The Scale-Invariant Space for Attention Layer in Neural Network

Yue Wang[a], Yuting Liu[a,*], Zhi-Ming Ma[b]

[a] *School of Science, Beijing Jiaotong University, Beijing, China*
[b] *Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China*

Abstract

Recently, the Transformer architecture has achieved state-of-the-art performance in many natural language processing tasks. One key component in the Transformer architecture is the attention layer, which captures the relation between tokens. In this paper, we show that the weight of the attention layer has scale-invariant property, i.e. the output is invariant to a rescaling of weights. However, optimization algorithms in the vector space of weight such as SGD are not scaling invariant. This mismatch will potentially hurt the optimization process. To solve the mismatch, we seek a new parameter space for attention layer that is both scale-invariant and can sufficiently represent the output of attention, so that we can employ optimization algorithms in the scale-invariant parameter space. To achieve this goal, we first show that the output of the attention layer can be represented using scale-invariant variables, which is called paths. Then, we define basis paths which are an independent subset of all paths and are sufficient to represent all other paths. We prove that the Scale-Invariant (SI) space for the attention layer is composed of the basis path. Finally, we design an *Attention Basis Path Identification(ABPI) Method* to identify the basis paths and propose optimizing the attention layer directly in its SI space. Several experiments on benchmark datasets show that we can obtain more effective neural networks with the attention layer by optimizing the attention layer directly in its SI space.

Keywords: Transformer, neural network, optimization, scale-invariant, attention layer

1. Introduction

Transformer architecture has been proposed recently and becomes one of the most commonly used neural network architectures in natural language processing. It has achieved state-of-the-art performance in many tasks such as machine translation and language modeling[1, 2, 3, 4, 5]. One key component in the Transformer architectures is the attention layer which can potentially capture the relation between tokens. The Transformer neural network is usually trained with variants of *Stochastic Gradient Descent*(SGD) which updates the neural network by following the negative direction of the gradient w.r.t. weight. Different from the traditional neural networks such as Multilayer Perception(MLP) and Convolutional Neural Network(CNN), there are attention layers in the Transformer[1]. The attention layer models the relation between the different elements in a sequence, called tokens, in a non-linear way and can potentially capture the relation between tokens.

In this paper, we show that the attention layer has scale-invariant property, i.e., the output is invariant to a rescaling transformation of weights. Specifically, if some weights in attention layers are multiplied by a non-zero constant and other weights are divided by this constant, the output will not change for any given input. It means that the output is scale-invariant while the weights are not. Therefore, the traditional *Stochastic Gradient Descent* (SGD) optimization algorithms and its variants are not scale-invariant, which means two neural networks are equivalent to each other, but the behavior of optimization algorithms are different. This

*corresponding author
Email addresses: 11271012@bjtu.edu.cn (Yue Wang), ytliu@bjtu.edu.cn (Yuting Liu), mazm@amt.ac.cn (Zhi-Ming Ma)

mismatch may potentially hurt the optimization process and make the optimization performance sensitive to the weight initialization and hyper-parameter choosing [6]. Recently, some theoretical studies on the Multilayer Perception(MLP) and Convolutional Neural Network (CNN) with ReLU activations show that these networks are invariant to a positively rescaling of their weights which we called Positively Scale-Invariant (PSI) property[6, 7, 8, 9]. Specifically, PSI property means that if the incoming weights of a hidden node(or a feature map for CNN) are multiplied by a positive constant c, and the outgoing weights of this hidden node(or the feature map for CNN) are divided by c, the output for the arbitrary input will keep the same. To solve the above problems, the path representation of MLP and CNN with ReLU activations have been proposed. A path is defined as the multiplication of weights along with a given permutation of hidden nodes from bottom to top layers, which is invariant to the positively rescaling transformation of weights. Previous work [6] shows that the paths can sufficiently represent the output of MLP or CNN and doing optimization directly in the parameter space (i.e., the path space) can help us find the more effective neural network.

Motivated by recent studies on positively scale-invariant parameter space investigated for MLP and CNN with ReLU activations, one natural question is that whether we can seek a parameter space for attention layer that is both scale-invariant and can sufficiently represent the output of attention so that we can employ optimization algorithm in the scale-invariant parameter space? Since the attention layer differs from MLP which simply stacked linear kernels with non-linear activations, the definition of the path in MLP cannot be applied in the attention layer directly.

In this paper, we answer the above questions positively and investigate the invariant property of the attention layer. Specifically, we first prove that the attention layer is invariant to a rescaling of weights and thus show the existence of the Scale-Invariant (SI) property of the attention layer. Second, we define the scale-invariant variable which we called *path* of the attention layer and show that the output of the attention layer can be represented by the path. However, the number of paths is too much to handle and paths are correlated with each other. Therefore, we formally investigate the path representation of the attention layer and define the basis paths which is a subset of all paths and is sufficient to represent all other paths. Leveraging the definition of basis path, we design an *Attention Basis Path Identify(ABPI) Method* to identify the basis paths. Third, by using the basis path, we construct the SI space for the attention layer which is invariant to the rescaling of weights. Finally, we propose to optimize the attention layer directly in its SI space and conduct several experiments on benchmark datasets to show it can help to obtain the more effective neural networks with the attention layer.

