IMPROVING MACHINE TRANSLATION BY SEARCHING SKIP CONNECTIONS EFFICIENTLY

Anonymous authors
Paper under double-blind review

ABSTRACT

As a widely used neural network model in NLP (neural language processing), transformer model achieves state-of-the-art performance in several translation tasks. Transformer model has a fixed skip connection architecture among different layers. However, the influence of other possible skip connection architectures are not discussed completely in transformer model. We search different architectures of skip connection to discover better architectures in different datasets. To improve the efficiency of trying different skip connection architectures, we apply the idea of network morphism to add skip connections as a procedure of fine-tuning. Our fine-tuning method outperforms the best models trained by the same or smaller datasets in WMT'16 En-De, WMT'14 En-Fr and WMT'18 En-De with 226M back-translation sentences. We also make experiment on transferring searched skip connection architectures to new transformer models.

1 INTRODUCTION

Neural architecture search (Baker et al., 2017; Zoph & Le, 2017) is an idea of automatically searching hyper-parameters of neural networks. There are different methods such as reinforcement learning and evolutionary algorithms for automatically designing CNN architectures. For neural language processing, So et al. (2019) propose an evolutionary method to search different transformer architectures.

However, when searching an architecture, training a new model from the beginning is usually required. These methods are very costly such as the method in Baker et al. (2017); Zoph & Le (2017) (requiring hundreds or thousands of GPU days). Network morphism Elsken et al. (2018); Chen et al. (2016); Wei et al. (2016); Cai et al. (2018) is a neural architecture search method which dramatically reduces these computational costs while still achieving competitive performance in CNN model.

The skip connections are important hyper-parameters in transformer model (Vaswani et al., 2017). A transformer model composes of encoder layers, decoder layers, residual connections inside encoder layers and decoder layers, the encoder-decoder attention connections between the last encoder layer and decoder layers. However, other works (Wu et al., 2019b; Guo et al., 2019) imply that different skip connection architectures can influence the performance of transformer model.

We focus on applying network morphism to transformer model in machine translation tasks. Specifically, our contributions are as follows:

- We apply network morphism in searching different skip connection architectures in machine translation tasks. We add skip connections (residual connections and encoder-decoder attention connections) to transformer models by network morphism. The skip connections are added by hill climbing strategy, which searches different connections step by step.
- We initialize transformer models by released pre-trained transformer parameters. The pre-trained models include transformer models trained by WMT'16 En-De and WMT'14 En-Fr bitext datasets described in Ott et al. (2018). We also use the transformer model trained by WMT'18 En-De bitext dataset as well as 226M newscrawl back-translation dataset described in Edunov et al. (2018a). Our method works as a fine-tuning process, which outperforms the best models trained by the same or smaller datasets by 0.3 to 1 BLEU.
- As a a fine-tuning process, our method requires few computing resources. We propose faster gradient trick to accelerate the adjustment of network morphism. Our method can reach its

peak by only using 0.05-0.39 of total training data while training a new model needs all training data. Our method is more efficient with larger training dataset. Since our method is simple, it can be applied to other existed models efficiently.

- To figure out whether the performance of skip connections is related to the model weights and datasets, we make experiment on transferring the architecture searched by network morphism to new transformer models. We extract the skip connection architectures of two fine-tuned models in two datasets, IWSLT'14 De-En and WMT'18 En-De with back-translation. Then we apply these skip connection architectures to new transformer models trained in IWSLT'14 De-En dataset. We find the architecture searched in IWSLT'14 De-En works better than the architecture searched in WMT'18 En-De for the models in IWSLT'14 De-En dataset. Besides, transferring a skip connection as a dynamic connection works better than adding them as a static connection (the connection created by network morphism is dynamic).

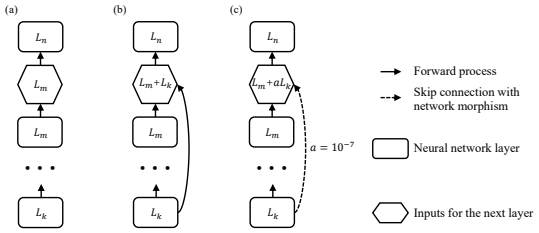

Figure 1: The structure of (a) Sequential layers (b) Sequential layers with added (residual) skip connection (c) Sequential layers with added skip connection by network morphism. Solid lines denote forward process, including added skip connection. Dashed line denotes the added skip connection by network morphism. The hexagons denote the inputs of layer L_n.

