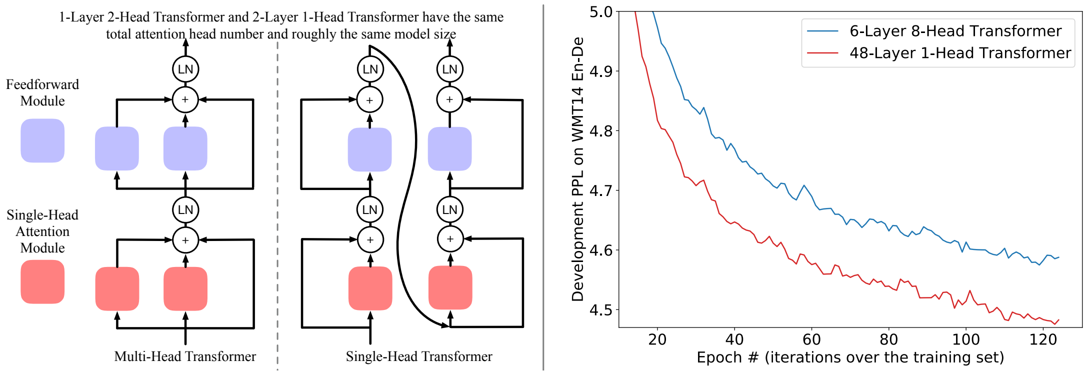

Figure 1: Left: both multi-head and single-head Transformer can attend multiple positions. Right: comparing to the shallow multi-head Transformer, the deep single-head Transformer has the potential to achieve a lower PPL score, while its training is more challenging (without Admin, the 48-layer 1-head Transformer training diverged).

2 Multi-Head and Single Head

Intuitively, with the same set of modules, no matter how these modules are integrated, the model can attend the same number of places. Still, some module integration strategies could be more effective in integrating modules.

In the original multi-head Transformer, modules in the same layer are combined in an ensemble manner and cannot enhance each other (more elaborations are included in Appendix A.3). For example, as in Figure 1, when constructed in a multi-head manner, the two attention heads would have the same input and are agnostic to each other. In this way, the second attention head cannot leverage or benefit from the information captured by the first attention head.

Intuitively, it could be beneficial to allow the second attention head to stand on the shoulders of the first attention head. To this end, we integrate these modules differently and reconstruct the shallow multi-head Transformer into the deep single-head Transformer (as in Figure 1). Note that both models have the same total number of attention heads, roughly the same model size, and roughly the same inference computation complexity.

3 Stability Comparison

As in Table 1, after changing the shallow multi-head Transformer to the deep single-head Transformer, the training fails to converge well for 2 out of 3 models. Note that, although the 1H-144L BERT-base model converges successfully, the model is sensitive to the choice of initialization. Specifically, the BERT-base model and BERT-large model are initialized with truncated normal distribution with 0.02 variance, instead of following the common practice (e.g., using the Kaiming initialization (He et al., 2015) or the Xavier initialization (Glorot and Bengio, 2010)). We observe that after changing the variance of the initialization, or following the common practice, the training of the 1H-144L BERT-base model would also fail.

Meanwhile, we show that, with the recent advances in deep learning, the training can be successfully stabilized by Adaptive Model Initialization (Admin), without changing any hyper-parameters (Liu et al., 2020b). Also, after employing the Admin initializatioin, the 1H-144L BERT-base model can be trained successfully when following the standard Xavier initialization. This shows that, although the deep single-head Transformer is harder to train, the training difficulty is no longer an obstacle.

4 Performance Comparison

For machine translation, we summarize the model performance in Table 2. With the same model size, the deep single-head Transformer (1H-48L-48L) achieves a 0.5 BLEU improvements over the shallow multi-head Transformer. Also, the deep single-head Transformer achieves the same performance with the architecture search algorithm the Evolved Transformer (So et al., 2019) and DARTSformer (Zhao et al., 2021), with slightly less parameters. Specifically, Evolved Transformer and DARTSformer conducts neural architecture search on Transformer, and treat the multi-head attention as the basic module (i.e., the deep single-head Transformer is not in their search space). Our deep single-head Transformer achieves comparable performance without hyper-parameter tuning, which further verifies its effectiveness.