2. Related Work

Attention mechanism has been proposed in many previous works in the area such as computer vision and natural language processing[10, 11, 4, 3, 1, 5]. The intuition behind the attention mechanisms is that we need to weight the relevance of any region of the input and take such a weight into account while extract information from it for a specific task. Attention mechanisms provide an automatic way for us to calculate the relevance of the input. The work [10] uses attention structure to help generate the image by dynamically determining the input region observed by the encoder and the output region modified by the decoder. In [11], the authors design local and global attention machines for neural machine translation. Both models yield large gains in the translation results. Also, self-attention has been used successfully in a variety of tasks including reading comprehension[12], abstractive summarization[13] and learning task-independent sentence representations[14, 3, 15]. In the work [1], the authors show the power of the attention mechanisms by defining and using attention layer to achieve state-of-the-art performance in many tasks of natural language processing. They formally design attention layers. Specifically, it calculates the unnormalized attention probability which measures the relevance of any tokens of the input sequence and then uses the probability to combine the information in the input. Many follow-up works[4, 3] also demonstrate the effectiveness of the attention mechanisms and the attention layer. All of these papers optimize the attention structure using traditional optimization algorithms in the weight space and ignore the Scale-Invariant (SI) property.

Recently, many works study the scale-invariant property of the weight space in the optimization for the neural networks with ReLU activations[16, 17, 6, 9, 18]. It is well known that the neural network with ReLU activations has the Positively Scale-Invariant property. However, the output is scale-invariant while

the weights are not. This mismatch will make some optimization algorithms in the original weight space perform badly in some cases. [16] proposes to adjust the optimization algorithm to be scale-invariant in Multi-Layer Perception (MLP) by adding a path-norm regularizer term. [17] does a similar thing in the recurrent neural network(RNN). In the work [6], the authors investigate the Positively Scale-Invariant space for the Multi-Layer Perception(MLP) and Convolutional Neural Networks (CNN). It identifies the PSI space for the MLP and CNN and proposes to optimize the neural networks in its PSI space. Based on the previous work of PSI space, in [9], the authors design a path norm in the PSI space of MLP and CNN and prove its relation with the generalization error. However, to the best of our knowledge, there is no work to study the scale-invariant property of attention layers.

Some other previous works consider the scale-invariant property regarding the input features rather than the weights of the neural network such as CNN and attention structure [19, 20, 21, 22]. For example, [19, 20] mentioned that even though CNN has achieved amazing performance in various computer vision tasks, it has limited ability to tolerate scale variations of the input image. [21] discusses that the attention layer can be better suited to handle the scale-invariance problem, the distance between arguments, in NLP compared to CNN. However, the scale-invariant property investigated in this paper is about the weight of the attention layer, which means that, no matter what the input looks like, we can apply a certain rescaling transformation on weights of the attention layer and the output of attention layer will keep the same. Therefore, this scale-invariant is mostly related to the NN structure and the optimization procedures. It is irrelevant to the inputs. Although the scale-invariant property with respect to the input feature is an interesting direction, it is out of the scope of this paper.

3. Preliminary

In this section, we introduce the background of the transformer architecture, the attention layer, and $\mathcal{G}$-SGD for MLP.

3.1. Introduction to Transformer Architecture

The Transformer architecture is usually developed by stacking the Transformer block[1]. The transformer block consists of a self-attention sub-layer, an encoder-decoder-attention sub-layer(in the decoder of the translation task) and a fully connected sub-layer. An illustration is shown in Figure 1.

The self-attention sub-layer is usually formulated as querying a dictionary with key-value pairs[23, 1]. Q (Query), K (Key), V (Value) are specified as the hidden representations of the previous layer. Using the notation, the output of self-attention sub-layer for given input x can be expressed as follows:

$$Attention(Q, K, V)(x) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{3.1}$$

$$Q = x \cdot W_Q \tag{3.2}$$

$$K = x \cdot W_K \tag{3.3}$$

Note that $QK^\top$ can be viewed as the unnormalized distribution matrix which we called unnormalized attention probability. One example of the unnormalized attention probability part is shown in Fig 3 The relative magnitude of each element on the i-th row of $QK^\top$ measures the contribution of each row in the V matrix with respect to the i-th row output matrix. d_k is the dimension of the keys(K).

The encoder-decoder-attention sub-layer is similar to the self-attention sub-layer. The only difference is that the Q (Query), K (Key) are calculated by using the output of the encoder and V (Value) is calculated by using the output of the previous layer in the encoder-decoder-attention.