2 NETWORK MORPHISM IN TRANSFORMER

2.1 BACKGROUND

Let $f^w(x)$ denote a neural network with input $x \in \mathcal{X}$. Then a network morphism is finding another neural network $g^{\tilde{w}}(x)$ where:

$$f^w(x) = g^{\tilde{w}}(x), \qquad \forall x \in \mathcal{X} \tag{1}$$

Network morphism is an idea that a well-trained neural network can be replaced by another neural network with different structure. With this replacement, a neural network with different structure, $g^{\tilde{w}}(x)$, can inherit the performance of a trained neural network, $f^w(x)$, and try the performance of the new structure of $g^{\tilde{w}}(x)$ with further training: In forward process, $g^{\tilde{w}}(x)$ and $f^w(x)$ make the same output; In backpropagation, the gradients of parameters $\tilde{w}$ are different with the gradients of w since $g^{\tilde{w}}$ and f^w are different.

Network morphism has three usual applications: Make the network deeper such as multiplying a network with identity matrices; Make the network wider such as dividing the weight matrix to several parts in a network; Add a skip connection, which is the aim of this paper.

In this paper, the application of skip connection of network morphism is inspired by Network morphism Type II in Elsken et al. (2018):

Let ${f_i}^{w_i}(x)$ denote some part of a neural network $f^w(x)$ and assume ${f_i}^{w_i}(x)$ has the form ${f_i}^{w_i}(x) = Ah^{w_h}(x) + b$ for an arbitrary function h. We replace ${f_i}^{w_i}$, $w_i = (w_h, A, b)$, by

$$\tilde{f_i}^{\tilde{w_i}}(x) = (A\ \tilde{A}) \begin{pmatrix} h^{w_h}(x) \\ \tilde{h}^{w_{\tilde{h}}}(x) \end{pmatrix} + b \tag{2}$$

The parameters of $\tilde{f_i}^{\tilde{w_i}}$ become $\tilde{w_i} = (w_h, w_{\tilde{h}}, A, \tilde{A}, b)$ with the added function $\tilde{h}^{w_{\tilde{h}}}$. Then, the requirement of network morphism, Eq.(1), can be satisfied by letting $\tilde{A} = 0$. In practice, we denote $\tilde{A}$ as a small value so that gradient can be transferred to $\tilde{h}^{w_{\tilde{h}}}$ in backpropagation.

2.2 ADDING A SKIP CONNECTION

The differences between different skip connection methods are shown in Figure 1. (a) corresponds to a trained model without skip connection; (b) corresponds to adding a residual skip connection to the trained model in (a); (c) corresponds to adding a network morphism skip connection to the trained model in (a). The input node, which refers to hexagon in the figure, reflects the differences between two methods in adding skip connection. In (b), the input of layer L_n after adding a residual skip connection becomes:

$$I_R = L_m + L_k \tag{3}$$

where I_R denotes the input of L_n with a residual skip connection; L_m and L_k denote the output of corresponding layers. To save computing resources, initializing by a trained model is better than training a new model with new architecture. However, the input of layer L_n after adding a residual skip connection becomes $L_m + L_k$, which is different from the input without adding connection, $I = L_m$. That means adding a residual skip connection to a trained model will change a layer's input in forward process. Since the input $I = L_m$ is a well-performed layer input of the trained model in Figure 1 (a) and there is no guarantee that $I_R = L_m + L_k$ also performs well, the method in Eq.(3) can't retain the performance of a trained model.

As a contrast, the input of layer L_n after adding a skip connection by network morphism are very similar to the layer input of trained model:

$$I_M = L_m + aL_k \tag{4}$$

$$I_M \approx L_m \tag{5}$$

Parameter a, corresponding to $\tilde{A}$ in Eq.(2), is a small value such as 10^{-7} so that I_M is approximately equal to $I = L_m$. Since a neural network can be viewed as a continuous function, the final predictions of a model before and after adding skip connections by Eq.(4) are also approximately equal.

