Multi-Path Transformer is Better: A Case Study on Neural Machine Translation

Ye Lin¹, Shuhan Zhou¹, Yanyang Li², Anxiang Ma¹, Tong Xiao¹,³, Jingbo Zhu¹,³

¹NLP Lab, School of Computer Science and Engineering, Northeastern University, Shenyang, China
²The Chinese University of Hong Kong, Hong Kong, China
³NiuTrans Research, Shenyang, China

{linye2015,zhoushuhan1997,blamedrlee}@outlook.com
{maanxiang,xiaotong,zhujingbo}@mail.neu.edu.cn

Abstract

For years the model performance in machine learning obeyed a power-law relationship with the model size. For the consideration of parameter efficiency, recent studies focus on increasing model depth rather than width to achieve better performance. In this paper, we study how model width affects the Transformer model through a parameter-efficient multi-path structure. To better fuse features extracted from different paths, we add three additional operations to each sublayer: a normalization at the end of each path, a cheap operation to produce more features, and a learnable weighted mechanism to fuse all features flexibly. Extensive experiments on 12 WMT machine translation tasks show that, with the same number of parameters, the shallower multi-path model can achieve similar or even better performance than the deeper model. It reveals that we should pay more attention to the multi-path structure, and there should be a balance between the model depth and width to train a better large-scale Transformer.

1 Introduction

The large-scale neural network has achieved great success at a wide range of machine learning tasks (Deng et al., 2009; Devlin et al., 2019; Shazeer et al., 2017). Among them, deep models show great potential to deal with complex problems (He et al., 2016a; Wang et al., 2019). By stacking more layers, deeper models generally perform better than shallower models since they provide more non-linearities to learn more complex transformations (Telgarsky, 2015; Liu et al., 2020a,b). Compared to increasing the model depth, broadening the model width can also benefit the model by providing richer features in a single layer.

There are two ways to broaden the model width: 1) Scaling the matrix dimensions, such as turning the model configuration from Transformer-base to Transformer-big (Vaswani et al., 2017). However, both the number of parameters and computational costs will increase quadratically, making such model training and deployment difficult; 2) Adopting the multi-path structure (Ahmed et al., 2017). The expressive power of such models can be improved by fusing abundant features obtained from different paths, and the parameters and computations will only increase linearly with the number of paths.

The multi-path structure has proven to be quite important in convolutional networks for computer vision tasks (Zhang et al., 2020; Zagoruyko et al., 2016). But in Transformer, this type of structure has not been widely discussed and applied (Ahmed et al., 2017; Fan et al., 2020). In this paper, we continue to study the multi-path structure and adopt a sublayer-level multi-path Transformer model. As shown in Fig. 1, the original multi-path models ■ significantly outperform wide models ▲ that scale the matrix dimensions. To make better use of the features extracted from different paths, we redesign the multi-path model by introducing three additional operations in each layer: 1) A normalization at the end of each path for regularization and ease of training; 2) A cheap operation to produce more features; 3) A learnable weighted mechanism to make the training process more flexible.

To demonstrate the effectiveness of our method,
we test it on the Transformer-base and Transformer-deep configurations. The experiments are run on 12 WMT machine translation benchmarks, including WMT14 English↔{German, French} and WMT17 English↔{German, Finnish, Latvian, Russian}. Experiments on the most widely used English→German task show that the multi-path Transformer model can achieve 2.65 higher BLEU points with the same depth as the Transformer-base model. What’s more, a shallower multi-path Transformer can achieve 29.68 BLEU points, which is even higher than the 48-layer Transformer-deep model of the same size. It inspires us that, model width is as important as model depth, instead of indefinitely stacking more layers, we should pay more attention to the multi-path structure.

2 Background

2.1 Transformer

Transformer is an attention-based encoder-decoder model that has shown promising results in many machine learning tasks (Vaswani et al., 2017; Devlin et al., 2019; Liu et al., 2020b). It mainly consists of two kinds of structures: the multi-head attention and the feed-forward network.