After the attention sub-layer, there is a fully connected sub-layer. It can be expressed by

$$FFN(x) = W_2 max(W_1 x + b_1, 0) + b_2 \tag{3.4}$$

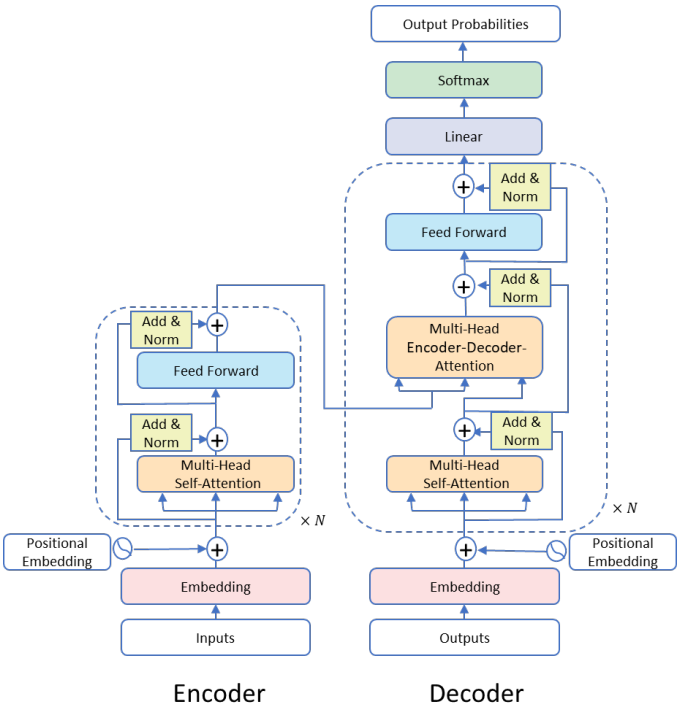

Figure 1: An illustration of Transformer architecture.

There are layer normalization[24] and residual connection[25] after the attention sub-layer and the fully connected sub-layer.

In the following context, we use the following notations in Table 1:

Table 1: Notation

notation	meaning
s	sequence length
d_1	input embedding dimension
d_2	hidden dimension

Thus, the dimension of input data x is $s \times d_1$. The dimensions of weight for both Q and K matrix W_Q, W_K are $d_1 \times d_2$. We use the subscript to denote the index of the element in the matrix. For example x_{ac} denote the element in the row a and column c in the input x which is the element in the sequence a and dimension c of input x.

3.2. Path Space of MLP with ReLU Activations and $\mathcal{G}$-SGD

According to the previous work[16, 6], MLP can be represented by a directed acyclic graph $G(V, E)$. A path of MLP is defined as a series of edges that connect one input node and one output node. Specifically, if we use notation O_i^l to denote the i-th node in the layer l, then we can denote the path as $(O_{i_0}^0 \to O_{i_1}^1 \cdots \to O_{i_L}^L)$. There are edges between the nodes and the value of path is defined as the product of the weights(the value of edges) along the path.

$$v_p = \Pi_{w \in p} w \tag{3.5}$$

The activation status of one path can be calculated as :

$$a_p = \Pi_{o_j \in p} \mathbb{I}(o_j > 0), \tag{3.6}$$

where o_j represent the node which is passed by the path. $\mathbb{I}$ is the indicative function.

The output can be computed by using the value of paths, activation status, and the inputs as follows:

$$N_w(x) = \sum_p v_p \cdot a_p \cdot x_p \tag{3.7}$$

Previous works[6] demonstrate that there exists a minimum subset of path, called basis paths, that is sufficient to represent other paths and independent with each other. We can optimize the MLP by optimizing the value of basis paths. The general framework to do the optimization in the new parameter space can be formulated as 4 steps: Back-Propagation, Inverse-Chain-Rule, Path-Value-Update, and Weight-Allocation. Specifically,

(1. Back-Propagation) We first calculate the gradient of weights $\frac{\partial L}{\partial w}$ by using traditional BP method.

(2. Inverse-Chain-Rule)Then, considering the value of path is the production of weights, the relation between the gradient of weights and the gradient of path is clear and we can easily obtain the gradient of basis path $\frac{\partial L}{\partial v_p}$. Formally, we can solve the following equation to get the gradient of path.

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial v_p} \cdot \frac{\partial v_p}{\partial w} \tag{3.8}$$

(3. Path-Value-Update)After that, we update the value of basis path by following the negative direction of the gradient w.r.t. path. Formally,

$$v_p^{t+1} = v_p^t - \eta_t \frac{\partial L}{\partial v_p} \tag{3.9}$$

(4. Weight-Allocation)Finally, since the value of path is the production of weights, we allocate the value of basis path into the weights for the convenience of the next iteration. Formally, since we already know w^t, v_p^t, v_p^{t+1} from the previous calculation, we can get the updated weights w^{t+1} by solving the following equation:

$$\frac{\Pi_{w \in p} w^{t+1}}{\Pi_{w \in p} w^t} = \frac{v_p^{t+1}}{v_p^t} = 1 - \eta_t \frac{\partial L}{\partial v_p^t} / v_p^t \tag{3.10}$$

The optimization process is showing in Figure 2. We detailed introduce the algorithm in Section 5 including the algorithm procedure, the derivation of the algorithm and a simple case illustration.

