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

Ye Lin[1], Shuhan Zhou[1], Yanyang Li[2], Anxiang Ma[1], Tong Xiao[1,3]*, Jingbo Zhu[1,3]
[1]NLP Lab, School of Computer Science and Engineering,
Northeastern University, Shenyang, China
[2]The Chinese University of Hong Kong, Hong Kong, China
[3]NiuTrans 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.

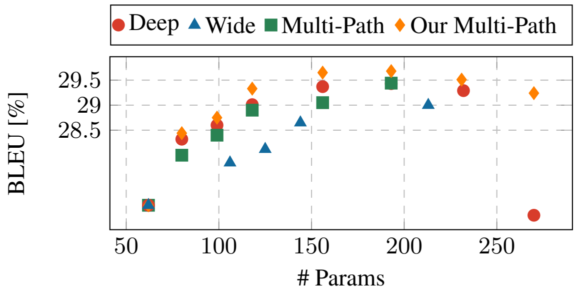

Figure 1: Performance (BLEU) vs. the number of parameters (M) on WMT14 En→De.

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,

*Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 5646–5656
December 7-11, 2022 ©2022 Association for Computational Linguistics

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 $\mathrm{MHA}(\cdot)$:

$$A_x = \mathrm{SoftMax}(\frac{XW_qW_k^TX^T}{\sqrt{d}}) \quad (1)$$

$$\mathrm{MHA}(X) = A_xXW_v \quad (2)$$

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 $\mathrm{FFN}(\cdot)$:

$$\mathrm{FFN}(X) = \mathrm{ReLU}(XW_1 + b_1)W_2 + b_2 \quad (3)$$

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.

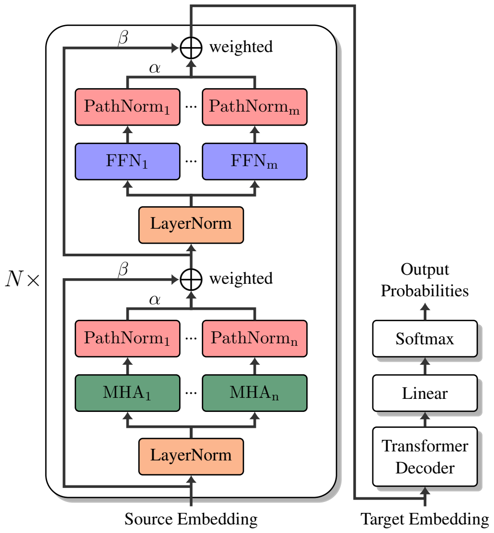

Figure 2: The architecture of the constructed multi-path Transformer model in this paper.

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

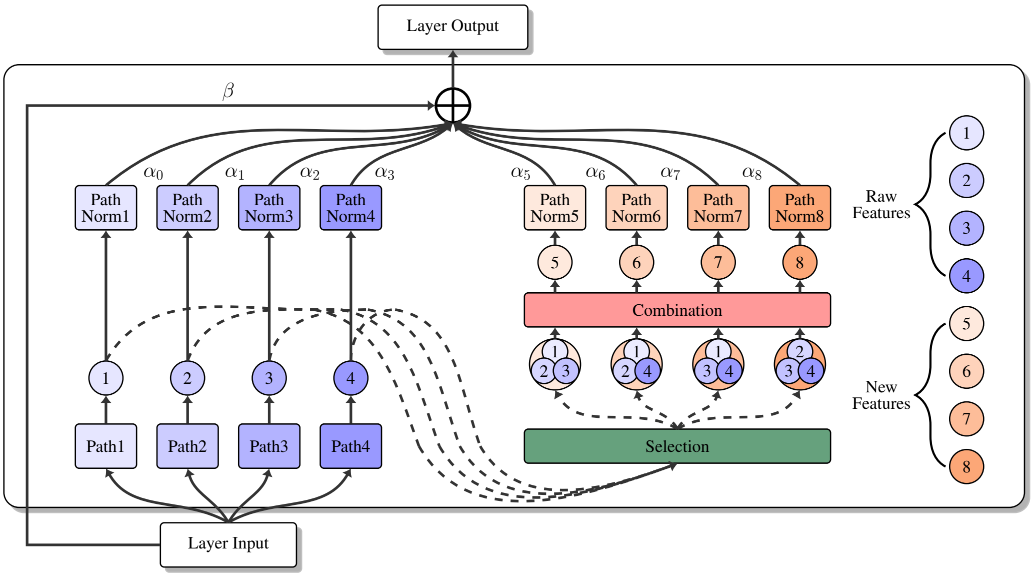

Figure 3: A running example of the process of generating more features from cheap operations.