	Transformer-base		BERT-base		BERT-large	
	8H-6L-6L	1H-48L-48L	12H-12L	1H-144L	16H-24L	1H-384L
Training	✓	× /✓(w. Admin)	✓	✓	✓	×/✓(w. Admin)

Table 1: Deep single-head Transformers are harder to train than shallow multi-head Transformers.

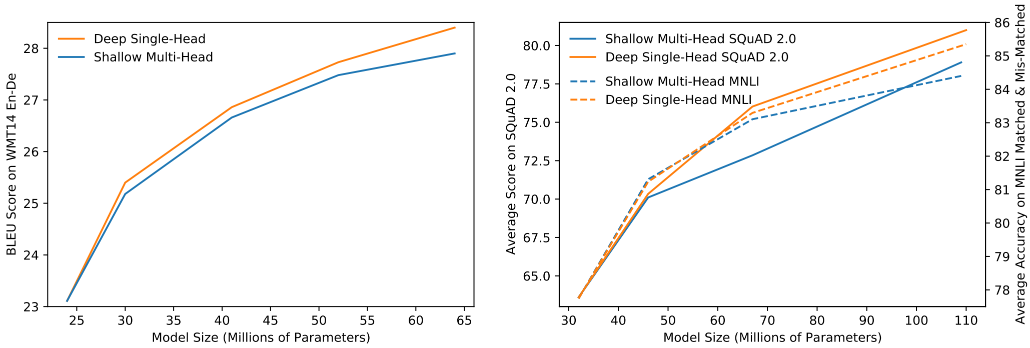

Figure 2: Performance with Different Model Size. Left: the performance of αH-6L-6L (α=1, 2, 4, 6, 8) and 1H-βL-βL (β=6,12,24,36,48), whose per-head dimension is the same with Transformer-base. Right: the performance of αH-12L (α=1, 3, 6, 12) and 1H-βL (β=12,36,72,144), whose per-head dimension is the same with BERT-base.

Model	BLEU	Param.
8H-6L-6L	27.90	63.2M
1H-48L-48L	28.40	63.6M
2D-CSANs(Yang et al., 2019)	28.18	88.0M
Evolved*(So et al., 2019)	28.4	64.1M
DARTSformer*(Zhao et al., 2021)	28.4	65.2M

Table 2: Performance on the WMT'14 EN-DE dataset. * indicates neural architecture search methods.

As in Table 3, the deep single-head Transformer achieves consistent performance improvements over the original shallow multi-head Transformer. Table 4 shows the test performance on the GLUE benchmark. The deep single-head Transformer outperforms the shallow multi-head Transformer on 7 out of 9 tasks, and improves the average score (GLUE) by roughly 1 point. In the mean time, it is worth mentioning that, on 2 out of 3 sentence similarity/paraphrase tasks, the shallow multi-head Transformer achieves better performance. This indicates the deep single-head Transformer can be further improved, and we will further explore this in the future work. These observations verified that the deep single-head Transformer could perform better than the shallow multi-head Transformer.

Impact of Model Initialization. Here, we aim to understand the impact of model initialization on model performance. As the 1H-144L BERT-base model converges well with both the vanilla initialization and the Admin initialization, we not only conduct training with the Admin initialization, but also the vanilla initialization. As summarized in Table 3, the default initialization and the Admin initialization achieve similar performance. This observation supports our intuition that the major benefit of the Admin initialization is on training stability, and the performance improvements mostly come from the change from shallow multi-head Transformer to deep single-head Transformer.

