Deeper vs Wider: A Revisit of Transformer Configuration

Fuzhao Xue\(^1\) Jianghai Chen\(^1\) Aixin Sun\(^2\) Xiaozhe Ren\(^3\) Zangwei Zheng\(^1\)
Xiaoxin He\(^1\) Xin Jiang\(^3\) Yang You\(^1\)

\(^1\)Department of Computer Science, National University of Singapore
\(^2\)School of Computer Science and Engineering, Nanyang Technological University
\(^3\)Huawei Noah’s Ark Lab

Abstract

Transformer-based models have delivered impressive results on many tasks, particularly vision and language tasks. In many model training situations, conventional configurations are typically adopted. For example, we often set the base model with hidden dimensions (i.e., model width) to be 768 and the number of transformer layers (i.e., model depth) to be 12. In this paper, we revisit these conventional configurations. Through theoretical analysis and experimental evaluation, we show that the masked autoencoder is effective in alleviating the over-smoothing issue in deep transformer training. Based on this finding, we propose Bamboo, an idea of using deeper and narrower transformer configurations, for masked autoencoder training. On ImageNet, with such a simple change in configuration, re-designed model achieves 87.1% top-1 accuracy and outperforms SoTA models like MAE and BEiT. On language tasks, re-designed model outperforms BERT with default setting by 1.1 points on average, on GLUE datasets.

1 Introduction

Transformer-based language models have achieved promising results on natural language understanding tasks, e.g., Q&A [17, 30], relation extraction [29, 35] and dialogue system [16]. Recently, on vision tasks, transformers [9, 27, 28, 33] also outperform convolution-based models by a large margin. With sufficient training data, transformer-based models can be scaled to trillions of trainable parameters [10, 11]. Through scaling along width (i.e., hidden dimension) and depth (i.e., number of transformer blocks), these huge transformers show effectiveness across various tasks and even areas.

When we use transformer, we typically follow existing work to set width and depth. For instance, we usually set the width of transformer-base model as 768 and the depth as 12. An interesting question here is: Why do we select these fixed hyper-parameters, even for problems in different areas? To answer this question, we revisit the conventional configurations from representative studies. For vision transformer [9], authors set the base ViT configuration according to those used in BERT [7], BERT selects such configuration following OpenAI GPT [18]. OpenAI also follows the original transformer paper [23]. In the original transformer paper, Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin [23] conduct a set of ablation studies on machine translation task to find the optimal configurations. That is, for a good range of tasks, we have largely follow the transformer configuration based on an ablation study on machine translation task, i.e., a sequence-to-sequence task.

Nowadays, transformer-based models can be trained with various different training objectives or strategies. Taking vision transformer [9] as an example, we can train transformer from scratch with a supervised learning setting for image classification. In this straightforward image classification task, each image is modeled as a sequence of multiple tokens, and each token corresponds to a patch in
the image. We use the global information (from all tokens/patches of the image) to predict a single label, the category of the image. Here, as the training objective is to capture the global information of an image, the differences between token representations would not be considered directly. This image classification task is quite different from machine translation task, which requests for strong understanding of each input token and generating another sequence. Hence, it is natural to assume that different optimal transformer configurations exist for these two different tasks.

Previous work has tried to train a deeper transformer from scratch. However, as reported in [12, 33], training by classification task (i.e., using the global signal of the input sequence) has the over-smoothing problem. That means, at the deeper transformer layers, all token representations tend to be identical. Such an issue harms the scalability of training vision transformer, especially scaling along depth. When we scale to a larger model, we only get a slight improvement or even worse accuracy. Recently, Gong, Wang, Li, Chandra, and Liu [12] and Zhou, Kang, Jin, Yang, Lian, Jiang, Hou, and Feng [33] show that, when we add special-designed regularization to avoid the “uniform tokens” (i.e., the over-smoothing problem), it is possible to train deeper transformer on the sequence (image) classification setting.

Different from training from scratch, the masked autoencoder is a two-stage training framework, including pre-training and fine-tuning. Given partially masked input sequence, the pre-training stage aims to recover the original unmasked sequence. The fine-tuning stage is similar to the aforementioned training from scratch. With masked autoencoder, recent studies [1, 13] successfully train large scale transformers, even without using additional training data compared with supervised learning. This is counterintuitive because we usually assume more training data is the key in self-supervised pre-training to improve effectiveness. However, masked autoencoder is able to scale transformer to larger or deeper without the need of bigger dataset. This result motivates us to rethink the reason behind it.

We hypothesize that masked autoencoder training can alleviate the over-smoothing issue, which is an important reason why masked autoencoder can help to scale transformer up. The main reason behind this is the different training objectives. In masked autoencoder frameworks (e.g., BERT, BEiT), the target is to recover the masked tokens according to the unmasked tokens. Compared with training transformer from scratch under supervised learning, whose target is a simple classification task, masked autoencoder framework is using a sequence labeling target. The sequence labeling task requires the model to learn semantic information from neighboring unmasked tokens. Since different masked tokens have different unmasked neighboring tokens, the unmasked token representations must carry sufficient semantics for accurate prediction of the masked tokens, which in turn prevents the token representations to become identical. In a word, we can inference the masked autoencoder’s training objective helps to alleviate over-smoothing problem by its regularization on token differences.