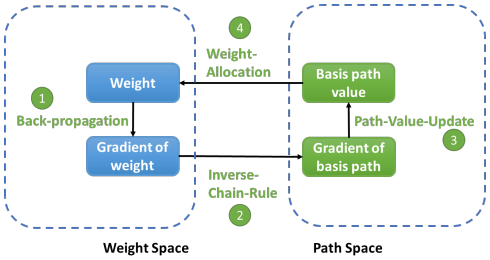

Figure 2: Framework to do optimization in path space

4. Scale-Invariant Space for Attention Layer

In this section, we investigate the invariant property of the attention layer. Firstly, we prove the existence of the Scale-Invariant(SI for short) property of the attention layer. The results show that the output of the attention layer can be calculated by the paths rather than the weight. Secondly, we construct the SI space of the attention layer by finding the basis path which is the maximal independent path that is enough to represent other paths.

Firstly, we introduce the Scale-Invariant transformation for the attention layer.

Definition 1. *Assume that Q, K in attention layer is calculated as $Q = W_Q \cdot x$ and $K = W_K \cdot x$. Suppose $c_1, \cdots, c_{d_2} \neq 0$. We define the rescaling transformation $T_{c_1,\cdots,c_{d_2}}(W_Q, W_K)$ of attention as $T_{c_1,\cdots,c_{d_2}}(W_Q, W_K) = (W'_Q, W'_K)$, where $W_{Q'_{i,\cdot}} = W_{Q_{i,\cdot}} \circ [c_1, \cdots, c_{d_2}]$ and $W_{K'_{i,\cdot}} = W_{k_{i,\cdot}} \circ [1/c_1, \cdots, 1/c_{d_2}]$, where $\circ$ is element wise production which is called Hadamard product. $W_{Q_{i,\cdot}}$ is the i-th row vector in matrix W_Q.*

Proposition 1. *The output of the attention layer is invariant under the rescaling transformation $T_{c_1,\cdots,c_{d_2}}(W_Q, W_K)$.*

4.1. Path Representation of Attention Layer

In this section, we will show that the output of the unnormalized attention probability part $QK^\top$ can be represented by the path that is invariant to the Scale-Invariant transformation. One illustration of the unnormalized attention probability part $QK^\top$ in attention layer is shown in Figure 3.

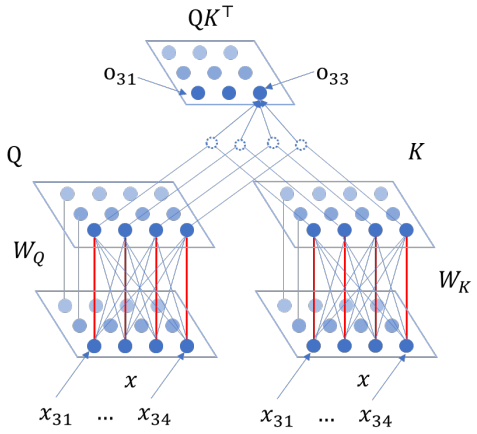

Figure 3: An Example of Unnormalized Attention Probability Part of Attention Layer

Now, we give an example of the representation of the output of the unnormalized attention probability.

$$o_{ab} = \sum_c Q_{ac} K^\top_{cb} \tag{4.1}$$

$$= \sum_c Q_{ac} K_{bc} \tag{4.2}$$

$$= \sum_c \left(\sum_e x_{ae} W_{Q_{ec}} \cdot \sum_{e'} x_{be'} W_{K_{e'c}} \right) \tag{4.3}$$

$$= \sum_{c,e,e'} \underbrace{W_{Q_{ec}} W_{K_{e'c}}}_{paths} x_{ae} x_{be'} \tag{4.4}$$

$$\tag{4.5}$$

where $a, b \in \{1, \cdots, s\}$, $c \in \{1, \cdots, d_2\}$, $e, e' \in \{1, \cdots, d_1\}$.

We can see that the output of the attention layer is invariant under the rescaling transformation $T_{c_1,\cdots,c_{d_2}}(W_Q, W_K)$ for any $c_1, \cdots, c_{d_2}$.

Definition 2. *We define one input node O_{in} as the quadratic term constructed by multiplying two elements in the input data. Define one output node O_{out} as one element in the unnormalized attention probability part. A path in the attention layer is the production of the weights that connect one input node and one output node.*

As an example, in Figure 3, the input data x has shape 3×4 with sequence length 3 and dimension 4. $\{x_{31}x_{31}, x_{31}x_{32}, x_{31}x_{33}, x_{31}x_{34}, \cdots\}$ are some input node and $\{o_{31}, o_{33}, \cdots\}$ are some output node. The production of the left red weight in W_Q and the left red weight in W_K connect the input node $x_{31}x_{31}$ and the output node o_{33} and thus is a path.

Until now, we have shown that the attention layer does have SI property and the output of the attention layer can be expressed by the path rather than the weights of Q and K. Obviously, the number of paths is much more than the number of weights and are correlated with each other.

The next proposition shows the number of paths.

Proposition 2. *Recall the notations in Table 1 in preliminary section. The number of paths is* $\#(paths) = s \times s \times d_1 \times d_1 \times d_2$ *and the number of weights is* $\#(weights) = 2 \times d_1 \times d_2$.