Impact of Head Number. Intuitively, the difference between deep single-head Transformers and shallow multi-head Transformers is proportional to the model size/head number (e.g., the difference between 2H-6L and 1H-12L should be smaller than the difference between 4H-6L and 1H-24L). We conduct experiments on Transformers with different head numbers, and visualize the results in Figure 2. It shows that when the architecture difference is between shallow multi-head Transformer and deep single-head Transformer is larger (i.e., with more number of heads), the performance improvement is also larger.

5 Efficiency Comparison

Inference Speed. The shallow multi-head Transformer and the deep single-head Transformer have roughly the same model size and computation complexity. Here, we calculated the average inference speed on an idle RTX 3060 GPU[2]. We find that,

[2]We used the FasterTransformer (version 4.0) as in

	FLOPs #	Param. #	MNLI Acc.		SQuAD v2.0	
			match	mis-match	exact match	F1
12H-12L BERT$_{\text{BASE}}$	46.3B	109.5M	84.4	84.4	77.4	80.4
1H-144L BERT$_{\text{BASE default}}$	46.9B	110.0M	85.6	85.1	79.6	82.4
1H-144L BERT$_{\text{BASE Admin}}$	46.9B	110.0M	85.2	85.4	79.2	82.5
16H-24L BERT$_{\text{LARGE}}$	161.8B	335.1M	86.3	86.4	81.0	84.3
1H-384L BERT$_{\text{LARGE Admin}}$	164.1B	337.4M	87.7	87.5	82.6	85.7

Table 3: The model performance on dev sets of MNLI and SQuAD 2.0. The FLOPs are calculated for the inference computation of one 512-length input sequence.

	GLUE	CoLA	SST-2	MRPC	SST-B	QQP	MNLI-m/mm	QNLI	RTE	WNLI
12H-12L	78.3	52.1	93.5	88.9/84.8	**87.1/85.8**	**71.2/89.2**	84.6/83.4	90.5	66.4	**65.1**
1H-144L	79.4	**59.2**	**94.2**	**89.3/85.4**	84.3/83.5	70.9/88.9	**85.1/84.3**	**91.0**	**69.0**	**65.1**
16H-24L	80.5	60.5	94.9	89.3/85.4	**87.6/86.5**	**72.1/89.3**	86.7/85.9	92.7	70.1	**65.1**
1H-384L	81.3	**62.7**	**95.1**	**90.5/87.2**	86.9/86.3	71.3/89.1	**87.4/86.5**	**93.3**	**72.7**	**65.1**

Table 4: The test performance on the GLUE benchmark with metrics described in Table 5.

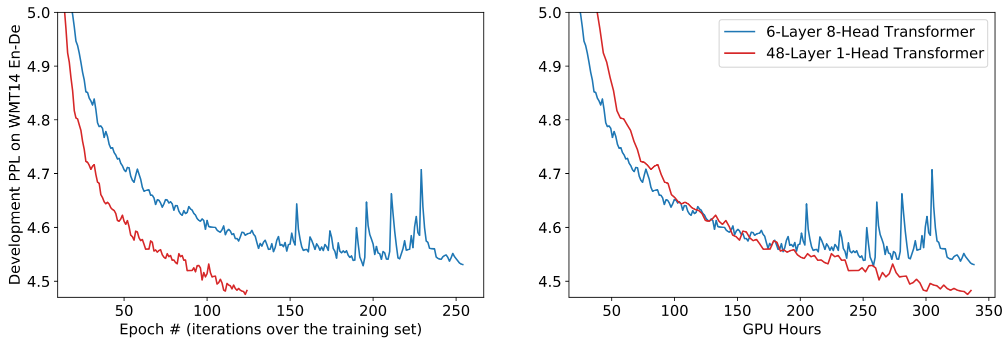

Figure 3: Transformer Training Efficiency (GPU Hours are calculated on an idle RTX 3060).

with an optimized implementation, the inference efficiency of the shallow multi-head Transformer and the deep single-head Transformer are roughly the same (visualized in Appendix, Figure 4).