The multi-head attention computes the attention distribution $A_x$ and then averages the input $X$ by $A_x$. We denote the attention network as $\text{MHA}(\cdot)$:

$$A_x = \text{SoftMax}(XW_qW_k^TX^T)$$

$$\text{MHA}(X) = A_xXW_v$$

where $X \in \mathbb{R}^{t \times d}$, $t$ is the target sentence length and $d$ is the dimension of the hidden representation, $W_q, W_k, W_v \in \mathbb{R}^{d \times d}$.

The feed-forward network applies a non-linear transformation to its input $X$. We denote the output as $\text{FFN}(\cdot)$:

$$\text{FFN}(X) = \text{ReLU}(XW_1 + b_1)W_2 + b_2$$

where $W_1 \in \mathbb{R}^{d \times 4d}$, $b_1 \in \mathbb{R}^{4d}$, $W_2 \in \mathbb{R}^{4d \times d}$ and $b_2 \in \mathbb{R}^{d}$.

Both the MHA and FFN are coupled with the residual connections (He et al., 2016a) and layer normalizations (Ba et al., 2016). For stabilizing deep models training, here we adopt the normalization before layers that has been discussed in Wang et al. (2019)’s work.

2.2 Multi-Path Transformer

As a way to enlarge the model capacity, the idea of multi-path networks has been explored widely and has proven to be important in several domains (Ahmed and Torresani, 2017; Zhang et al., 2019). Among them, Ahmed et al. (2017) replace the multi-head attention with multiple self-attentions. Fan et al. (2020) propose MAT, in which the attention layer is the average of multiple independent multi-head attention structures. The MoE proposes to dynamically choose paths in a very large-scale network (Shazeer et al., 2017).

Here, we continue to study the sublayer-level multi-path structure based on the Transformer model. The multi-path structure is applied both on the multi-head attention and feed-forward network as shown in Fig. 2, and the constructed model can be seen as a case of the MoE without dynamic computation. We discuss that width is an important factor that should not be ignored especially when the model becomes too deep.

3 Methods

3.1 PathNorm and The Weighted Mechanism

Fig. 3 shows the architecture of the multi-path Transformer model constructed in this paper. In the implementation, we adopt the normalization before layers because it has proven to be more robust to deep models than the normalization after layers (Baevski and Auli, 2019; Xiong et al., 2020; Nguyen and Salazar, 2019). In this model, different paths are split after each layer normalization and
fused at a sublayer level. To better fuse features extracted from different paths, three additional operations are proposed in this paper. In this section, we will introduce two of these three operations, the other one will be introduced in Section 3.2.

PathNorm. As shown in Fig. 2, an additional normalization (named PathNorm) is introduced at the end of each multi-head attention (MHA) or feed-forward network (FFN). Different from Shleifer et al. (2021)'s work, the proposed PathNorm aims to bring the magnitudes of output distributions closer, which we think is more conducive to the fusion of different paths. When the number of paths becomes relatively large, it also plays a role in regularization and ensures the stability of the model training.

The Weighted Mechanism. To enable the model to learn how to combine paths on its own, a learnable weighted mechanism is introduced. As shown in Fig. 2, learnable weights \( \alpha \) are added on all model paths, and the residual connections surrounding layers are also equipped with learnable weights \( \beta \). By adopting this strategy, the model can automatically distinguish which part is more important and the training process will be more flexible.

For this multi-path Transformer model, we can write the output of multi-head attention or feed-forward network as:

\[
Y = \beta X + \sum_{i=1}^{n} \alpha_i \text{Path}_i(X)
\]

where \( X \) denotes the layer input, \( Y \) denotes the layer output, \( n \) denotes the total number of paths. \( \alpha \) and \( \beta \) are respectively learnable weights added on the model paths and residual connections. We denote the computation of the multi-head attention in Eq. 2 as MHA(·) and the computation of the feed-forward network in Eq. 3 as FFN(·). In the multi-head attention or feed-forward network, each path can be computed as:

\[
\text{Path}(X) = \text{PathNorm}(\text{Func}(\text{LN}(X)))
\]

\[
\text{Func}(X) = \text{MHA}(X) \text{ or FFN}(X)
\]

where PathNorm is the normalization added after the computation of each multi-head attention or feed-forward layer. LN is the layer normalization.