From the above proposition, we know that there are a large number of paths and one natural question is whether these paths are independent with each other. If the answer is no, can we use much less number of path to express these correlated paths? We will discuss this topic in the following.

4.2. Basis Path of Attention Layer

The following examples show that the paths are not independent with each other. Some paths can be calculated by other paths. Moreover, the number of paths is related not only to the network structure but also to the data.

Example 1. *For example, consider the case that the sequence length is 2, the embedding dimension is 2 and the attention hidden size is 1. There are 16 input nodes:*

$$x_{11}x_{11}, x_{11}x_{12}, x_{11}x_{21}, x_{11}x_{22}, \cdots, x_{22}x_{11}, x_{22}x_{12}, x_{22}x_{21}, x_{22}x_{22}.$$

There are 4 weights: 2 in W_Q *and 2 in* W_K*. The size of* W_Q *and* W_K *are both* 2×1*. The number of paths is 4. They are*

$$p_1 = W_{K_{11}} W_{Q_{11}}, \tag{4.6}$$
$$p_2 = W_{K_{11}} W_{Q_{21}}, \tag{4.7}$$
$$p_3 = W_{K_{21}} W_{Q_{11}}, \tag{4.8}$$
$$p_4 = W_{K_{21}} W_{Q_{21}}. \tag{4.9}$$

The size of output is 2×2*, we can calculate the output by using the paths and the inputs. It is easy to verify that* $p_4 = \frac{p_2 * p_3}{p_1}$.

As we can show, paths are correlated with each other, and their number is even related to the data. So, optimizing all the paths directly is incorrect. To leverage the benefit of the SI property of the path representation and optimize the attention layer in its SI space directly, we need to investigate the correlation among the value of paths and find the basis path. The basis path is a set of paths that is independent with each other and is sufficient to represent other paths.

To formally describe the basis path, we introduce the definition of structure vectors and structure matrix which are motivated by Meng et al.[6].

Definition 3. *Structure vector of a path* p *is a vector* $C_p = (c_p^1, \cdots, c_p^m)$*, where* c_p^m *is 1 if the path* p *passes the weight* m *and otherwise 0.* m *is the number of parameters in* W_Q *and* W_K*. Structure matrix is defined as the concatenation of all the structure vectors(column vectors). The size of the structure matrix is* m *times the number of paths.*

Using the definition of structure vectors and structure matrix, we can now define the *basis path*.

Definition 4 (Basis Paths in Attention Layer)**.** *A set of paths* $\mathcal{P}_b$ *in the attention layer are called basis paths if their corresponding structure vectors compose the maximal linear independent group of the column vectors in the structure matrix.*

In Example 1, if we arrange the order of weights in the expression of structure vector as $W_{K_{11}}, W_{K_{12}}, W_{Q_{11}}, W_{Q_{21}}$, the structure vectors of $\{p_1, p_2, p_3, p_4\}$ can be listed as

$$\begin{aligned} C_{p_1} &= (1, 0, 1, 0) \\ C_{p_2} &= (1, 0, 0, 1) \\ C_{p_3} &= (0, 1, 1, 0) \\ C_{p_4} &= (0, 1, 0, 1). \end{aligned}$$

The basis path can be selected as $\{p_1, p_2, p_3\}$ since the structure vectors $C_{p_1}, C_{p_2}, C_{p_3}$ compose the maximal linear independent group of the four structure vectors.

The next proposition shows that the basis paths have some nice properties. Specifically, the basis paths are the smallest subset of all paths that can represent all paths. Through the theorem, we know that we can directly optimize the basis path without losing any information.

Proposition 3. *(1)Any path $p \in \mathcal{P}$ can be represent by the basis paths in $\mathcal{P}_b$ with non-all-zero coefficients $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_z)$ as $p = \Pi_j p_j^{\alpha_j}$, where z is the number of the basis path. (2) Any basis path in $\mathcal{P}_b$ cannot be represented by other other basis paths. In other words, for $\forall i \in \{1, \cdots, z\}$ there are no non-all-zero coefficients $\alpha = (\alpha_1, \cdots, \alpha_{i-1}, \alpha_{i+1}, \cdots, \alpha_z)$ that $p_i = \Pi_{j \neq i} p_j^{\alpha_j}$.*

4.3. Attention Basis Path Identify(ABPI) Method

In this section, we design an efficient method called *Attention Basis Path Identify(ABPI) Method*(Algorithm 1) to identify the basis paths.

In general, the method selects the basis path according to the following principle.

(1) For the attention layer, we first select the skeleton weights. For each column in W_Q and W_K weight matrix, we choose one element as the skeleton weights. All other weights are non-skeleton weights.

(2) The path containing at most one non-skeleton weights is the basis path.

As illustrated in Figure 3, the red line is the skeleton weights. The next theorem shows the correctness of the proposed ABPI algorithm.