2.3 FASTER GRADIENT TRICK

There is a weakness of adding a skip connection by network morphism: The effect of a skip connection lies in how much it influences the input of the connected layer. In network morphism where skip connection is initialized with a small value a, the influence of an added skip connection is also small. In order to expand the influence of a skip connection at a faster speed, we apply faster gradient trick which multiplies the gradient of a by a factor F:

$$a = a + FG(a) \tag{6}$$

where $G(a)$ corresponds to the value that a will be added in backpropagation, which also considers learning rate. Faster gradient trick accelerates the changes of a by F times in backpropagation, which enables the added skip connection to influence the model prediction apparently. Especially, the learning rate decreases when training a transformer model so when adding a skip connection to a trained transformer model by network morphism, the learning rate should be initialized by a small value. In a word, the small initialized value and small learning rate of a restrict the influence of added skip connections so the training needs to be accelerated by Eq.(6). F is set to 60 in our experiments.

After the added skip connection has enough influence on the model, the faster gradient trick should be disabled in order to fine adjust parameter a. As shown in Algorithm 1, the condition to stop faster gradient trick is enough training step s. If the enough influence of an added skip connection corresponds to a large enough $|a|$, the condition to stop faster gradient trick corresponds to a large enough $|a| + S\sum|FG(a)|$. Since $|a| + S\sum|FG(a)| \geq |a + S\sum FG(a)|$, the condition implies if a has appropriate gradients, it will have large value with faster gradient trick. However, if a is not supported by its gradients (e.g., the gradients offset each other or the gradients are small), the skip connection won't have much influence. By faster gradient trick and the trainable factor a, only the connections that are supported by backpropagation can have enough influence.

2.4 SKIP CONNECTION IN TRANSFORMER

There are three kinds of skip connections in transformer models: The residual connection inside an encoder layer, the residual connection inside a decoder layer and the encoder-decoder attention connection between an encoder layer and a decoder layer (Vaswani et al., 2017). The residual

Algorithm 1 Applying faster gradient trick to parameter a

Initialize network morphism skip connection factor $a = 10^{-7}$.
# Start training.
for training step $s = 1$ **to** L **do**
  For other parameters: $P = P + G(P)$
  if $s \leq S$ **then**
    # S times of training should have enough influence on a.
    $a = a + FG(a)$
  else
    $a = a + G(a)$
  end if
end for

connection inside an encoder or decoder layer is contained in layer normalization, which can be denoted by the equation:

$$y = \text{LayerNorm}(I + \text{Sublayer}(I)) \tag{7}$$

where Sublayer(I) is the function implemented by the sublayer in an encoder or decoder layer. I is the input of an encoder or decoder layer and y is the output.

Inspired by the residual connection inside an encoder (decoder) layer, we add skip connection between two encoder (decoder) layers based on Eq.(4). Denote L_k as the outputs of k_{th} encoder (decoder) layer. Initializing a skip connection from k_{th} encoder (decoder) layer to l_{th} encoder (decoder) layer ($k < l$) by network morphism can be described as:

$$\hat{y} = \text{LayerNorm}((I + aL_k) + \text{Sublayer}(I + aL_k)) \tag{8}$$

where I is the inputs of l_{th} layer without the new skip connection from k_{th} encoder (decoder) layer and $\hat{y}$ denotes the outputs of l_{th} layer after adding the skip connection.

The encoder-decoder attention connection is contained in multi-head attention of transformer model. Multi-head (MH) attention can be denoted by:

$$\text{MH}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ...)W^O \tag{9}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{10}$$

where Q, K, V denote query, memory key and value vectors in attention mechanism. W_i^Q, W_i^K, W_i^V, W^O are parameter matrices. In an encoder-decoder attention connection, the query vectors come from the previous decoder layer; The memory key and value vectors come from the output of the final encoder layer.