### 3.2 More Features from Cheap Operations

With the increasing number of paths, the model tends to get better performance, but the number of parameters and computational costs will also increase correspondingly. What’s worse, one model with too many paths will be hard to train because of much more GPU memory resources consuming.

To solve the above-mentioned problem, here we propose to generate more features from the existing ones through a cheap operation. This method can help the multi-path model achieve better performance with almost negligible additional computational costs, and it has no effect on the overall parameters. Specifically, here we adopt a "selection then combination" strategy. In the example of Fig. 3, Paths 1 ∼ 4 denote paths of the current Transformer layer, the features generated from...
these paths are called "raw features", and the features further generated by "raw features" are called "new features". This process can be divided into the listed two steps.

**Selection.** Firstly, we need to select a few paths for the next combination operation. Since we want to get different features, the selection must be without repetition. Since different paths are independent, so the selection order does not matter. It should be noted that if each path is selected too many times, it will weaken the feature diversities. While, if each path is chosen only a small number of times, there will be fewer benefits. Here we select \( n - 1 \) paths from the total \( n \) paths once time, until all subsets of paths that meet this condition are selected without repetition. From \( n \) paths, there will be \( C_n^{n-1} = n \) subsets of paths, and the corresponding number of newly generated features will be \( n \). Through this selection strategy, there will be a balance between the rawly existing features and the newly generated features.

**Combination.** Since we have obtained \( n \) subsets of paths from the selection operation, we need to combine paths from the same subset to generate new features. To compute the combination result, here we adopt a simple average operation. Specifically, we average the outputs of different paths in each attention or feed-forward network which have been computed in Eq. 6. Suppose the number of paths is set to \( n \), then the number of paths in one subset is \( k = n - 1 \), the average operation can be denoted as below:

\[
\text{AVG}(X) = \frac{\sum_{i=1}^{k} \text{Func}_i(X)}{k} \quad (7)
\]

After the combination operation to produce \( n \) new features, we add additional normalizations as described in Section 3.1. Different from the previous description, here we add PathNorm on both the "raw features" and "new features". Besides, learnable weights \( \alpha \) and \( \beta \) are also added for weighting these two kinds of features as shown in Fig. 3.

**Efficiency.** Considering the parameter efficiency, although we need to introduce additional normalizations with twice the number of model paths, it has little effect on the total number of parameters since the parameters in each normalization are very limited. As for the computation efficiency, because the average operation is quite lightweight and the dimension of the sublayer output is relatively small, the combination operation will only have a small impact on the overall training efficiency. Since we only experiment on the Transformer encoder, it has nearly no impact on the inference efficiency.

### 3.3 The Initialization of \( \alpha \) and \( \beta \)

In Section 3.1, we have introduced the learnable weights \( \alpha \) and \( \beta \) for the path outputs and residual connections respectively. Here we introduce how to initialize \( \alpha \) and \( \beta \) in this work. Since the result of one path coupled with \( \text{PathNorm} \) follows the normal distribution with mean 0 and variance 1, to make the sum of multiple paths approximately equal to the standard normal distribution, here we set \( \alpha = \frac{1}{\sqrt{n}} \), where \( n \) is the number of "raw features" in the current layer. To balance the residual connections and paths in the initial training stage, we set \( \beta = 1 \) in all sublayers.

### 4 Experiments

#### 4.1 Setup

We evaluate our methods on 12 machine translation tasks (6 datasets \( \times \) 2 translation directions each), including WMT14 English↔{German, French} (En↔{De, Fr}) and WMT17 English↔{German, Finnish, Latvian, Russian} (En↔{De, Fi, Lv, Ru}). The statistics of all datasets are shown in Table 1.