Algorithm 1: Attention Basis Path Identification Method

Input: initial W_Q, W_K, the dimension of weight matrix $d_1 \times d_2$, the basis path buffer $\mathcal{P}_b$

1 **for** $t = 1, \ldots, d_2$ **do**
2 select the element in the t mod d_1 row and t columns as the skeleton weights. $W_{Q_{t \bmod d_1, t}}, W_{K_{t \bmod d_1, t}}$;
3 add the path $W_{Q_{t \bmod d_1, t}} \cdot W_{K_{t \bmod d_1, t}}$ into $\mathcal{P}_b$;
4 **for** $s = 1, \ldots, d_1$ **do**
5 **if** $W_{Q_{s,t}}$ *is not skeleton weight* **then**
6 add the path $W_{Q_{s,t}} \cdot W_{K_{t \bmod d_1, t}}$ into $\mathcal{P}_b$
7 **end**
8 **if** $W_{K_{s,t}}$ *is not skeleton weight* **then**
9 add the path $W_{Q_{t \bmod d_1, t}} \cdot W_{K_{s,t}}$ into $\mathcal{P}_b$
10 **end**
11 **end**
12 **end**

Output: $\mathcal{P}_b$

Theorem 1. *The set of path selected by the* ABPI Method *are basis paths.*

Proof of Theorem 1. According to the ABPI Method, there are some of the basis path that composed of all skeleton weights(Line 3 in Algorithm 1) and others include one non-skeleton weights (Line 6 and 9 in

Algorithm 1). So the counting vector of the paths in $\mathcal{P}_b$ can be listed as column vectors and expressed by the form

$$S = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}.$$

For the submatrix A, the rows of A represent the skeleton weight and the columns of A represent the paths in $\mathcal{P}_b$ that composed of all skeleton weights. For the submatrix C, the rows of C represent the non-skeleton weight and the columns of C represent the path in $\mathcal{P}_b$ that includes one non-skeleton weight.

Through the *ABPI Method*, we know that every non-skeleton weight can only appear on one basis path. So, C is an identity matrix $\mathbb{I}$. The rank of identity matrix $\mathbb{I}$ is the number of the non-skeleton weights. Therefore, through the row transformation of the matrix, B can be transformed into a zero matrix.

Next, we will show that the rank of matrix A is the number of its columns. According to the *ABPI Method*, all skeleton weights will only appear in one of the paths that composed of all skeleton weights.

Until now, we have proved the independent of the columns of the structure vectors for the paths selected by the *ABPI Method*. The remaining issue is to prove that other paths can be represented by the path in $\mathcal{P}_b$. For arbitrary path p that composed of two non-skeleton weights, denote as $p = w_{ns}^1 w_{ns}^2$. We choose three paths in $\mathcal{P}_b$ denote as p_1, p_2, p_3. p_1 and p_2 contain one non-skeleton weight w_{ns}^1, w_{ns}^2 of p respectively. Specifically, denote $p_1 = w_{ns}^1 w_s^1, p_2 = w_{ns}^2 w_s^2$. p_3 is the all-skeleton weight path that contain the skeleton weight in p_1 and p_2. Specifically, denote $p_3 = w_s^1 w_s^2$. So path $p = w_{ns}^1 w_{ns}^2 = \frac{p_1 p_2}{p_3}$. Since the path p is arbitrarily selected, we prove that other paths can be represent by the path in $\mathcal{P}_b$.

Thus, we have shown the independent and maximal property of the path in $\mathcal{P}_b$.

□

Now, we can identify the basis path efficiently by using the ABPI method. Using the basis path, we can construct the SI space for the attention layer.

Definition 5. *If $\{ p_1 \cdots, p_z \}$ are basis paths, the SI space of the attention layer is the vector space*

$$V := \{v = (p_1, \cdot, p_z) | v \in (\mathbb{R} \backslash 0)^z\} \tag{4.10}$$

The next theorem shows the dimension of the SI space.

Theorem 2. *The dimension of the SI space is m-H, where m is the number of parameters in W_Q and W_K and H is the number of attention hidden size.*

Proof. By definition of the new space is equal to the number of the basis path. According to the proof of the Theorem 1, we can see that the number of the basis path is the number of the columns of S. The number of columns of A is the hidden size of the attention layer H. The number of the columns of C is the number of non-skeleton weights. Note there are $2H$ skeleton weights in weight matrix Q and K. Therefor, the number of non-skeleton weights is $m - 2H$. So the number of basis paths is $m - H$ in total. □

5. Experiments

In this section, we conduct experiments to verify our theoretical analysis and show the effectiveness of the proposed SI space. First, we introduce the optimization algorithms in SI space and present the derivation method in detail. Second, we show our experiment results together with the experiment settings.

5.1. Optimization algorithms in SI Space – SI-SGD

We leverage the optimization algorithms in SI space proposed in [6], which update the model by doing Stochastic Gradient Decent according to the gradient value of the basis path in the SI space. After we identify the basis path using the ABPI method, we can leverage the algorithm to optimize the attention layer in SI space and we call is SI-SGD.

$$v_i^{t+1} = v_i^t - \eta_t \frac{\partial L(v)}{\partial v_i} \tag{5.1}$$

As showing in Fig 2, we leverage several steps such as Inverse-Chain-Rule and Weight-Allocation proposed in [6] to do the optimization in the SI-space. The end-to-end update rule is shown in the Algorithm 2.