The encoder-decoder attention connection in this paper can be initialized from any encoder layer to any decoder layer. Denote L_k as the outputs of the encoder layer in a encoder-decoder attention connection, which connects to l_{th} decoder layer. A new encoder-decoder attention connection can be initialized by network morphism:

$$\hat{K} = K + aL_k \tag{11}$$

$$\hat{V} = V + aL_k \tag{12}$$

where K, V are the key and value vectors in multi-head attention in l_{th} decoder layer without the new encoder-decoder attention connection; $\hat{K}, \hat{V}$ are the key and value vectors after initialized a new skip connection from k_{th} encoder layer to l_{th} decoder layer.

3 Experimental Setup

3.1 Datasets

We run experiments on three pre-trained models, based on two WMT English to German (En-De) datasets and a WMT English to French (En-Fr) dataset. For bitext WMT'16 En-De, we replicate the setup of Ott et al. (2018), consisting of WMT'16 training data with 4.5M sentence pairs. Following

Table 1: Tokenized BLEU on various test sets of WMT'18 En-De when adding skip connections by network morphism compared to back-translation sentence pairs obtained by various generation methods.

Model	News2013	News2014	News2015	News2016	News2017	Average
Bitext	27.84	30.88	31.82	34.98	29.46	31.00
+beam	27.82	32.33	32.20	35.43	31.11	31.78
+greedy	27.67	32.55	32.57	35.74	31.25	31.96
+top10	28.25	33.94	34.00	36.45	32.08	32.94
+sampling	28.81	34.46	34.87	37.08	32.35	33.51
+beam+noise	**29.28**	33.53	33.79	37.89	32.66	33.43
+sampling+network morphism (ours)	29.16	**35.39**	**35.34**	**38.54**	**33.68**	**34.42**

the setup of Ott et al. (2018), we validate on newstest2013 and test on newstest2014. The vocabulary is a 32K joint source and target byte pair encoding (BPE; Sennrich et al., 2016).

For bitext WMT'14 En-Fr, we replicate the setup of Ott et al. (2018) with 36M training sentence pairs. We validate on newstest2012-2013 and test on newstest2014. The BPE vocabulary is with 40K types.

For WMT'18 En-De with back-translation, we replicate the setup of Edunov et al. (2018a), including 5.18M En-De bitext sentence pairs where the sentences longer than 250 words as well as sentence-pairs with a source/target length ratio exceeding 1.5 are removed. The dataset also contains the back-translation monolingual data with 226M sentences after removing duplicates. The BPE vocabulary is with 35K types. We validate on newstest2012 and test on newstest2013-2017.

IWSLT'14 De-En dataset is also used in the paper, where we replicate the setup of Edunov et al. (2018b) for test dataset and 10K joint BPE vocabulary.

The majority of results in this paper are in terms of case-sensitive tokenized BLEU. We only measure detokenized BLEU on IWSLT'14 De-En.

3.2 Models and Hyper-parameters

Since network morphism is a method to improve a trained model, we use the pre-trained transformer models released in the fairseq-py toolkit (Edunov et al., 2017)[1], which correspond to the models in Ott et al. (2018); Edunov et al. (2018a). Then we add skip connections to these models by network morphism and fine-tunes the changed models. The big transformer models in WMT datasets share some hyper-parameters, i.e. big transformer architecture with 6 blocks in the encoder and decoder, word representations of size 1024, feed-forward layers with inner dimension 4096, 16 attention heads in multi-head attention. Models are optimized with Adam (Kingma & Ba, 2015) using $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 1e-8$. All models use label smoothing with 0.1 weight (Szegedy et al., 2016; Pereyra et al., 2017).

For WMT'16 En-De and WMT'14 En-Fr, we use the pre-trained models with the setup of Ott et al. (2018). For WMT'18 En-De, there are five pre-trained models released with the setup of Edunov et al. (2018a) and we randomly select the third model for experiment, where the pre-trained model is trained by En-De sentence pairs as well as monolingual back-translated sentence pairs and we only use En-De sentence pairs for fine-tuning. The learning rate starts from $3e-6$ in fine-tuning process.

Since there are about 3×10^{23} possible skip connection architectures, searching all architectures is not practical and we use hill climbing strategy to randomly search appropriate skip connections, which is shown in Appendix.