To justify the reasoning above, we conduct both theoretical analysis and experimental verification. The results show that the over-smoothing issue is indeed alleviated in masked autoencoder. Compared with training under masked autoencoder, training transformer by a simple classification task (e.g., training vision transformer from scratch) does not have such strength.

If the masked autoencoder well alleviates the over-smoothing issue, which is a challenge for scaling transformer along depth, does this mean the masked autoencoder prefers deeper models than wider ones for some tasks? We argue that we need to re-visit the configurations for different training objectives, especially for masked autoencoder. To this end, we conduct experiments to investigate the masked autoencoder configurations and propose our idea, Bamboo. Bamboo is, when we train transformer with masked autoencoder, we suggest to use deeper and narrower configurations to achieve better effectiveness with comparable computation cost as a typical setting. To evaluate our new configurations, we conduct comprehensive experiments on computer vision and natural language processing tasks. On vision tasks, we evaluate our configuration on large-scale vision transformer training. With Bamboo configurations, the masked autoencoders outperforms baselines by a large margin. For instance, on ImageNet, with comparable number of trainable parameters and computation cost, our narrower and deeper base-scale masked autoencoder, Bamboo-B, outperform MAE-B by 0.9% in terms of top-1 accuracy. On natural language processing tasks, we conduct experiments on BERT. Results show that our configurations can improve BERT-L by 1.1 points on GLUE datasets.

In summary, our main contributions are three folds:

- We propose the insight that masked autoencoder is better at scaling along depth. Training objective of masked autoencoder helps to handle transformer over-smoothing. We justify this reasoning by both theoretical analysis and experimental verification.

2
• We argue that the existing transformer configurations cannot fully use the strength of masked autoencoder. To this end, we propose Bamboo, an idea to scale transformer along depth with training with masked autoencoder. We show that narrower and deeper version show better performance than existing configurations, in a plug-and-play manner.

• We further verify our Bamboo configurations on larger scale vision transformer pre-training and language tasks. Results show that our Bamboo achieves state-of-the-art top-1 accuracy on image classification, and outperforms original BERT configurations by 1.1 points on GLUE.

The following sections are organized based on the analysis process of this work. In Section 2, we briefly review the over-smoothing problem in transformer and show the strength of masked autoencoder-based self-supervised learning, respectively. We use supervised vision transformer (ViT) [9] and vision masked autoencoder (MAE) [13] as two representative platforms to show our insights in both theoretical and experimental analysis.

2 Over-smoothing in Masked Autoencoder

The over-smoothing issue is common in graph neural networks. When we stack many graph convolution networks, the node representations tend to be identical [3]. Recent studies show that transformer has a similar problem [9], known as “uniform tokens” in transformer models. In deep transformer, each token representation can be seen as a node in the graph, and each attention score is an edge. Therefore, different token representations tend to be identical when we scale transformer to deeper. In this work, we mainly focus on the over-smoothing problem due to different training settings.

2.1 Theoretical Analysis

We do theoretical analysis to compare the over-smoothing issue on sequence-level supervised learning, and masked autoencoder-based self-supervised learning, respectively. We use supervised vision transformer (ViT) [9] and vision masked autoencoder (MAE) [13] as two representative platforms to show our insights in both theoretical and experimental analysis.

Formally, transformer block’s output (i.e., token representations) can be written as \( h = \{h_0, h_1, \ldots, h_T\} \), where \( h_t \in \mathbb{R}^d \), \( d \) is the hidden dimension, \( T \) is the number of tokens at different transformer blocks. Usually, \( T \) is fixed across transformer blocks. For the token representations after \( l \)th transformer block, we denote the token representations as \( h_l^t \).

ViT is trained from scratch. The input is \( x = \{x_0, x_1, \ldots, x_{T-1}\} \), and the label is \( y \in \mathbb{R}^C \), where \( C \) is the number of classes. We analyze the over-smoothing issue when we use either the [CLS] token or average token representations as the prediction head.

If we use [CLS] token to classify an input image, the training target is to minimize the distance between \( f(h_0^T) \) and \( y \), where \( f \) is a linear projection. Observe that the final classification only relays on one token representation from the [CLS] token. More importantly, when over-smoothing happens, \( i.e., \ h_0^L = h_1^L = \ldots = h_{T-1}^L \), the prediction head remains meaningful for the task. If we use the average pooling prediction head, we reach the same conclusion. In this case, we are to minimize the distance between \( f(h_L^T) \) and \( y \), where \( h_L^T \) is the average pooling vector from \( L \)th layer token representations. If \( h_0^L = h_1^L = \ldots = h_{T-1}^L = h_L^T \), the classification output remains the same. However, model effectiveness will be limited since less attention layers are effectively used.