Training Speed. As in Figure 3, we can find that the training computation speed of the 1H-48L-48L Transformer is about two times slower than the 8H-6L-6L Transformer. Meanwhile, the 8H-6L-6L Transformer converges faster with regard to epoch number, or GPU hours. This phenomenon verifies our intuition that the network depth of the 6-Layer Transformer has become a bottleneck of the model capacity, which restricts the model performance.

6 Conclusion

Here, we focus on understanding the effectiveness of the multi-head Transformer. We first show that deep single-head Transformer also attends multiple positions and performs better than the popular shallow multi-head Transformer. Then, we suggest the main advantage of multi-head attention is the training stability since it has fewer layers than the single-head attention when attending the same number of positions. We also show that, with recent advances in deep learning, the training stability is no longer an obstacle and it can lead to consistent performance improvements by turning shallow single-head Transformer into deep multi-head Transformer.

Our work opens up new possibilities to not only further push the state-of-the-art but understand the effectiveness of Transformer better. It leads to various interesting future work. For example, intuitively, both shallow multi-head Transformer and deep single-head Transformer should not be the optimal architecture, and neural architecture search can be employed to find a good balance between multi-head and single-head.

https://github.com/NVIDIA/FasterTransformer

References

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv*, abs/1607.06450.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Michael Schuster, Zhi-Feng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. 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-HLT*.

Yong Fang, J. Gao, C. Huang, H. Peng, and R. Wu. 2019. Self multi-head attention-based convolutional neural networks for fake news detection. *PLoS ONE*, 14.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

A. Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *ArXiv*, abs/1410.5401.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv: Learning*.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2019. Music transformer: Generating music with long-term structure. In *ICLR*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *ArXiv*, abs/1703.03130.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020a. On the variance of the adaptive learning rate and beyond. In *ICLR*.

Liyuan Liu, X. Liu, Jianfeng Gao, Weizhu Chen, and J. Han. 2020b. Understanding the difficulty of training transformers. In *EMNLP*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *ArXiv*, abs/1508.04025.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *NeurIPS*.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. In *IWSLT*.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *ICML*.

Hao Peng, Roy Schwartz, Dianqi Li, and Noah A. Smith. 2020. A mixture of h-1 heads is better than h heads. In *ACL*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. 2019. Stand-alone self-attention in vision models. In *NeurIPS*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL-HLT*.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Petar Velickovic, Guillem Cucurull, A. Casanova, Adriana Romero, P. Lio', and Yoshua Bengio. 2018a. Graph attention networks.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018b. Graph attention networks. In *ICLR*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019. Learning deep transformer models for machine translation. In *ACL*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shu xin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Li-Wei Wang, and Tie-Yan Liu. 2019. On layer normalization in the transformer architecture. *ArXiv*, abs/2002.04745.

Baosong Yang, Longyue Wang, Derek Wong, Lidia S Chao, and Zhaopeng Tu. 2019. Convolutional self-attention networks. *arXiv preprint arXiv:1904.03107*.

Yuekai Zhao, Li Dong, Yelong Shen, Zhihua Zhang, Furu Wei, and Weizhu Chen. 2021. Memory-efficient differentiable transformer architecture search. *ArXiv*, abs/2105.14669.

A Appendix

A.1 Related Work

There exist two aspects of related work regarding the topic here, *i.e.*, Attention and Transformer.

Attention and Multi-Head Structure. Attention modules are first proposed to capture the long-term dependency in sequence-to-sequence models (Graves et al., 2014; Bahdanau et al., 2015). To calculate the output for a token in the target sequence, the attention module would calculate a weighted average of source token representations, while the weight is calculated by applying softmax on attention scores. Different variants of attention modules calculate attention scores differently. For example, to calculate the attention score, Graves et al. (2014) uses the cosine similarity, Bahdanau et al. (2015) uses the perception network, and Luong et al. (2015) uses dot product. While these modules can only attend one position in one layer, attempts like multi-head attention have been made to jointly attend multiple positions (Lin et al., 2017; Vaswani et al., 2017), which is identified as one major reason behind the success of Transformer (Chen et al., 2018). Also, it has inspired several follow-up studies to analyze the multi-head structure (Michel et al., 2019; Peng et al., 2020). Specifically, Michel et al. (2019) observes single-head Transformer performing better than multi-head Transformer for model pruning. Still, no study has been conducted on deep single-head Transformer training, due to its training difficulty.