Please note that SI-SGD only needs a little extra computation cost compared with SGD in weight space. In the realistic experiment, the extra training time of optimization algorithms in SI space is less than 10% compare with the traditional optimization algorithms in weight space.

We denote the skeleton weights in weight matrix W_Q as w_{Q_j} and the non-skeleton weights as w_{Q_i}. Denote the skeleton weights in weight matrix W_K as w_{K_j} and the non-skeleton weights as w_{K_i}. G_{W_Q} and G_{W_K} are the gradient of weights W_Q and W_K respectively. η_t is the learning rate at time-step t. $\{w_{Q_i:K_j}\}$ is the set of of non-skeleton weight in W_Q that is connected with the skeleton weight w_{K_j}. Also, we denote the basis path that composed of all skeleton weights as p_s(Line 3 in Algorithm 1). We denote the rest basis path in Line 6 and 9 of Algorithm 1 as $p_{Q_{ns}}$ and $p_{K_{ns}}$ respectively (Subscript Q or K is according to the location of the non-skeleton weight of this basis).

Algorithm 2: SI-SGD for attention layer

Input: weights of attention layer W_Q, W_K, data X, Y, loss function $loss$, learning rate η

1 **for** $i = 1 \cdots d_2$ **do**

2 For W_Q, select the element in the i mod d_1 row and t columns ($W_{Q_{i \bmod d_1, i}}$) as the skeleton weights.

3 For W_K, select the element in the i mod d_1 row and t columns ($W_{K_{i \bmod d_1, i}}$) as the skeleton weights.

4 Initialize all the skeleton weight as 1.

5 **end**

6 t = 0;

7 **repeat**

8 t = t +1 ;

9 Sample a batch of data (x, y), compute the loss function $l = loss(F(x, W_Q, W_K), y)$;

10 Using Backpropagation to calculate the gradient w.r.t the weight: $G_{W_Q}, G_{W_K} = BP(l, W_Q, W_k)$;

11 ## **Calculate the gradient w.r.t. basis path by using Inverse-Chain-Rule :** $G_{p_s} = G_{w^t_{Q_j}} - \sum_{w_{K_i}:Q_j} \frac{G_{w^t_{K_i}} \cdot w^t_{K_i}}{w^t_{Q_j}}$;

12 $G_{p_{Q_{ns}}} = G_{w_{Q_i}}$;

13 $G_{p_{K_{ns}}} = G_{w_{K_i}} / w_{Q_j:K_i}$;

14 ## **Calculate the Update Ratio :**

15 $R(p_s) = 1 - \eta_t G_{p_s} / w_{Q_j}$

16 ## **Weight-Allocation and update weight matrices as :**

17 $w^{t+1}_{Q_j} = w^t_{Q_j} \cdot R(p_s)$

18 $w^{t+1}_{Q_i} = w^t_{Q_i} - \eta_t G_{w_{Q_i}}$

19 $w^{t+1}_{K_j} = w^t_{K_j}$

20 $w^{t+1}_{K_i} = \frac{w^t_{K_i} - \eta_t G_{p_{K_{ns}}}}{R(p_s)}$

21 **until** *stopping criterion is satisfied*;

Output: Weights of Attention Layers W_Q, W_K

Here, we review the derivation of the Inverse-Chain-Rule and Weight-Allocation which are previously designed in the work [6]. Before we fall into the detailed math, we first illustrate the algorithms in a simple case.

Consider a simple attention layer with $d_1 = 2$ and $d_2 = 1$. Then the shapes of weight matrix W_Q and W_K are both 2×1. See the Figure 4 for an example.

According to the Line 1 - 4 in the algorithms 2, we select the $W_{Q_{11}}, W_{K_{11}}$ as the skeleton weights.

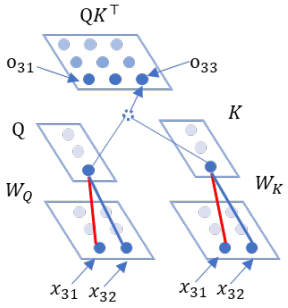

Figure 4: An example of attention layer

According to the Theorem 2, there are $4-1=3$ basis paths :

$$p_{b_1} = p_s = W_{Q_{11}} \times W_{K_{11}} \tag{5.2}$$
$$p_{b_2} = p_{K_{ns}} = W_{Q_{11}} \times W_{K_{12}} \tag{5.3}$$
$$p_{b_3} = p_{Q_{ns}} = W_{Q_{12}} \times W_{K_{11}} \tag{5.4}$$
$$\tag{5.5}$$

Assume the gradients of each weight are $G_{Q_{11}}, G_{Q_{12}}, G_{K_{11}}, G_{Q_{12}}$ Execute the Algorithm 2 into line 11, we can get the gradient for the basis paths.

$$G_{p_s} = G_{Q_{11}} - \frac{G_{K_{12}} \times W_{K_{12}}}{W_{Q_{11}}} \tag{5.6}$$
$$G_{p_{K_{ns}}} = G_{K_{12}} / W_{Q_{11}} \tag{5.7}$$
$$G_{p_{Q_{ns}}} = G_{Q_{12}} \tag{5.8}$$