<table>
<thead>
<tr>
<th>Source</th>
<th>Task</th>
<th>Train</th>
<th>Valid</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sent.</td>
<td>word</td>
<td>sent.</td>
<td>word</td>
</tr>
<tr>
<td></td>
<td>word</td>
<td></td>
<td>word</td>
<td></td>
</tr>
<tr>
<td>WMT14</td>
<td>En+De</td>
<td>4.5M</td>
<td>220M</td>
<td>3000</td>
</tr>
<tr>
<td></td>
<td>En+Fr</td>
<td>35M</td>
<td>2.2B</td>
<td>26K</td>
</tr>
<tr>
<td>WMT17</td>
<td>En+De</td>
<td>5.9M</td>
<td>276M</td>
<td>8171</td>
</tr>
<tr>
<td></td>
<td>En+Fi</td>
<td>2.6M</td>
<td>108M</td>
<td>8870</td>
</tr>
<tr>
<td></td>
<td>En+Lv</td>
<td>4.5M</td>
<td>115M</td>
<td>2003</td>
</tr>
<tr>
<td></td>
<td>En+Ru</td>
<td>25M</td>
<td>1.2B</td>
<td>8819</td>
</tr>
</tbody>
</table>

Table 1: Data statistics (# of sentences and # of words).

1. http://statmt.org/
Table 2: Results on WMT14 En→De (We mark the best system with ✓ under the same number of parameters. The original multi-path models with different paths are represented as “Multi-Path2~8”. Our models with PathNorm and learnable weighted mechanism are represented as “+ Ours”, our models with more features based on “+ Ours” are represented as “+ More Features”).

Models. Our baseline system is based on the open-source implementation of the Transformer model presented in Ott et al. (2019)’s work. For all machine translation tasks, we construct baseline models with the Transformer-base and Transformer-deep (Wang et al., 2019) settings. All baseline systems consist of a 6-layer encoder and a 6-layer decoder, except that the Transformer-deep encoder has 12~48 layers (depth) (Li et al., 2020). The embedding size is set to 512 for both the Transformer-base and deep. The FFN hidden size equals 4× embedding size in all settings. As for the multi-path Transformer models, except for the number of paths, all other model hyperparameters are the same as the baseline models. The multi-path models in this paper consist of 2~8 paths.

Training Details. For training, we use Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.997$. For the Transformer-base setting, we adopt the inverse square root learning rate schedule with 8,000 warmup steps and 0.001 learning rate. For the Transformer-deep and multi-path settings, we adopt the inverse square root learning rate schedule with 16,000 warmup steps and 0.002 learning rate. The training batch size of 4,096 is adopted in the base setting, and 8,192 is adopted in the deep and multi-path settings. All experiments are done on 8 NVIDIA TITIAN V GPUs with mixed-precision training (Micikevicius et al., 2018). All results are reported based on the model ensembling by averaging the last 5 checkpoints.

4.2 Results
Table 2 and Table 3 show the results of different systems on WMT14 En→De and WMT14 En→Fr. In all tasks, the original multi-path models can not perform as well as the deep models, which proves...
Table 3: Results on other WMT14 tasks (We mark the best system with ✓, ✗ means that the model cannot continue training due to the gradient exploding problem.).

![Figure 4: Comparisons of different systems on WMT17 tasks (The figure shows ΔBLEU of each system exceeds the baseline model. Deep, MP, and Ours denote results of deep models, original multi-path models, and our multi-path models. Blocks with/without lines represent results on validation/test set. Different colors denote tasks of En↔De, En↔Fi, En↔Lv and En↔Ru.).](image)

that model depth does play a crucial role in performance. However, our multi-path models can achieve similar or even better results than the deep models. On the En→De dataset, our best multi-path system achieves 0.12/0.32/0.28/0.25 higher BLEU points than the deep model when the number of parameters is set to 80/118/156/193 megabytes. It shows the potential of the multi-path models and proves that model width is as important as model depth. Under the same model depth, multi-path models with more features significantly perform better than the original multi-path models, which demonstrates the effectiveness of our proposed method in Section 3. Note that in our method of generating more features, there will be $c_{n-1} = n$ new features. In a 2-path model, since there are only 2 paths, no new features will be generated.