4 Results

4.1 Fine-tuning Results

We first report results on WMT'18 En-De with 226M back-translated sentences where we compare our results to the baselines in Edunov et al. (2018a), which is also the winner of WMT'18. Table 1 provides results on a wide range of test sets (newstest 2013-2017). Bitext model is only trained

[1] https://github.com/pytorch/fairseq

Table 2: Tokenized BLEU on newstest2014 for WMT English-German (En-De) and English-French (En-Fr). a-h use only WMT bitext (WMT'14, except for b,c,e,f,g which train on WMT'16 in En-De). In En-De, baseline i uses WMT'18 bitext dataset as well as 226M back-translation sentences.

	Model	Param (En-De)	En-De	En-Fr
Bitext	a.Gehring et al. (2017)	216M	25.2	40.5
	b.Vaswani et al. (2017)	213M	28.4	41.0
	c.Ahmed et al. (2017)	213M	28.9	41.4
	d.Chen et al. (2018)	379M	28.5	41.0
	e.Shaw et al. (2018)	-	29.2	41.5
	f.Ott et al. (2018)	210M	29.3	43.2
	g.Wu et al. (2019a)	213M	29.7	43.2
	h.Wu et al. (2019b)	-	29.9	43.3
	Our result	210M	**30.2**	**44.3**
Back-translation	i.Edunov et al. (2018a)	213M	35.0	45.6
	Our result	213M	**35.4**	-

by bitext data. Other baselines refer to different methods of generating back-translated sentences, including beam search with beam size 5, greedy search, restricting sampling to the 10 highest scoring outputs, unrestricted sampling from the model distribution and adding noise to the beam outputs. Our network morphism fine-tuning model is based on the trained model where back-translated sentences are generated by sampling method.

The experiments show that skip connections added by network morphism can improve the performance of transformer model trained by additional back-translation dataset. Although we only use the bitext data for fine-tuning, the network morphism fine-tuning model outperforms baseline back-translation models on various test datasets except newstest 2013.

We also report results on WMT newstest2014 En-De and En-Fr where we compare to the best results trained by the same or smaller datasets. Table 2 shows that our network morphism fine-tuning models outperform all baselines by 0.3 to 1 BLEU, including the models trained by bitext sentences and model trained by En-De back-translation sentences. Our work doesn't include the En-Fr model trained by back-translation sentences since the corresponding pre-trained model hasn't been released. The results show that with skip connections added by network morphism, the models reach the best results on same or smaller datasets.

We also notice that the released pre-trained models have few differences with the results in Ott et al. (2018); Edunov et al. (2018a). To better show the performances of skip connections added by network morphism, we report the fine-tuning improvements in Table 3. As shown in Table 3, the pre-trained models on WMT'16 En-De and WMT'14 En-Fr have better performances than they are reported in Ott et al. (2018). On the other hand, the performance of pre-trained model on WMT'18 En-De is lower than the result in Edunov et al. (2018a), probably because Edunov et al. (2018a) uses the average of different checkpoints and we only select one checkpoint for fine-tuning.

The fine-tuning of skip connections added by network morphism improves WMT'18 En-De with back-translation model by 0.9 BLEU, WMT'16 En-De by 0.4 BLEU and WMT'14 En-Fr by 0.2 BLEU. The differences maybe due to the utilization of different models on different datasets. In common sense, the models with similar model structure and same dataset have an upper bound of evaluation score. Comparing BLEU of pre-trained model in Table 3 to the baselines in Table 2, we find that the WMT'16 En-De pre-trained model has nearly the best performance in 8 baselines, which indicates the potential improvement of fine-tuning is not so significant. The WMT'14 En-Fr pre-trained model outperforms the best of model in 8 baselines by 0.8 BLEU, which indicates the pre-trained model takes nearly full advantage of the dataset and there is little potential improvement of fine-tuning. For WMT'18 En-De with back-translation, the pre-trained model has lower BLEU than the baseline in Table 2, which indicates the potential improvement of fine-tuning is significant. Besides, the back-translation dataset is about 40 times larger than bitext dataset, which makes the model harder to take full advantage of the dataset.

4.2 Time Complexity

Searching time complexity Unlike other neural architecture search methods, network morphism retains model performance while searching, which significantly reduces the searching time. For a

Table 3: Tokenized BLEU of adding skip connections by network morphism on three pre-trained models. The first two models correspond to bitext models in Ott et al. (2018). The third model corresponds to back-translation (BT) model in Edunov et al. (2018a).