Then, what would happen if we use masked autoencoder to pre-train? For masked autoencoder, the input is \( x = \{x_0, x_1, \ldots, x_{T-1}\} \). If \( x_t \) token is masked, we denote this masked token as \( x_t^m \). The set of the masked tokens is \( x_m \). The corresponding original token set before masking is \( \tilde{x}_m \). During masked autoencoder pre-training, we are to minimize the distance between \( h^m \) and \( \tilde{x}_m \). We now state a key lemma to show that the masked autoencoder training target regularizes token representations. For simplicity, we treat each token representation as a scalar to analyze the variance Var(·). The following results can be extended to the vector case directly.
Lemma 2.1 (Variance of final token representations is bounded in masked autoencoder.) In masked autoencoder, given the variance of masked token representation $\sigma^2$, the variance of final token representations is bounded by the masked tokens:

$$\text{Var}(h^{L \text{m}}_{\text{LM}}) \geq \sigma^2$$

The proof of the Lemma above is in Appendix A.1. As the transformer can also be seen as a weighted multi-view graph, given the Lemma above, following the over-smoothing issue in graph neural network, we can also infer that the intermediate token representations are also bounded:

Lemma 2.2 (Variance of intermediate token representation is bounded in masked autoencoder.) In masked autoencoder, given the variance of intermediate token representations from $l$th layer is bounded by the variance of the token representation from $(l+1)$th layer:

$$\text{Var}(h^{l \text{m}}_{lm}) > \text{Var}(h^{l+1 \text{m}}_{l+1m})$$

The proof of the Lemma above is in Appendix A.2. Given the theoretical analysis above, we can find both the intermediate and final token representations are bounded by the variance of the masked tokens, i.e., $\text{Var}(h^{l \text{m}}_{lm}) > \text{Var}(h^{l+1 \text{m}}_{l+1m}) \geq \sigma^2$. Therefore, the over-smoothing issue is alleviated in masked autoencoder.

In fact, revisiting the training of ViT from scratch again, Lemma 2.2 also holds for the image classification setting. However, the difference is, Lemma 2.1 cannot work on training ViT from scratch. Without the masked token modeling target in MAE, there is no bound to push last layer’s token representations to be different. Since the Lemma 2.2 still holds, the variance of token representations decreases layer-by-layer. When we scale transformer to deeper, over-smoothing happens.

We do not take residual connection into consideration in the two lemmas above. In reality, the residual connection is a key module in transformer training to alleviate over-smoothing. The residual connection would keep variance increasing along depth, although over-smoothing may still be observed [8]. We thus, based on the two lemmas above, state the following theorem to show the variance in masked autoencoder increases faster than the variance in training transformer by simple classification setting.

Theorem 2.3 (Variance in masked autoencoder increases faster than the variance in training transformer by simple classification.) Given a converged masked autoencoder $g_1(\cdot)$ and transformer trained by simple classification $g_2(\cdot)$ with residual connection, the increase in variance

$$\Delta \text{MAE} = \text{Var}(x + g_1(x)) - \text{Var}(x)$$

is larger than,

$$\Delta \text{ViT} = \text{Var}(x + g_2(x)) - \text{Var}(x):$$

$$\Delta \text{MAE} > \Delta \text{ViT}$$

The proof of the theorem above is in Appendix A.3.

In summary, when we train transformer on the supervised learning setting directly and treat the task as a simple classification task, the transformer does not need to consider the differences across token representations. Such behavior means there is no regularization to avoid over-smoothing. For masked autoencoder training framework, the training target is to recover the masked tokens base on the neighboring tokens. Since the different masked tokens have different semantic-related neighbors, the different unmasked tokens must cover different semantics in deeper layers. Therefore, although we are not regularizing the transformer model in a straightforward way, the masked autoencoder training framework still pushes the token representations to be different.

2.2 Experimental Verification

Standard deviation. To verify our reasoning above, we conduct two sets of quantitative analysis. For the first set of quantitative analysis, we focus on the standard deviation of token representations at different transformer layers. To compare the over-smoothing issue between supervised learning and masked autoencoder, we calculate the mean standard deviation of the token representations at different transformer layers:

$$ms^4 = \frac{1}{T-1} \sum_{t=0}^{T-1} (h^t - \bar{h}^t)^2$$

2.2 Experimental Verification
where $\bar{h}_i = \frac{1}{T-1} \sum_{t=0}^{T-1} h^t_i$ is the mean vector of token representations at $i^{th}$ transformer block. We use a deeper version of ViT-L and MAE-L as platforms to validate our analysis above. Note that, the ViT-L is trained on the supervised learning setting and treats image classification as a simple classification task, by average pooling prediction head. The MAE-L is trained on the masked autoencoder setting, which means the pre-training target is closed to sequence labeling task. We then fine-tune the pre-trained model for image classification. We set the width as 768 and the depth as 48. The total number of trainable parameters and computation cost is comparable with the original configuration (i.e., width is 1024, depth is 24). The results are shown in Figure 1a.

Observe that mean standard deviations of ViT-L and MAE-L both increase along depth. Such a result matches well with our theoretical analysis. The residual connection within transformer indeed alleviates over-smoothing (to some extent) [8]. Nevertheless, the over-smoothing issue still exists even if we have residual connection as stated in [8, 19]. We can observe that the increasing on MAE-L is much faster than that on ViT-L, which matches well with our Theorem 2.3. That means the deeper transformer blocks can learn different semantics for different token representations in MAE. In other words, the over-smoothing issue is alleviated in the transformer model pre-trained with the masked autoencoder setting.