Experiments on En→Fr, De→En and Fr→En also show the competitive performance of the multi-path Transformer models. Note that on Fr→En task, in the mix-precision training process of the 24-layer model, we met the problem of loss exploding. It means that the minimum loss scale (0.0001 in the fairseq fp16 optimizer) has been reached and the loss is probably exploding. We further validate our conclusions on 8 WMT17 tasks, including En↔{De, Fi, Lv, Ru}. Experiments in Fig. 4 show a similar phenomenon and further verify that, instead of indefinitely stacking more layers, we should pay more attention to wider structures, such as the multi-path models.

5 Analysis

5.1 Shallower Networks

In this section, we study the performance of our multi-path structure in shallower networks. Fig. 5 shows the results of Transformer models with different numbers of depths and paths. When the model is relatively shallower, increasing the number of paths will produce slightly worse perfor-

4https://github.com/pytorch/fairseq/blob/v0.6.2/fairseq/optim/fp16_optimizer.py
Figure 5: Results of shallower networks with different depths and paths on WMT14 En→De (Darker color means better performance.).

<table>
<thead>
<tr>
<th>System</th>
<th>Params</th>
<th>BLEU</th>
<th>Test</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>62M</td>
<td>27.00</td>
<td>25.88</td>
<td></td>
</tr>
<tr>
<td>+ Multi-Path</td>
<td>156M</td>
<td>28.87</td>
<td>26.58</td>
<td></td>
</tr>
<tr>
<td>+ PathNorm</td>
<td>156M</td>
<td>28.72</td>
<td>26.28</td>
<td></td>
</tr>
<tr>
<td>+ Learnable Weights</td>
<td>156M</td>
<td>29.13</td>
<td>26.93</td>
<td></td>
</tr>
<tr>
<td>- PathNorm</td>
<td>156M</td>
<td>28.86</td>
<td>26.58</td>
<td></td>
</tr>
<tr>
<td>+ More Features</td>
<td>156M</td>
<td>29.65</td>
<td>26.89</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Ablation study on WMT14 En→De.

5.2 Ablation Study

Table 4 summarizes and compares the contributions of each part described in Section 3. Each row of Table 4 is the result of applying the current part to the system obtained in the previous row. This way helps to illustrate the compound effect of these parts. Here we adopt the 6-layer 6-path model for study. In the first two rows, different paths in the same layer are added with the fixed weights (1/6 in the + Multi-Path model and 1/√6 in the + PathNorm model). We can see that the + Multi-Path model significantly surpasses the baseline model. However, the + PathNorm model performs slightly worse than the + Multi-Path model. In order to verify the importance of PathNorm in our method, we conduct an additional experiment to use learnable weights alone (- PathNorm). As can be seen in Table 4, neither the learnable weights (- PathNorm) nor PathNorm (+ PathNorm) works well alone, which verifies the importance of the combination of learnable weights and PathNorm (+ Learnable Weights) in our method.

5.3 Training Study

We plot the training and validation loss curves of different systems with the same number of parameters, including the deep model, the original multi-path model (Multi-Path) and our model without/with more features (+ Ours/+ More Features), for studying their convergence. All these four systems have been shown in Table 2. We can see that all systems converge stably. The original multi-path model has a higher loss than other models in both the training and validation sets and it does perform the worst. The deep model has the lowest loss, but the performance is close to + Ours and + More Features, which means that the loss cannot absolutely reflect the model performance.

5.4 Learnable Weights

As the learnable $\alpha$ is considered to be a way of measuring the importance of different paths, the difference of $\alpha$ from different paths can be seen as the diversities among these paths. Fig. 7 studies the behavior of $\alpha$, the solid lines denote the absolute value of the difference of $\alpha$. Here we adopt the
24-layer 2-path Transformer system to study $\alpha$ in deep models, we let $|d| = |\alpha_1 - \alpha_2| / (|\alpha_1 + \alpha_2|)$ denote the above mentioned absolute value of difference.