	Pre-trained	Network morphism fine-tuning	Δ BLEU
WMT'16 En-De	29.8	30.2	+0.4
WMT'14 En-Fr	44.1	44.3	+0.2
WMT'18 En-De + BT	34.5	35.4	+0.9

Table 4: The sentence numbers that network morphism fine-tuning needs on various datasets. Fine-tuning subset denotes how many sentences are used for fine-tuning. Each sentence is used only once. Total sentence denotes the total sentence number of a dataset. Fine-tuning rate corresponds to the rate when fine-tuning subset number is divided by total sentence number. Back-translation sentences are not used in fine-tuning.

	Fine-tuning subset	Total sentence	Fine-tuning rate
WMT'16 En-De	1.5M	4.5M	0.33
WMT'14 En-Fr	2M	36M	0.05
WMT'18 En-De	2M	5.18M	0.39
En-De BT	-	226M	-

new architecture, other methods need to train the whole model from the beginning while our method works as a fine-tuning for an existed model.

In the experiments, we find that fine-tuning doesn't need the whole dataset to reach the best BLEU in valid set. As shown in Table 4, after adding skip connections to a model, we only needs 1.5M-2M sentences for fine-tuning (train each sentence one time). Comparatively small fine-tuning subset also means we don't need back-translation sentences for fine-tuning the WMT'18 En-De model. Adding skip connections by hill climbing strategy needs to repeat fine-tuning procedure several times. In our experiments with hill climbing strategy, fine-tuning procedure is repeated 9 times, which corresponds to 0.45-3.51 times sentence number of bitext datasets. Since the training epochs can be larger than 20 in training transformer, the time complexity for searching architectures by network morphism is acceptable. We also notice fine-tuning subsets have similar sizes corresponding to different total dataset sizes, which means fine-tuning won't spend more time when training dataset expends. Therefore, when comparing to the models trained by back-translation sentences and WMT'14 En-Fr, the time complexity of fine-tuning is a relatively small value.

Running time complexity Adding new skip connections will make a model more complex but it won't highly influence the speed of running the model. According to Eq.(8, 11-12), a new skip connection only adds one parameter a to the model. Besides, each skip connection corresponds to 1-2 add operation, which doesn't occupy many computing resources.

4.3 Architecture Transfer

The performance of skip connections generalized by network morphism is related to the parameters of the neural network such as weight values. To discover whether the architecture searched from a network works for a different neural network, we make another experiment on IWSLT'14 De-En. We use the small transformer in fairseq-py toolkit hyper-parameters for baseline transformer model.

The added skip connections in a fine-tuned model consist of skip connection relationships such as the sender & receiver of a skip connection and the factor a. Then we can extract only the skip connection relationships of the fine-tuned model and train a new transformer model initialized with that architecture in IWSLT'14 De-En. The transformer models we used in IWSLT'14 De-En consist of 6 encoder and decoder blocks and 512 dim word representation. Since the number of encoder and decoder blocks is the same with the big transformer models used in WMT datasets, the skip connection relationships generated by network morphism in WMT datasets can be transferred to the transformer models in IWSLT'14 De-En. We transfer two architectures, including the architecture searched in WMT'18 En-De dataset with back-translation and the architecture searched in IWSLT'14 De-En dataset. There are also two ways of applying new skip connection relationships to a new model: Adding new connections as dynamic connection and adding them as static connection. Dynamic connection means initializing new connections with a small factor a as it is done in Eq. (4) and a is trainable. Static connection means initializing new connections as it is done in Eq.(3) in classic

transformer model. We also test the improvement of network morphism fine-tuning on the transferred transformer models. The results are shown in Table 5. We provide an example of skip connection architecture in Appendix.

Table 5: Detokenized BLEU of different transformer models on IWSLT'14 De-En. Different rows correspond to different architectures used to train new transferred models in IWSLT'14 De-En. Different columns correspond to different procedure of training new models, including basically training the models with different architectures and fine-tuning new skip connections by network morphism.