Patch-pair cosine similarity. To further verify that the over-smoothing issue can be alleviated in masked autoencoder, following Gong, Wang, Li, Chandra, and Liu [12], we compare the patch-pair cosine similarity between ViT-L and MAE-L with deeper and narrower configuration. If there is an over-smoothing issue in the model, we should observe that the patch-pair cosine similarity increases along depth. The more serious the over-smoothing issue is, the faster cosine similarity increases. To remove the impact from input representations (residual connection), we use the zero-centered token representations $\tilde{h}_i = h^t_i - \bar{h}_i$ for evaluation. The results are shown in Figure 1b. The cosine similarity of ViT increases along depth due to over-smoothing. However, for the model pre-trained by masked autoencoder framework, cosine similarity keeps constant along depth. This comparison is interesting, as we can barely observe over-smoothing issue on the model pre-trained by the masked autoencoder, even if we are using a deeper model than usual.

3 Bamboo

Our analysis above shows that the masked autoencoder helps to alleviate the over-smoothing issue, which is a main challenge in scaling transformer along depth. Based on this finding, we propose Bamboo, an idea of using deeper and narrower transformer configurations, for masked autoencoder pre-training. Certainly, deeper and narrower configurations do not always improve performance, as there are a few other reasons hindering scaling transformer along depth. In this section, we devote to find a sweet point following Bamboo idea, to achieve effective masked autoencoder by experiments.

To validate our reasoning above, we conduct experiments on training transformer on ILSVRC-2012 ImageNet [6] (ImageNet-1K), and report the top-1 accuracy. We use the original vit trained by supervised learning and MAE pre-trained masked image modeling as the test platform. For a fair comparison with the original transformer configurations, when we scale transformer to deeper, we also reduce the width to keep comparable number of trainable parameters and computation cost. The configurations we used are summarized in Table 1.
Table 1: The configurations we used to scale transformer along depth. For a fair comparison, we keep a comparable computation cost with the original transformer configurations (i.e., depth=12 for Base, depth=24 for Large).

| Scale | Base | Large |
|-------|------|-------|
| Depth | 12   | 24    | 48  | 96  | 24  | 48  | 60  | 96  |
| Width | 768  | 512   | 384 | 256 | 1024| 768 | 640 | 512 |
| #Attention Heads | 12 | 8 | 6 | 4 | 16 | 12 | 10 | 8 |
| Computation Cost | 1× | 0.9× | 1× | 0.9× | 1× | 1.1× | 1× | 1× |

Figure 2: Scaling ViT and MAE with comparable number of parameters and computation cost.

For supervised learning models (i.e., ViT), we train base scale models for 300 epochs and large scale models for 200 epochs following He, Chen, Xie, Li, Dollár, and Girshick [13]. We use RandAugment [5], drop path [14], mixup, cutmix [32], label smoothing [20] for data augmentation. Detailed hyper-parameters are summarized in Appendix B. For the masked autoencoder, we pre-train the base scale models for 1600 epochs and fine-tune for 100 epochs. For large models, we pre-train for 800 epochs and fine-tune for 50 epochs. We use the per-patch normalization following He, Chen, Xie, Li, Dollár, and Girshick [13] for better representations.

The results of scaling to deeper transformers are summarized in Figure 2. For both the base-scale and large-scale models, we report the top-1 accuracy on ImageNet-1K dataset. We can observe the models trained by supervised image classification directly (i.e., ViT-B and ViT-L) cannot improve accuracy when we use the deeper architectures. For ViT-B, the top-1 accuracy on ImageNet-1K keeps comparable when the depth is smaller than 50, but after that, there is a significant accuracy drop. For ViT-L, the accuracy decreases even faster than the ViT-B. There is a significant accuracy drop when we use the narrower and deeper models. However, for masked autoencoders, we can observe totally different patterns. Even if we keep the comparable trainable parameters and computation cost, with such a simple modification, the masked autoencoders gain significant improvements. More importantly, when we scale to 48 layers, both MAE-B and MAE-L reach sweet points. When scaling to 96 layers, we observe the training is unstable compared with shallower models. This result matches well with our observation in Figure 1b. At around 40th layer, the over-smoothing issue starts to happen slightly in masked autoencoder, which is much later than the ViT model. We suggest another reason of the unstable deep model training is the too large model updates [25]. In this work, we focus on the over-smoothing issue of deep transformer training instead. The large model updates in deeper layers are out-of-scope. We leave that as our future work.

According to the experimental results above, we find that masked autoencoder can indeed scale transformer well along depth. Even if we keep comparable trainable parameters and computation cost with the original transformer, the model achieves better accuracy. Another observation is, both transformer-base and transformer-large reach their sweet points at around 50 layers. We thus recommend a new set of transformer configurations in Table 2 following our Bamboo idea, which are deeper and narrower than the original transformer configurations.
Table 2: Re-designed configurations under Bamboo idea.

| Scale | Base       | Large      | Huge       |
|-------|------------|------------|------------|
|       | Original   | Bamboo     | Original   | Bamboo     | Original   | Bamboo     |
| Depth | 12         | 48         | 24         | 48         | 32         | 64         |
| Width | 768        | 384        | 1024       | 768        | 1280       | 896        |
| #Attention Heads | 12         | 6          | 16         | 12         | 16         | 14         |
| Computation Cost | 1×         | 1×         | 1×         | 1.1×       | 1×         | 1×         |