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 $\mathrm{PathNorm}$) 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 $\mathrm{PathNorm}$ 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 α are added on all model paths, and the residual connections surrounding layers are also equipped with learnable weights β. 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:

$$Y = \beta X + \sum_{i=1}^{n} \alpha_i \mathrm{Path}_i(X) \tag{4}$$

where X denotes the layer input, Y denotes the layer output, n denotes the total number of paths. α and β 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 $\mathrm{MHA}(\cdot)$ and the computation of the feed-forward network in Eq. 3 as $\mathrm{FFN}(\cdot)$. In the multi-head attention or feed-forward network, each path can be computed as:

$$\mathrm{Path}(X) = \mathrm{PathNorm}(\mathrm{Func}(\mathrm{LN}(X))) \tag{5}$$

$$\mathrm{Func}(X) = \mathrm{MHA}(X) \textit{ or } \mathrm{FFN}(X) \tag{6}$$

where $\mathrm{PathNorm}$ 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 $1 \sim 4$ 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:

$$\mathrm{AVG}(X) = \frac{\sum_{i=1}^{k} \mathrm{Func}_i(X)}{k} \tag{7}$$

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 $\mathrm{PathNorm}$ on both the "raw features" and "new features". Besides, learnable weights α and β 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.

Source	Task	Train		Valid		Test	
		sent.	word	sent.	word	sent.	word
WMT14	En↔De	4.5M	220M	3000	110K	3003	114K
	En↔Fr	35M	2.2B	26K	1.7M	3003	155K
WMT17	En↔De	5.9M	276M	8171	356K	3004	128K
	En↔Fi	2.6M	108M	8870	330K	3002	110K
	En↔Lv	4.5M	115M	2003	90K	2001	88K
	En↔Ru	25M	1.2B	8819	391K	3001	132K

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

3.3 The Initialization of α and β

In Section 3.1, we have introduced the learnable weights α and β for the path outputs and residual connections respectively. Here we introduce how to initialize α and β in this work. Since the result of one path coupled with $\mathrm{PathNorm}$ 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 $\alpha = \frac{1}{\sqrt{2n}}$, 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 × 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.

Datasets. For the En↔De tasks (4.5M pairs), we choose *newstest-2013* as the validation set and *newstest-2014* as the test set. We share the source and target vocabularies. For the En↔Fr tasks (35M pairs), we validate the system on the combination of newstest-2012 and newstest-2013, and test it on newstest-2014. We use the concatenation of all available preprocessed validation sets in WMT17 datasets as our validation set. All WMT datasets are provided within the official website[1].

For all datasets, we tokenize every sentence using the script in the Moses[2] toolkit and segment