Transformer. Transformer (Vaswani et al., 2017) has led to a series of breakthroughs in various domains (Devlin et al., 2019; Velickovic et al., 2018b; Huang et al., 2019; Parmar et al., 2018; Ramachandran et al., 2019). Meanwhile, Transformer training has been found to be more challenging and attracted lots of attention to analyze why Transformer is harder to train and how to stabilize Transformer training (Liu et al., 2020a; Baevski and Auli, 2019; Nguyen and Salazar, 2019; Wang et al., 2019; Xiong et al., 2019; Liu et al., 2020b). Many efforts have been made to improve Transformer, e.g., relative position encoding (Shaw et al., 2018) or replacing dot-product attention with locality-sensitive hashing (Kitaev et al., 2020). Here, we choose to focus our study on the original Transformer model as proposed in Vaswani et al. (2017), and uses the initialization technique Admin to stabilize model training (Liu et al., 2020b), since this method does not include any additional hyper-parameters and its final model is equivalent to the original Transformer.

A.2 Transformer Architecture

The Transformer architecture contains two types of sub-layers, i.e., Attention sub-layers and Feedforward sub-layers. Each sub-layer is constructed with the shortcut connection and the Layer Norm. Specifically, it calculates the output as $\mathbf{x}_{i+1} = f_{\text{LN}}(\mathbf{x}_i + f(\mathbf{x}_i))$, where $\mathbf{x}_i$ is the input of layer i and the output of layer $i-1$ (top layers have larger indexes), f_{LN} is the Layer Norm , and $f(\cdot)$ is multi-head attention $f_{\text{ATT}}(\cdot)$ or feedforward $f_{\text{FFN}}(\cdot)$ for Attention sub-layers and Feedforward sub-layers respectively.

Layer Norm. Layer norm (Ba et al., 2016) plays a vital role in the Transformer architecture. It is defined as $f_{\text{LN}}(\mathbf{x}) = \boldsymbol{\gamma}\frac{\mathbf{x}-\mu}{\sigma} + \boldsymbol{\nu}$, where μ and σ are the mean and standard deviation of $\mathbf{x}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\nu}$ are learnable parameters.

Feedforward. Transformers use two-layer perceptrons as feedforward networks, *i.e.*, $f_{\text{FFN}}(\mathbf{x}) = \phi(\mathbf{x}W^{(1)})W^{(2)}$, where $W^{(\cdot)}$ are parameters, and $\phi(\cdot)$ is the non-linear function. Specifically, the original Transformer ReLU as the activation function, while later study uses other types of activation function, e.g., BERT uses GELU as the activation function (Hendrycks and Gimpel, 2016).

Attention. Transformers use the multi-head attention to capture the dependency among input tokens, which is based on the scaled dot-product attention. Scaled dot-product attention tries to query information from the source sequence that is relevant to the target sequence. Specifically, assuming the length of the source sequence and the target sequence to be n and hidden dimension to be m, the target sequence would be encoded as $Q \in R^{n\times m}$, source sequence would be encoded as $K \in R^{n\times m}$ and $V \in R^{n\times m}$. The scaled dot-product attention would calculate the output as $f_{\text{Scaled Dot-Product Attention}}(Q, K, V) =$