Table 3: Top-1 accuracy on ImageNet-1K. We report two versions of ViT training from scratch. The first one is from original ViT paper [9], and the second one is from He, Chen, Xie, Li, Dollár, and Girshick [13]'s re-implementation with strong data augmentation. For MAE-B, we reproduce the results by running the official code and obtain a slightly different result (denoted by 83.3*). The original result is 83.6.

| Method          | Pre-train Data | Base | Large | Huge |
|-----------------|----------------|------|-------|------|
| ViT from scratch [9] | -              | 77.9 | 76.5  | -    |
| DeepViT [33]    | -              | 80.9 | -     | -    |
| DeiT [22]       | -              | 81.8 | -     | -    |
| ViT from scratch [13] | -              | 82.1 | 81.5  | 80.9 |
| DINO [2]        | IN1K           | 82.8 | -     | -    |
| MoCo v3 [4]     | IN1K           | 83.2 | 84.1  | -    |
| BEiT [1]        | IN1K + DALL-E  | 83.2 | 85.2  | -    |
| MaskFeat [26]   | IN1K           | 84.0 | 85.7  | -    |
| IBOT [34]       | IN1K           | 84.0 | 84.8  | -    |
| MAE [13] (Direct baseline) | IN1K       | 83.3* | 85.9 | 86.9 |
| Bamboo (Ours)   | IN1K           | 84.2 | 86.3  | 87.1 |

4 Evaluation on Vision Tasks

4.1 Settings

We further evaluate our deeper and narrower transformer configurations on vision task. We train different models with more training epochs and compare with state-of-the-art vision models. We conduct experiments on ImageNet-1K and compare with recent supervised vision models e.g., DeepViT [33] and DeiT [22], and self-supervised vision models e.g., DINO [2], MoCo v3 [4], BEiT [1] and MAE [13]. Compared with the MAE, the only difference is the configurations. That is, MAE uses original transformer configurations and we use our Bamboo configurations. The experiments are to verify that such a simple modification can improve model effectiveness.

We evaluate the models on three different scales, i.e., base, large, and huge. The data augmentation setting is exactly the same as MAE for a fair comparison. Again, the only difference is that we use the Bamboo configurations instead of the original transformer configurations. During fine-tuning, we use the same script with training ViT from scratch. We fine-tune base models for 100 epochs. For large and huge models, we fine-tune them for 50 epochs following existing work [1, 13].

4.2 Results

Results on ImageNet-1K are reported in Table 3. For training ViT from scratch, we report the original results [9] and the results with strong data augmentation [13]. A few recent work [22, 33] focusing on training ViT from scratch on ImageNet-1K are also included. In general, we can find the models pre-trained by self-supervised learning (e.g., DINO, MoCo v3, MAE) perform much better than training from scratch. If we only consider the self-supervised learning approaches, masked image modeling-based methods (e.g., BEiT, MaskFeat, IBOT, MAE) outperform the contrastive learning-based methods (e.g., DINO, MoCo v3) significantly, especially on larger scale.

Since we are focusing on the scalability of training transformer with masked autoencoder, we choose MAE as our direct baseline, recent work with better scalability. We train MAE with Bamboo
Table 4: Results of fine-tuning on GLUE benchmark.

| Method  | CoLA | MNLI | MRPC | QNLI | QQP | RTE | SST-2 | STS-B | Avg |
|---------|------|------|------|------|-----|-----|-------|-------|-----|
| BERT-B  | 59.6 | 83.7 | 88   | 90.4 | 88.8| 68.6| 91.5  | 89.4  | 82.5|
| BERT-L  | 60.9 | 86.2 | 89.3 | 92.3 | 89.6| 73.1| 92.5  | 90.4  | 84.3|
| Bamboo-B| 60.5 | 84.3 | 88.4 | 90.5 | 89.0| 70.0| 92.2  | 89.5  | 83.1|
| Bamboo-L| **62.9** | **87.1** | **89.8** | **92.4** | **89.4**| **77.3**| **93.8** | **90.6** | **85.4**|

configurations and report the results in Table 3. Observe that our Bamboo achieves the best top-1 accuracy on all scales. On the base scale, Bamboo achieves state-of-the-art performance, 84.2 top-1 accuracy, which is 0.9 (0.6) points higher than MAE-B. When we scale the model up to large scale and huge scale, Bamboo remains the best performer and achieves 86.3 and 87.1 top-1 accuracy respectively. Note that, compared to other scales, the improvement is not so significant on the huge scale. One reason is, the original huge configuration has been deep (i.e., 32 layers), which is closer to the sweet point. Similarly, this can also explain why the configurations designed under Bamboo can improve MAE-B significantly.