[1] http://statmt.org/

[2] https://github.com/moses-smt/mosesdecoder/blob/maste

System	Params	Depth	Path	Test	Δ_{BLEU}	Valid	Δ_{BLEU}	Δ_{Average}
Transformer-base	62M	6	1	27.00	+0.00	25.88	+0.00	+0.00
Deep12 ✓	**80M**	**12**	**1**	28.32	+1.32	**26.65**	**+0.77**	**+1.05**
Multi-Path2	80M	6	2	28.00	+1.00	26.22	+0.34	+0.67
+ Ours	80M	6	2	**28.44**	**+1.44**	26.43	+0.55	+1.00
Deep24	118M	24	1	29.01	+2.01	26.94	+1.06	+1.54
Multi-Path2	118M	12	2	28.90	+1.90	26.67	+0.79	+1.35
+ Ours ✓	**118M**	**12**	**2**	**29.33**	**+2.33**	**27.23**	**+1.35**	**+1.84**
Multi-Path4	118M	6	4	28.25	+1.25	26.18	+0.30	+0.78
+ Ours	118M	6	4	29.04	+2.04	26.80	+0.92	+1.48
+ More Features	118M	6	4	29.19	+2.19	26.96	+1.08	+1.64
Deep36	156M	36	1	29.37	+2.37	**27.14**	**+1.26**	+1.82
Multi-Path3	156M	12	3	29.05	+2.05	26.69	+0.81	+1.43
+ Ours	156M	12	3	29.08	+2.08	26.93	+1.05	+1.57
+ More Features	156M	12	3	29.20	+2.20	27.05	+1.17	+1.69
Multi-Path6	156M	6	6	28.87	+1.87	26.58	+0.70	+1.29
+ Ours	156M	6	6	29.13	+2.13	26.93	+1.05	+1.59
+ More Features ✓	**156M**	**6**	**6**	**29.65**	**+2.65**	26.89	+1.01	**+1.83**
Deep48	193M	48	1	29.43	+2.43	27.12	+1.24	+1.84
Multi-Path2	193M	24	2	29.44	+2.44	27.05	+1.17	+1.81
+ Ours ✓	**193M**	**24**	**2**	**29.68**	**+2.68**	**27.41**	**+1.53**	**+2.11**
Multi-Path4	193M	12	4	29.04	+2.04	26.73	+0.85	+1.45
+ Ours	193M	12	4	29.46	+2.46	27.03	+1.15	+1.81
+ More Features	193M	12	4	29.56	+2.56	27.00	+1.12	+1.84
Multi-Path8	193M	6	8	28.62	+1.62	26.71	+0.83	+1.23
+ Ours	193M	6	8	29.21	+2.21	27.07	+1.19	+1.70
+ More Features	193M	6	8	29.56	+2.56	26.95	+1.07	+1.82

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".).

every word into subword units using Byte-Pair Encoding (Sennrich et al., 2016). The number of the BPE merge operations is set to 32K in all these tasks. In addition, we remove sentences with more than 250 subword units (Xiao et al., 2012) and evaluate the results using multi-bleu.perl[3].

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

r/scripts/tokenizer/tokenizer.perl

[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

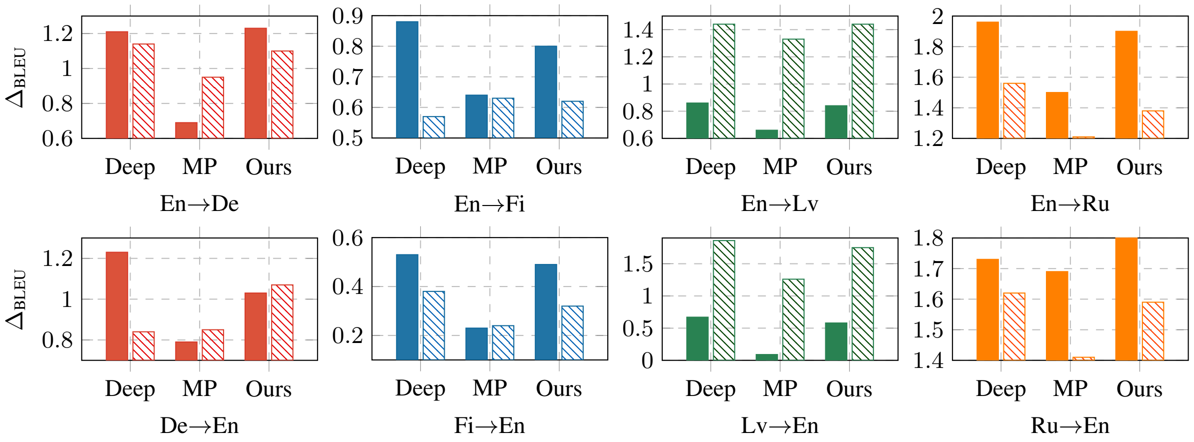

Figure 4: Comparisons of different systems on WMT17 tasks (The figure shows Δ_{BLEU} of each system exceeds the baseline model. Deep, MP, and Ours denote results of deep models, original multi-path models, and our multi-path models. Blocks with/without dotted lines represent results on validation/test set. Different colors 🟥🟦🟩🟧 denote tasks of En↔De, En↔Fi, En↔Lv and En↔Ru.).