	Basic	+ fine-tuning
Transformer	34.6	34.8
+ dynamic connection from IWSLT'14	34.8	34.9
+ static connection from IWSLT'14	34.2	34.3
+ dynamic connection from WMT'18 (BT)	34.4	34.6
+ static connection from WMT'18 (BT)	32.6	32.7

From the results we notice that different datasets have different suitable architectures: The architecture searched in IWSLT'14 De-En improves the new model in IWSLT'14 De-En while the architecture searched in WMT'18 En-De with back-translation decreases the model performances in IWSLT'14 De-En. Besides, adding new connections as dynamic connection works much better than adding them as static connection. Actually adding new connections as static connection decreases the performance of models, probably because the searched skip connections only improve the model when a connection has the factor a with special value and dynamic connection can search a automatically. In addition, some useless skip connections can be disabled with a very small a in dynamic connection, which is supported by the performance of dynamic connection comparing to static connection when applying the relationship searched in WMT'18.

5 Related Works

Our work follows the idea of network morphism (Elsken et al., 2018; Chen et al., 2016; Wei et al., 2016; Cai et al., 2018), which can automatically search several different architectures such as wider network, deeper network and skip connection. The architecture search of network morphism is based on the former network parameters so that it works as fine-tuning and saves computing resources comparing to other neural architecture search methods. We apply network morphism of adding skip connections to transformer model and our method can automatically search different combinations of skip connections efficiently. Similar to Elsken et al. (2018), we also use hill climbing strategy to search skip connections from several candidates. Unlike Elsken et al. (2018), we propose faster gradient trick to accelerate the influence of added skip connections. We also propose the network morphism way of adding encoder-encoder residual connection, decoder-decoder residual connection and encoder-decoder attention connection to a transformer model.

So et al. (2019) also apply the idea of neural architecture search in transformer models, which is called Evolved Transformer. Evolved Transformer searches transformer architectures based on different blocks, which consists of two branches with different layer types (such as convolution) and activations. However, the Evolved Transformer pays less attention to the skip connections between different layers, which is the aim of our work. In Evolved Transformer, parameters of different architectures have no relationship with each other, which means a new architecture needs to be trained from the beginning. In our work, parameters of a new architecture inherits the parameters of the former architecture, which saves most computing resources.

6 Conclusion

To discover the performance of adding skip connections in transformer with high efficiency, we apply the idea of network morphism to transformer skip connection architecture as a method of fine-tuning. Our fine-tuning method outperforms the best models trained by the same or smaller datasets in different situations. We also test the performance of transferred architectures and show that different skip connection architectures suit different datasets. In the future, we want to find a better universal transformer architecture by using larger datasets and with more iterations in searching and transferring architectures.

REFERENCES

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*, 2017.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations*, 2017.

Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations*, 2016.

Sergey Edunov, Myle Ott, and Sam Gross. Fairseq. 2017.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 489–500, Brussels, Belgium, October-November 2018a. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 355–364, 2018b.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks. In *6th International Conference on Learning Representations*, 2018.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 123–135, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1315–1325, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 1–9, Belgium, Brussels, October 2018. Association for Computational Linguistics.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. In *5th International Conference on Learning Representations*, 2017.

Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, 2016.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, 2018.

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

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 564–572, 2016.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2019a.

Lijun Wu, Yiren Wang, Yingce Xia, Fei Tian, Fei Gao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. Depth growing for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5558–5563, Florence, Italy, July 2019b. Association for Computational Linguistics.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations*, 2017.

A Appendix

A.1 Iteration of Searching Architectures

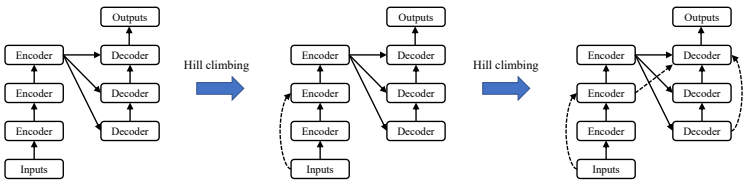

Figure 2: The sketch map of adding skip connections through hill climbing strategy steps. The figure contains a simplified transformer which only has three encoder and decoder layers. At each step, new skip connections are searched and added to the current model structure. Dashed lines denote the skip connections added by network morphism. The three kinds of skip connections are added to the transformer in the figure.