5 Evaluation on Language Tasks

5.1 Settings

We further evaluate our Bamboo configurations on language tasks. We select BERT [7] as the platform to evaluate our Bamboo configuration because it is widely used on many language tasks. Note that BERT is a post-layer normalization transformer model. If our configuration shows better performance on BERT, that means it has good generalization performance. We follow the BERT paper to use Wikipedia and bookscorpus to pre-train. The masked ratio is 15% as by default. During pre-training, we use LAMB optimizer and set batch size and learning rate as 4096 and 1.76e-3, respectively, following You, Li, Reddi, Hseu, Kumar, Bhojanapalli, Song, Demmel, Keutzer, and Hsieh [31]. During fine-tuning, we conduct experiments on General Language Understanding Evaluation (GLUE) benchmark. The GLUE benchmark [24] is widely used in natural language understanding tasks, which include 8 tasks, i.e., CoLA, MNLI, MRPC, QNLI, QQP, RTE, SST-2 and STS-B. We set the learning rate as 1e-5 or 2e-5. Batch size is fixed as 32. For small datasets, i.e., CoLA, MRPC, RTE and STS-B, we fine-tune for 10 epochs. For larger datasets, i.e., MNLI, QNLI, QQP and SST-2, we fine-tune for 3 epochs. Matthew’s correlation is used as metric for CoLA. For MNLI, we report the average accuracy on MNLI-m and MNLI-mm. QNLI and RTE also adapt the accuracy as metric. The results on MRPC and QQP are reported with the average of F1 and accuracy. We use Spearman correlation on STS-B. For all fine-tuning results, we run the code 5 times and report the median.

5.2 Results

The results on language tasks are reported in Table 4. Under the same pre-training and fine-tuning settings, models with Bamboo configurations outperform BERT by a large margin. Bamboo-L achieves the best performance in Table 4. Compared with BERT-L, our Bamboo-L wins on 7 out of 8 datasets, and surpassed BERT-L by 1.1 points on average. It is also notable Bamboo configurations can outperform baselines on both large datasets (e.g., MNLI) and small datasets (e.g., CoLA).

6 Discussion

Compared with simply scaling along depth, this work keeps comparable computation cost and number of trainable parameters. When scaling along depth, we also make the deep transformer narrower. Under such setting, the deeper and narrower configurations re-designed under Bamboo idea can still outperform the baseline configurations, suggesting that we should consider narrower and deeper transformer when training by masked autoencoder. One related work is [21], an empirical study of practical scaling of transformer, which has a similar observation, deeper and narrower can improve accuracy. On this basis, our insight is, the insight in [21] depends on different training objectives.
Compared with brute-force hyper-parameter tuning, we provide both theoretical analysis and experimental verification to justify our insight, i.e., masked autoencoder alleviates over-smoothing issue in transformer. Motivated by this, we propose Bamboo by experimental investigation. There is no guarantee to ensure the configurations are always the optimal ones as different tasks may require different configurations. We believe it is impossible in deep learning to know the optimal configurations before experiments. However, we justify the tendency that masked autoencoder gets more benefits from deeper configurations. Our insight may instruct future works to consider configurations according to the training objectives.

Instead of proposing a new approach or a new set of configuration, this work focuses on an existing but neglected problem. After revisiting, we find configurations should be re-designed for different training objectives. We then re-design a set of configurations for masked autoencoder, which can improve transformer on both vision and language tasks. On vision tasks, we even achieve SoTA top-1 accuracy on ImageNet. However, we highlight our main contribution is an insight instead of a SoTA model, and this insight is orthogonal to the advanced masked autoencoder modifications in the future.

7 Conclusion

We first rethink the masked autoencoder’s scalability in this work. Through theoretical analysis, we show that training transformer with masked autoencoder can alleviate the over-smoothing issue when scaling transformer along depth. We then verify this finding through experiments. The results show that, compared with supervised learning, model pre-trained by masked autoencoder can handle over-smoothing (i.e., uniform tokens) better. Under this insight, we rethink the widely used configurations in vision and language transformer, and suggest deeper transformer when training with masked autoencoder. We then explore this strategy on base and large scale vision transformer and re-design a set of vision transformer configurations for masked autoencoder empirically. To further verify the effectiveness of our configuration, we conduct experiments on larger scale vision transformer training and achieved SoTA top-1 accuracy with comparable computation cost on ImageNet-1K. To show our configurations are better than original transformer configurations, we further conduct experiments on language tasks. Compared with widely used language masked autoencoder, BERT, our model surpass it by 1.1 points on average on GLUE benchmark. In summary, our work provides an insight to explain why masked autoencoder can scale transformer along depth, and we propose Bamboo, a set of configurations to fully utilize such strength.

In the future, we plan to integrate the existing methods focusing on scaling along depth, e.g., DeepNorm [25] to further scale transformer. Another future direction is to verify our Bamboo configurations on other masked autoencoder-based models (e.g., BEiT [1], RoBERTa [15]).

References

[1] H. Bao, L. Dong, and F. Wei, “Beit: Bert pre-training of image transformers,” arXiv preprint arXiv:2106.08254, 2021.
[2] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9650–9660.
[3] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, “Measuring and relieving the over-smoothing problem for graph neural networks from the topological view,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 3438–3445.
[4] X. Chen, S. Xie, and K. He, “An empirical study of training self-supervised vision transformers,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9640–9649.
[5] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “RandAugment: Practical automated data augmentation with a reduced search space,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 702–703.
[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.
[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.