	System	Params	D-P	Test	Valid	Δ_{Average}
WMT14 De-En	Baseline	62M	6-1	30.50	30.34	+0.00
	Deep24 ✓	**118M**	**24-1**	31.92	**31.37**	**+1.23**
	Multi-Path2	118M	12-2	31.62	31.00	+0.89
	+ Ours ✓	**118M**	**12-2**	**32.00**	31.29	**+1.23**
	Multi-Path4	118M	6-4	31.52	30.80	+0.74
	+ Ours	118M	6-4	31.85	30.95	+0.98
	+ More Features	118M	6-4	31.89	31.03	+1.04
WMT14 En-Fr	Baseline	111M	6-1	40.82	46.80	+0.00
	Deep24	168M	24-1	42.40	48.41	+1.60
	Multi-Path2	168M	12-2	42.40	48.37	+1.58
	+ Ours ✓	**168M**	**12-2**	**42.44**	**48.45**	**+1.64**
	Multi-Path4	168M	6-4	41.76	47.93	+1.04
	+ Ours	168M	6-4	41.90	48.19	+1.24
	+ More Features	168M	6-4	42.32	48.37	+1.54
WMT14 Fr-En	Baseline	111M	6-1	36.33	47.03	+0.00
	Deep24	168M	**loss exploding ✗**			
	Multi-Path2	168M	12-2	38.19	48.02	+1.43
	+ Ours ✓	**168M**	**12-2**	38.24	**48.45**	**+1.67**
	Multi-Path4	168M	6-4	37.94	48.18	+1.38
	+ Ours	168M	6-4	38.20	48.30	+1.57
	+ More Features	168M	6-4	**38.26**	48.36	+1.63

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.).

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 $C_n^{n-1} = n$ 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[4]) 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-

[4] https://github.com/pytorch/fairseq/blob/v0.6.2/fairseq/optim/fp16_optimizer.py

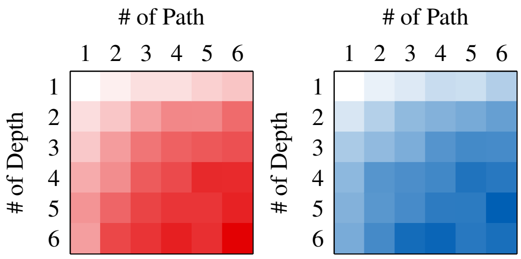

Figure 5: Results of shallower networks with different depths and paths on WMT14 En→De (Darker color means better performance.).

System	Params	BLEU	
		Test	Valid
Baseline	62M	27.00	25.88
+ Multi-Path	156M	28.87	26.58
+ PathNorm	156M	28.72	26.28
+ Learnable Weights	156M	29.13	26.93
- PathNorm	156M	28.86	26.58
+ More Features	156M	**29.65**	**26.89**

Table 4: Ablation study on WMT14 En→De.

mance than increasing the number of depths (e.g., the 1-layer 6-path model vs. the 6-layer 1-path model). When the model is deeper, increasing the number of paths has a greater advantage (e.g., the 2-layer 5-depth model vs. the 5-layer 2-depth models). In most instances, changing the depth and path have almost the same effect on model performance.

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 $1/\sqrt{6}$ 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.

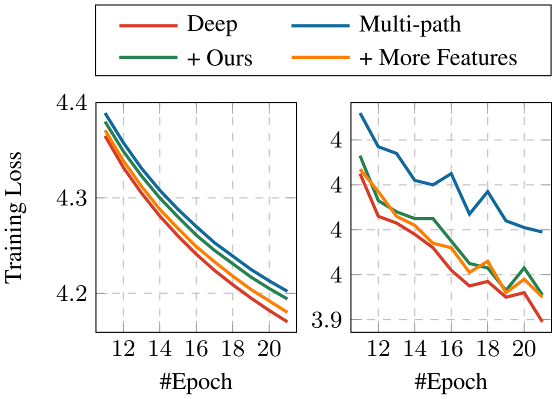

Figure 6: Loss vs. the number of epochs on WMT14 En-De (The left figure plots the training losses, the right figure plots the validation losses.).