In a transformer model with 6 encoder and decoder layers, there are $2 * \sum_{i=1}^{6} i + 6^2 = 78$ different possible single skip connections. If we don't restrict the number of skip connections each layer receives and sends, there are $2^{78} \approx 3 \times 10^{23}$ different combinations of skip connections. Searching all architectures is not practical, so we apply hill climbing strategy in Russell & Norvig (2002); Elsken et al. (2018) to search a temporarily better model architecture by iteration.

Algorithm 2 Skip connection search by hill climbing

$model_0 \triangleq$ transformer model to start with.
Initialize best current model structure.
$model_{best} = model_0$
# Start hill climbing steps.
for $i = 1$ **to** n_{steps} **do**
  # In each step, try n_{neigh} different model structures.
  for $j = 1$ **to** n_{neigh} **do**
    # Apply network morphism with n_{NM}
    connections. n_E, n_D, n_{ED}=RandomSplit(n_{NM})
    # $n_E + n_D + n_{ED} = n_{NM}$ $model_j$ =AddE($model_{best}, n_E$) $model_j$ =AddD($model_j, n_D$)
    $model_j$ =AddED($model_j, n_{ED}$) $model_j$ =Train($model_j$,data)
  end for
  # Get best model on validation set. $model_{best} = \underset{j=1...n_{neigh},best}{\text{argmax}}(\text{vaild}(model_j))$
end for

As illustrated in Algorithm 2, hill climbing contains n_{steps} steps of search, where each step inherits the best skip connection structure of the last step and adds new skip connections. At each step, hill climbing strategy searches n_{neigh} different new skip connection combinations and selects the best in validation set. Each new skip connection combination contains n_{NM} new skip connections, each of which is randomly split into three parts: n_E, n_D, n_{ED}. AddE, AddD and AddED are three functions that randomly add a number of connections in three kinds of skip connections (encoder-encoder, decoder-decoder and encoder-decoder) by network morphism. Figure 2 also visualizes the adding skip connections procedure through network morphism where model structure becomes more complex step by step. In experiments, we set n_{steps} and n_{neigh} to 3, n_{NM} to 5.

A.2 Skip Connection Architecture

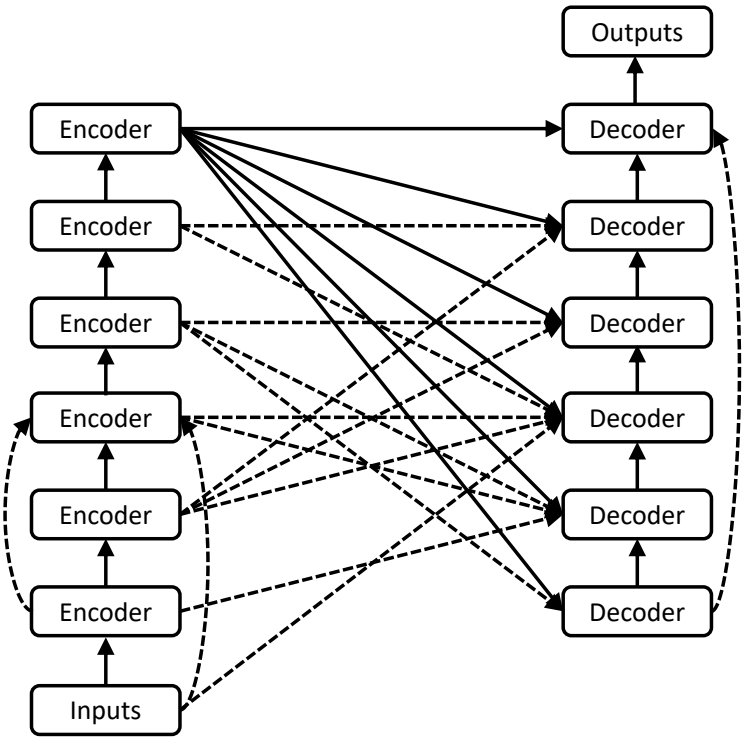

Figure 3: The figure of skip connection architecture trained in IWSLT'14. Dashed lines denote the skip connections added by network morphism.