[8] Y. Dong, J.-B. Cordonnier, and A. Loukas, “Attention is not all you need: Pure attention loses rank doubly exponentially with depth,” in International Conference on Machine Learning, PMLR, 2021, pp. 2793–2803.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” arXiv preprint arXiv:2112.09055, 2021.

[10] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” arXiv preprint arXiv:2101.03961, 2021.

[11] C. Gong, D. Wang, M. Li, V. Chandra, and Q. Liu, “Vision transformers with patch diversification,” arXiv preprint arXiv:2111.06377, 2021.

[12] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” arXiv preprint arXiv:2111.06377, 2021.

[13] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep networks with stochastic depth,” in European conference on computer vision, Springer, 2016, pp. 646–661.

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” arXiv preprint arXiv:1907.11692, 2019.

[15] J. Ni, T. Young, V. Panalea, F. Xue, V. Adiga, and E. Cambria, “Recent advances in deep learning based dialogue systems: A systematic survey,” arXiv preprint arXiv:2105.04387, 2021.

[16] C. Qu, L. Yang, M. Qiu, W. B. Croft, Y. Zhang, and M. Iyyer, “Bert with history answer embedding for conversational question answering,” in Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 1133–1136.

[17] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” 2018.

[18] H. Shi, J. Gao, H. Xu, X. Liang, Z. Li, L. Kong, S. M. Lee, and J. Kwok, “Revisiting over-smoothing in bert from the perspective of graph,” in International Conference on Learning Representations, 2021.

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

[20] Y. Tay, M. Dehghani, J. Rao, W. Fedus, S. Abnar, H. W. Chung, S. Narang, D. Yogatama, A. Vaswani, and D. Metzler, “Scale efficiently: Insights from pre-training and fine-tuning transformers,” arXiv preprint arXiv:2109.10686, 2021.

[21] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Training data-efficient image transformers & distillation through attention,” in International Conference on Machine Learning, PMLR, 2021, pp. 10.347–10.357.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 6000–6010.

[23] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, “GLUE: A multi-task benchmark and analysis platform for natural language understanding,” in Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446. [Online]. Available: https://aclanthology.org/W18-5446.

[24] H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei, “Deepnet: Scaling transformers to 1,000 layers,” arXiv preprint arXiv:2203.00555, 2022.

[25] C. Wei, H. Fan, S. Xie, C.-Y. Wu, A. Yuille, and C. Feichtenhofer, “Masked feature prediction for self-supervised visual pre-training,” arXiv preprint arXiv:2112.09133, 2021.
[27] F. Xue, X. He, X. Ren, Y. Lou, and Y. You, “One student knows all experts know: From sparse to dense,” arXiv preprint arXiv:2201.10890, 2022.
[28] F. Xue, Z. Shi, F. Wei, Y. Lou, Y. Liu, and Y. You, “Go wider instead of deeper,” arXiv preprint arXiv:2107.11817, 2021.
[29] F. Xue, A. Sun, H. Zhang, and E. S. Chng, “An embarrassingly simple model for dialogue relation extraction,” arXiv preprint arXiv:2012.13873, 2020.
[30] Z. Yang, N. Garcia, C. Chu, M. Otani, Y. Nakashima, and H. Takemura, “Bert representations for video question answering,” in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 1556–1565.
[31] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, “Large batch optimization for deep learning: Training bert in 76 minutes,” arXiv preprint arXiv:1904.00962, 2019.
[32] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “Cutmix: Regularization strategy to train strong classifiers with localizable features,” in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6023–6032.
[33] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Z. Jiang, Q. Hou, and J. Feng, “Deepvit: Towards deeper vision transformer,” arXiv preprint arXiv:2103.11886, 2021.
[34] J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. Yuille, and T. Kong, “Ibot: Image bert pre-training with online tokenizer,” arXiv preprint arXiv:2111.07832, 2021.
[35] W. Zhou, K. Huang, T. Ma, and J. Huang, “Document-level relation extraction with adaptive thresholding and localized context pooling,” arXiv preprint arXiv:2010.11304, 2020.
Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releases new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
Appendix

A Missing Proofs

In this section we fill out the proofs of lemmas made in the main paper. For simplicity, we treat each token representation as a scalar to analyze the variance. The following results can be extent to the vector case easily.

A.1 Proof of Lemma 2.1

Lemma 2.1. In masked autoencoder, given the variance of masked token representation $\sigma^2$, the variance of final token representations is bounded by the masked tokens:

$$\text{Var}(h_L^m) \geq \sigma^2$$  \hspace{1cm} (5)

Proof. In masked autoencoder, the input is $x = \{x_0, x_1, ..., x_{T-1}\}$. During pre-training, we are to minimize the distance between last layer masked token representations $h_L^m$ and corresponding original token set is $\tilde{x}^m$. Given $\text{Var}(\tilde{x}^m) = \sigma^2$, as we are minimizing $||h_L^m - \tilde{x}^m||$, we can assume $h_L^m - \tilde{x}^m = \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Then, the variance of $h_L^m$ can be written as:

$$\text{Var}(h_L^m) = \text{Var}(\tilde{x}^m + \epsilon) = \text{Var}(\tilde{x}^m) + \text{Var}(\epsilon) = \sigma^2 + \sigma^2 \epsilon$$  \hspace{1cm} (6)

Obviously, since $\sigma^2 \epsilon \geq 0$, $\text{Var}(h_L^m) \geq \sigma^2$.