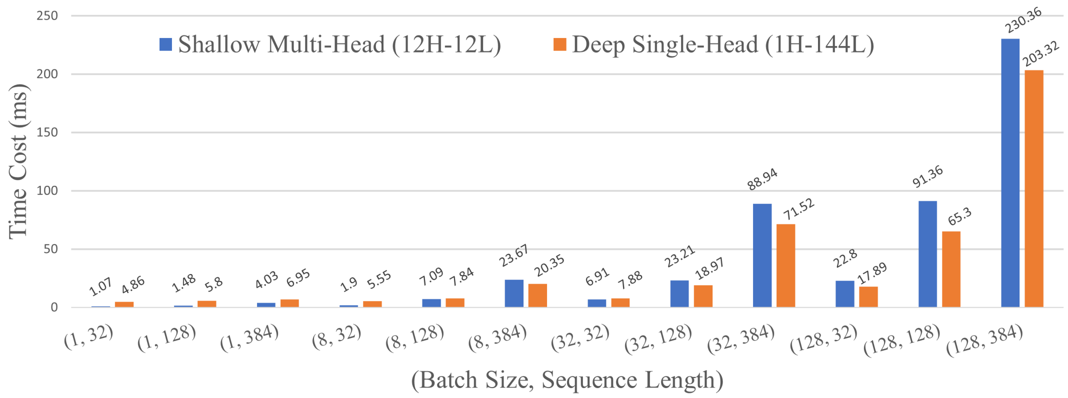

Figure 4: The Inference Speed of BERT-base with Different Batch Size and Sequence Length.

$\text{softmax}(\frac{QK^T}{\sqrt{m}})V$, where $\text{softmax}(\cdot)$ is the row-wise softmax.

One scaled dot-product attention is believed to attend only one position in each row (for each target token), since the output of softmax typically would have one dimension significantly larger than other dimensions in each row. Multi-head attention was proposed to jointly attend multiple positions, which employs multiple scaled dot-product attention in parallel. Specifically, it calculates the output as $f_{\text{ATT}}(Q, K, V) = [\text{head}_1; \cdots; \text{head}_h]W^{(O)}$, where $\text{head}_i = f_{\text{Scaled Dot-Product Attention}}(QW_i^{(Q)}, KW_i^{(K)}, VW_i^{(V)})$, $W^{(\cdot)}$ are learnable parameters, and h is the number of heads.

Transformer. Transformer has two types of layer configurations when serving as the encoder and the decoder respectively. Here, we use $\mathbf{x}_i$ to mark the input of sub-layer i. Each Transformer encoder layer contains two sub-layers, i.e., one attention sub-layer in a self-attention manner and one feedforward sublayer. Specifically, the attention sub-layer calculates outputs as $\mathbf{x}_{2i+1} = f_{\text{LN}}(\mathbf{x}_{2i} + f_{\text{ATT}}(\mathbf{x}_{2i}, \mathbf{x}_{2i}, \mathbf{x}_{2i}))$ and the feedforward sub-layer calculates outputs as $\mathbf{x}_{2i+2} = f_{\text{LN}}(\mathbf{x}_{2i+1} + f_{\text{FFN}}(\mathbf{x}_{2i+1}))$. Notice that the attention sub-layer sets Q, K, and V as the same value $\mathbf{x}_{2i}$, capturing the dependency among tokens within the same sequence, which is referred to as self-attention.

Each Transformer decoder layer contains three sub-layers, besides the self-attention sublayer and the feedforward sublayer, it also includes an encoder-decoder attention sub-layer between them. Specifically, the encoder-decoder attention sub-layer calculates outputs as $\mathbf{x}_{3i+2} = f_{\text{LN}}(\mathbf{x}_{3i+1} + f_{\text{ATT}}(\mathbf{x}_{3i+1}, \mathbf{h}, \mathbf{h})$, where K and V are set to the encoder output $\mathbf{h}$.

A.3 Implicit Ensemble Structure

As in Figure 1, multi-head attention sub-layers and feedforward sub-layers have the implicit ensemble structure, i.e., each of these sub-layers can be viewed as an ensemble of smaller models. Now let us proceed to introduce those parallel structures in detail. Notations are introduced in Section A.2.