As can be seen in Fig. 7, either in the attention layer or the feed-forward layer, $|d|$ changes significantly in different model depths. In the feed-forward layer, except for the first and last several layers (e.g., 1, 22, and 23), the value of $|d|$ is smaller than the attention layer, which reflects from the side that the diversity of the feed-forward layer is smaller than the attention layer. In the attention layer, the value of $|d|$ is larger in the middle layers (e.g., from 4 to 20), indicating that more model diversities can be learned in the middle layers than the bottom and top layers.

5.5 Training Efficiency

Here we record the computation times required per 100 training steps of different models. To exclude the influence of data transfer, we train these models on a single GPU. Since each path is computed independently, the multi-path structure adopted in this paper has the inherent advantage of high computational parallelism. However, due to the limitations of related computational libraries, this kind of model does not achieve its ideal efficiency as can be seen in Fig. 8. As one type of model structure with great potential, the multi-path model should get more attention from us, and the related computational libraries should also be completed.

6 Discussion

Model depth or width which is more important becomes a hot topic in recent years (Nguyen et al., 2021; Vardi et al., 2022; Eldan and Shamir, 2016; Lu et al., 2017; Cheng et al., 2016). In general, one model can benefit more from increasing the depth (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015), the reasons can be summarized as follows: 1) Expressivity, deep models have better non-linear expressivity to learn more complex transformations. 2) Efficiency, both the number of parameters and the computational complexity will be changed quadratically corresponding to the model width (referring to scaling the matrix dimensions) while linearly with the model depth, thus the cost of increasing width is often much higher than that of depth.

For tasks such as computer vision, the model depth can even reach hundreds or thousands of layers (He et al., 2016a; Zagoruyko and Komodakis, 2017). For the Transformer model, Wang et al. (2022) even train a Transformer model with 1,000 layers. However, the training process of deep models is not as simple as scaling the number of layers. When the model becomes too deep, the degradation problem caused by the back propagation will be exposed (He et al., 2016a).

To seek new solutions to further improve large-scale neural networks, here we adopt the parameter-efficient multi-path structure. As shown in Fig. 1, the multi-path models significantly outperform wide models that scale the matrix dimensions. From Fig. 8 and Fig. 9 we can see that, although the multi-path model does not achieve its ideal efficiency because of computational libraries support, it still takes less training cost than wide models. The above discussions show that multi-path is a better option to broaden the model width, and the width of one model is as important as its depth for the purpose of improving capacity.

7 Conclusion

In this work, we construct a sublayer-level multi-path structure to study how model width affects the Transformer model. To better fuse features extracted from different paths, three additional operations mentioned in Section 3 are introduced. The experimental results on 12 machine translation benchmarks validate our point of view that, instead of indefinitely stacking more layers, there should be a balance between the model depth and width to train a better large-scale Transformer.
Acknowledgments

This work was supported in part by the National Science Foundation of China (Nos. 61876035 and 61732005), the China HTRD Center Project (No. 2020AAA0107904), Yunnan Provincial Major Science and Technology Special Plan Projects (Nos. 202002AD080001 and 202103AA080015), National Frontiers Science Center for Industrial Intelligence and Systems Optimization (Northeastern University, China. No. B16009) and the Fundamental Research Funds for the Central Universities. The authors would like to thank anonymous reviewers for their comments.

Limitations

For the limitation of our work, we will discuss it from three aspects.

Non-Ideal Training Efficiency. As discussed in Section 5.5, although the multi-path structure adopted in this paper has an inherent advantage of high computational parallelism, the training efficiency of this kind of model does not achieve its theoretical height. As one type of model structure with great potential, the multi-path network should get more attention from us, and the related computational libraries should also be completed.

Non-Optimal Hyperparameters. Just like the training hyperparameters are quite different among Transformer-base, big and deep systems, the optimal hyperparameters for models with different depths and widths tend to be different. However, due to the limited computing resources, we do not tune but choose the same hyperparameters as the deep models, which may lead to a non-optimal setting.

Very Large-Scale Networks. Limited by the hardware and memory resources, we did not explore very large models with much more layers and paths. All we can do here is provide insights about how to choose a better combination of model depth and width with limited resources.

References