A.2 Proof of Lemma 2.2

Lemma 2.2. In masked autoencoder, given the the variance of intermediate token representations from $l^{th}$ layer is bounded by the variance of the token representation from $l + 1^{th}$ layer:

$$\text{Var}(h_l^m) \geq \text{Var}(h_{l+1}^m)$$  \hspace{1cm} (7)

Proof. Since the attention mechanism can be seen as weighted averaging, we obtain the variance of intermediate token representations from $l^{th}$ layer:

$$\text{Var}(h_{l+1}^m) = \text{Var}(g(f(\sum_{i=0}^{T-1} w_i h_i^m))) = \sum_{i=0}^{T-1} \text{Var}(w_i h_i^m)$$  \hspace{1cm} (8)

where $f(\cdot)$ is linear projection and $g(\cdot)$ is ReLU activation function. Since we initialize $f(\cdot)$ by Xaiver initialization in usual, it will not increase the variance in usual. After that, the ReLU would also only decrease the variance of input. Since the attention weight $w$ has been normalized by softmax, $\sum_{i=0}^{T-1} w_i^2 \leq 1$. Then, $\text{Var}(h_{l+1}^m)$ can be written as:

$$\text{Var}(h_{l+1}^m) = \text{Var}(h_i^m) \sum_{i=0}^{T-1} w_i^2 < \text{Var}(h_i^m)$$  \hspace{1cm} (9)
A.3 Proof of Theorem 2.3

Theorem 2.3. Given a converaged masked autoencoder $g_1(\cdot)$ and transformer trained by simple classification $g_2(\cdot)$ with residual connection, the increase in variance $\Delta \text{MAE} = \text{Var}(x + g_1(x)) - \text{Var}(x)$ is larger than, $\Delta \text{ViT} = \text{Var}(x + g_2(x)) - \text{Var}(x)$:

$$\Delta \text{MAE} > \Delta \text{ViT}$$

Proof. We assume both MAE and ViT are well-trained on the same dataset until convergence, and the MAE and ViT have the same model architecture (e.g., the same hidden dimension). We use transformer with one layer to prove in this section. The results can also be extent to multiple transformer layers. First of all, we need to note the input of masked autoencoder is different from the transformer trained by simple classification. The input tokens of masked autoencoder is partly masked, we denote it by $x^m$. Instead, the input of simple classification transformer is a sequence of tokens without masking $x$. Then, the variance of MAE’s output can be written as:

$$\text{Var}(h^m) = \text{Var}(x^m + g_1(x^m)) = \text{Var}(x^m) + \text{Var}(g_1(x^m)) + 2\text{Cov}(x^m, g_1(x^m))$$

where $h^m$ is the output of masked autoencoder. Similarly, the variance of simple classification transformer’s output can be written as:

$$\text{Var}(h) = \text{Var}(x + g_2(x)) = \text{Var}(x) + \text{Var}(g_2(x)) + 2\text{Cov}(x, g_2(x))$$

To simplify Eq. 11, we set $x^m = x - x^m$, where $x^m$ is original token value if the token is masked, and $x^m = 0$ if not the token is not masked. Then, according to Lemma 2.1, $\text{Var}(h^m)$ can be written as:

$$\text{Var}(h^m) = \text{Var}(x^m) + \text{Var}(g_1(x^m)) + 2\text{Cov}(x^m, x^m)$$

Then, $\Delta \text{MAE}$ can be written as:

$$\Delta \text{MAE} = \text{Var}(h^m) - \text{Var}(x^m) = 2\text{Var}(x) + \text{Var}(g_1(x^m)) - 2\text{Cov}(x^m, x)$$

Similarly, for $\Delta \text{ViT}$, we have:

$$\Delta \text{ViT} = \text{Var}(h) - \text{Var}(x) = \text{Var}(g_2(x)) + 2\text{Cov}(g_2(x), x)$$

For each linear projection without residual connection, we assume $g_2(x)$ is independant to $x$ in a well-trained transformer:

$$\Delta \text{MAE} > 3\text{Var}(x) - 2\text{Cov}(x^m', x)$$

According to Lemma 2.2, we have:

$$\Delta \text{MAE} > \text{Var}(x) > \text{Var}(g_2(x)) = \Delta \text{ViT}$$
## B Fine-tuning Hyper-parameters

Table 5: Hyper-parameters on ImageNet fine-tuning

| Parameter                | Base | Large | Huge |
|--------------------------|------|-------|------|
| Epoch                    | 100  | 50    | 50   |
| Warmup Epochs            | 5    |       |      |
| Batch Size               | 1024 |       |      |
| Learning rate            | 2e-3 |       |      |
| Layer-wise learning rate decay | 0.65 | 0.75  | 0.75 |
| Weight Decay             | 0.05 |       |      |
| DropPath                 | 0.1  | 0.2   | 0.2  |
| Label smoothing          | 0.1  |       |      |
| Erasing prob.            | 0.25 |       |      |
| RandAug                  | 9/0.5|       |      |
| Mixup prob.              | 0.8  |       |      |
| Cutmix prob.             | 1.0  |       |      |