Attention. We split the weight matrix $W^{(O)}$ into h parts by rows, i.e., we mark $W^{(O)} = [W_1^{(O)^T}; \cdots; W_h^{(O)^T}]^T$. Then, the multi-head attention calculates outputs as:

$$
\begin{aligned}
& f_{\text{ATT}}(Q, K, V) \\
=& [\text{head}_1; \cdots; \text{head}_h]W^{(O)} = \sum_{i=1}^{h} \text{head}_i W_i^{(O)} \\
=& \sum_{i=1}^{h} \text{softmax}(\frac{QW_i^{(Q)} W_j^{(K)^T} K^T}{\sqrt{m}}) VW_i^{(V)} W_i^{(O)}
\end{aligned}
$$

Note that each head can be viewed as a low-rank version of the general attention (Luong et al., 2015).

Thus, the multi-head attention can be viewed as jointly attending multiple places by ensembling multiple conventional attention modules. Specifically, the general attention module (Luong et al., 2015) calculates outputs as:

$$
f_{\text{General Attention}}(Q, K, V) = \text{softmax}(QW_1K^T)VW_2
$$

Comparing f_{ATT} and $f_{\text{General Attention}}$, we can find their major difference is that the multi-head attention decomposes the $m \times m$ matrix W_1 and W_2 into $\frac{W_i^{(Q)} W_i^{(K)^T}}{\sqrt{m}}$ and $W_i^{(V)} W_i^{(O)}$, where $W_i^{(Q)}, W_i^{(K)}, W_i^{(V)}, W_i^{(O)^T} \in R^{m \times \frac{m}{h}}$. With this low-rank decomposition, the parameter number and computation complexity of the multi-head attention module would stay the same no matter what the value of h is (i.e., how many heads one layer has).

Feedforward. Similar to the Attention module, we can also rewrite the Feedforward sub-layer as an ensemble of h modules.[3] Specifically, we split the weight matrix $W^{(1)}$ into h parts by rows and $W^{(2)}$ into h parts by columns, i.e., we mark $W^{(1)} = [W_1^{(1)}; \cdots; W_h^{(1)}]$ and $W^{(2)} = [W_1^{(2)^T}; \cdots; W_h^{(2)^T}]^T$. Then, the feedforward sub-layer calculates outputs can be rewrote as:

$$f_{\text{FFN}}(\mathbf{x}) = \phi(\mathbf{x}W^{(1)})W^{(2)} = \sum_{i=1}^{h} \phi(\mathbf{x}W_i^{(1)})W_i^{(2)}$$

Thus, the Feedforward sub-layer can be viewed as an ensemble of h sub-modules. Note that since the sum of the h sub-modules would be normalized by Layer Norm, their outputs are integrated in an averaging manner.

Average Ensemble. Each Transformer sub-layer calculates outputs as $f_{\text{LN}}(\mathbf{x} + f(\mathbf{x}))$, where $f(\cdot)$ could be $f_{\text{FFN}}(\cdot)$ and $f_{\text{ATT}}(\cdot)$. Thus, the sum calculated in the computation of f_{ATT} and f_{FFN} would be normalized by $\text{Var}[\mathbf{x} + f(\mathbf{x})]$. In this way, the joint effect of layer norm and the sum would be similar to combining these modules in an average ensemble manner.

A.4 Experiment Setup

In our experiments, we adopt hyper-parameter settings from previous work (Liu et al., 2020b; Devlin et al., 2019), and more experiment details can be found in the appendix.

Transformer Model Configurations. We conduct experiments with three Transformer models, i.e., Transformer-base for the WMT'14 EN-DE translation task, BERT-base, and BERT-large for the language model pre-training. Specifically, the original Transformer-base model is 8H-6L-6L[4], and we compare it with 1H-48L-48L. The original BERT-base and BERT-large models are 12H-12L and 16H-24L, and we compare them with 1H-144L and 1H-384L. We use the Admin initialization (Liu et al., 2020b) to stabilize 1H-48L-48L Transformer-base and 1H-384L BERT-large. More detailed configurations are included in the appendix.