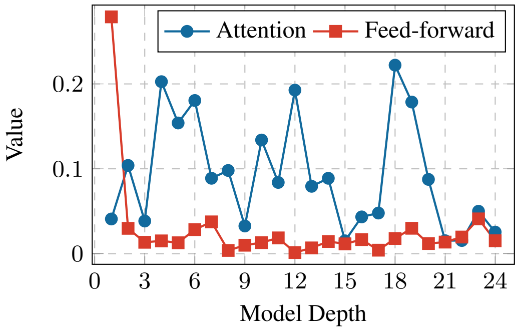

Figure 7: The diversities among different paths vs. the model depth on WMT14 En→De.

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 α is considered to be a way of measuring the importance of different paths, the difference of α from different paths can be seen as the diversities among these paths. Fig. 7 studies the behavior of α, the solid lines denote the absolute value of the difference of α. Here we adopt the

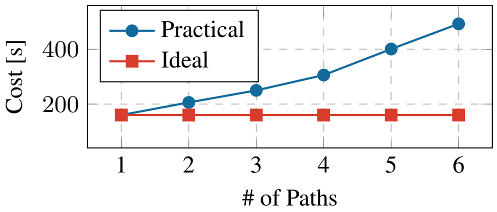

Figure 8: Training cost vs. the number of paths on WMT14 En→De.

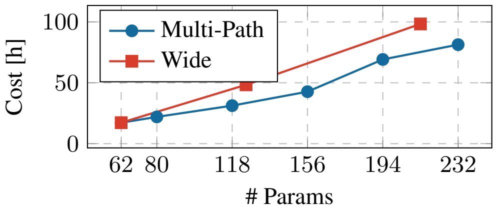

Figure 9: Training costs of different systems on WMT14 En→De.

24-layer 2-path Transformer system to study α in deep models, we let $|d| = \frac{|\alpha_1 - \alpha_2|}{|\alpha_1 + \alpha_2|}$ 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

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted transformer network for machine translation. *CoRR*, abs/1711.02132.

Karim Ahmed and Lorenzo Torresani. 2017. Connectivity learning in multi-branch networks. *CoRR*, abs/1709.09582.

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

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 7–10. ACM.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Ronen Eldan and Ohad Shamir. 2016. The power of depth for feedforward neural networks. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 907–940. JMLR.org.

Yang Fan, Shufang Xie, Yingce Xia, Lijun Wu, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2020. Multi-branch attentive transformer. *CoRR*, abs/2006.10270.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114.

Bei Li, Ziyang Wang, Hui Liu, Quan Du, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2020. Learning light-weight translation models from deep transformer. *CoRR*, abs/2012.13866.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020a. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5747–5763. Association for Computational Linguistics.

Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. 2020b. Very deep transformers for neural machine translation. *CoRR*, abs/2008.07772.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. 2017. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6231–6239.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. 2021. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. *CoRR*, abs/1910.05895.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Sam Shleifer, Jason Weston, and Myle Ott. 2021. Normformer: Improved transformer pretraining with extra normalization. *CoRR*, abs/2110.09456.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society.

Matus Telgarsky. 2015. Representation benefits of deep feedforward networks. *CoRR*, abs/1509.08101.

Gal Vardi, Gilad Yehudai, and Ohad Shamir. 2022. Width is less important than depth in relu neural networks. *CoRR*, abs/2202.03841.

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 *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. 2022. Deepnet: Scaling transformers to 1, 000 layers. *CoRR*, abs/2203.00555.

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 *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1810–1822. Association for Computational Linguistics.

Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. 2012. Niutrans: An open source toolkit for phrase-based and syntax-based machine translation. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the System Demonstrations, July 10, 2012, Jeju Island, Korea*, pages 19–24. The Association for Computer Linguistics.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR.

Sergey Zagoruyko and Nikos Komodakis. 2017. Diracnets: Training very deep neural networks without skip-connections. *CoRR*, abs/1706.00388.

Sergey Zagoruyko, Adam Lerer, Tsung-Yi Lin, Pedro Oliveira Pinheiro, Sam Gross, Soumith Chintala, and Piotr Dollár. 2016. A multipath network for object detection. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press.

Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander J. Smola. 2020. Resnest: Split-attention networks. *CoRR*, abs/2004.08955.

Hongyang Zhang, Junru Shao, and Ruslan Salakhutdinov. 2019. Deep neural networks with multi-branch architectures are intrinsically less non-convex. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 1099–1109. PMLR.