Translation. Here, we conduct experiments on WMT'14 EN-DE and evaluate model performance based on their BLEU score on the test set and perplexity score on the development set.

BERT. Here, we follow the training setting from Devlin et al. (2019) and evaluate pre-trained language models on the SQuAD 2.0 (Rajpurkar et al., 2018) datasets for question answering, and the GLUE benchmark (Wang et al., 2018), which includes 9 subtasks (as in Table 5).

A.5 Transformer Model Configurations

For machine translation, the original Transformer-base model is 8H-6L-6L Transformer encoder-decoder with 512-dimension word embedding, 64-dimension per-head attention output, and 2048-dimension feedforward network (Vaswani et al., 2017). Here, we compare it with 1H-48L-48L Transformer encoder-decoder with 512-dimension word embedding, 64-dimension per-head attention output, and 256-dimension feedforward network. For language model pre-training, BERT-base model is 12H-12L Transformer encoder with 768-dimension word embedding, 64-dimension per-head attention output, and 3072-dimension feedforward network; BERT-large model is 16H-24L Transformer encoder with 1024-dimension word embedding, 64-dimension per-head attention output, and 4096-dimension feedforward network (Devlin et al., 2019). Here, we compare them with deep single-head BERT-base model (1H-144L Transformer encoder with 768-dimension word embedding, single-head 64-dimension per-head attention output, and 256-dimension word embedding) and deep single-head BERT-large model (1H-384L Transformer encoder with 768-dimension word embedding, 64-dimension per-head attention output, and 256-dimension word embedding). To stabilize 1H-48L-48L Transformer-base and 1H-384L BERT-large, we use the Admin initialization (Liu et al., 2020b).

A.6 Implementation Detail

Besides the layer number and head number, we adopted all hyper-parameters from previous work. Specifically, we followed (Liu et al., 2020b) for machine translation experiments and (Devlin et al., 2019) for language model pre-training experiments.

[3] Note h here is decided to be consistent with the Multi-Head Attention sub-layers.

[4] We use "γH-αL(-βL)" to denote that a model has γ-head α-layer encoder and γ-head β-layer decoder.

Corpus	|Train|	|Label|	Task	Metric(s)	Domain
			Single-Sentence Classification		
CoLA	8.5k	2	acceptibility	Matthews corr.	misc.
SST-2	67k	2	sentiment	accuracy	movie reviews
			Sentence Similarity/Paraphrase		
MRPC	3.7k	2	paraphrase	accuracy/F1	news
STS-B	5.7k	-	similarity	Pearson/Spearman corr.	misc.
QQP	364k	2	similarity	accuracy/F1	social QA questions
			Natural Language Inference (NLI)		
MNLI	393k	3	NLI	(mis)matched acc.	misc.
QNLI	108k	2	QA/NLI	accuracy	Wikipedia
RTE	2.5k	2	NLI	accuracy	misc.
WNLI	634	2	coreference/NLI	accuracy	fiction books

Table 5: GLUE task descriptions and statistics. The second and fourth column denotes the number of training examples and the number of classes. Note that STS-B is a regression task.

It is worth mentioning that, in (Liu et al., 2020b), the default initialization method is the Xavier initialization (Glorot and Bengio, 2010), which depends on the size of the weight matrix. Here, to control variables, we fix the initialization scale to be the same with original multi-head shallow Transformer. Meanwhile, for language model pre-training, since (Devlin et al., 2019) fixes the initialization scale for all models, we directly adopt the initialization strategy without modification.

A.7 Training Detail

For machine translation experiments, we followed (Liu et al., 2020b) to conduct data pre-processing, conduct model training on Nvidia GPUs (including Quadro RTX 8000, GeForce RTX 3060, and Quadro RTX A6000). As to language model pre-training experiments, we followed (Devlin et al., 2019) to conduct data pre-processing, conduct model training with Google TPU v3.