Execute the Algorithm 2 into line 13, we can get the update ratio for the skeleton basis paths.

$$R(p_s) = 1 - \eta_t G_{p_s} / W_{Q_{11}} \tag{5.9}$$

Execute the Algorithm 2 from line 15 to line 18, we can get the update rule for all weights relatively.

$$W_{Q_{11}} = W_{Q_{11}} \times R(p_s) \tag{5.10}$$
$$W_{Q_{12}} = W_{Q_{12}} - \eta_t G_{Q_{12}} \tag{5.11}$$
$$W_{K_{11}} = W_{K_{11}} \tag{5.12}$$
$$W_{K_{12}} = \frac{W_{K_{12}} - \eta_t G_{k_{ns}}}{R(p_s)} \tag{5.13}$$

Until now, we have finished one round of optimization. And the next step is to do forward and backward pass to get the new gradient of weights and repeat the above procedure.

In the following, we formally show the derivation of the Inverse-Chain-Rule and the Weight-Allocation method.

Inverse-Chain-Rule calculates the gradient of the basis path by using the gradients of wights.

$$\left(\frac{\partial L}{\partial w_1}, \cdots, \frac{\partial L}{\partial w_m}\right) = \left(\frac{\partial L}{\partial v_{p_1}} \cdots \frac{\partial L}{\partial v_{p_{m-H}}}\right) \cdot \begin{bmatrix} \frac{\partial v_{p_1}}{\partial w_1} & \cdots & \frac{\partial v_{p_1}}{\partial w_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial v_{p_{m-H}}}{\partial w_1} & \cdots & \frac{\partial v_{p_{m-H}}}{\partial w_m} \end{bmatrix} \tag{5.14}$$

Solving this equation, the gradients w.r.t basis paths can be derived as:

$$G_{p_s} = G_{w^t_{Q_j}} - \sum_{w_{K_i}:Q_j} \frac{G_{w^t_{K_i}} \cdot w^t_{K_i}}{w^t_{Q_j}} \tag{5.15}$$

$$G_{p_{Q_{ns}}} = G_{w_{Q_i}} \tag{5.16}$$

$$G_{p_{K_{ns}}} = G_{w_{K_i}} / w_{Q_j:K_i} \tag{5.17}$$

Weight-Allocation project the update on basis paths back to the updates of weight.

$$\frac{p^t_s - \eta \frac{\partial L}{\partial p^t_s}}{p^t_s} = \left(\frac{p^{t+1}_s}{p^t_s} \triangleq R(p_s) \right) = \frac{w^{t+1}_{Q_j} \cdot w^{t+1}_{K_j}}{w^t_{Q_j} \cdot w^t_{K_j}} \tag{5.18}$$

$$\frac{p^t_{K_{ns}} - \eta \frac{\partial L}{\partial p^t_{K_{ns}}}}{p^t_{K_{ns}}} = \left(\frac{p^{t+1}_{K_{ns}}}{p^t_{K_{ns}}} \triangleq R(p_{K_{ns}}) \right) = \frac{w^{t+1}_{Q_i} \cdot w^{t+1}_{K_j}}{w^t_{Q_i} \cdot w^t_{K_j}} \tag{5.19}$$

$$\frac{p^t_{Q_{ns}} - \eta \frac{\partial L}{\partial p^t_{Q_{ns}}}}{p^t_{Q_{ns}}} = \left(\frac{p^{t+1}_{Q_{ns}}}{p^t_{Q_{ns}}} \triangleq R(p_{Q_{ns}}) \right) = \frac{w^{t+1}_{Q_j} \cdot w^{t+1}_{K_i}}{w^t_{Q_j} \cdot w^t_{K_i}} \tag{5.20}$$

Solving this equation, the update rule for weight can be derived as:

$$w^{t+1}_{Q_j} = w^t_{Q_j} \cdot R(p_s) \tag{5.21}$$

$$w^{t+1}_{Q_i} = w^t_{Q_i} - \eta_t G_{w_{Q_i}} \tag{5.22}$$

$$w^{t+1}_{K_j} = w^t_{K_j} \tag{5.23}$$

$$w^{t+1}_{K_i} = \frac{w^t_{K_i} - \eta_t G_{p_{K_{ns}}}}{R(p_s)} \tag{5.24}$$

5.2. Language Modeling

In this section, we optimize the Transformer model for the language modeling tasks. Language modeling task[26, 27] aims to construct a probability model that approximates the joint probability of sequences of words in a language. Specifically, given a corpus of tokens(a sentence) $x = (x_1, x_2, \cdots, x_T)$, we need to estimate the joint probability $P(x) = \Pi^T_{t=1} p(x_t | x_{<t})$. Language model has many successful applications such as natural language generation[28, 29] and unsupervised pretraining [30, 3]. In this experiments, we use the Transformer model with attention layers to approximate the joint probability.

We consider three datasets: word-level Penn Treebank dataset(PTB)[31], Wikitext-2 datasets[32], character-level Penn Treebank dataset(PTBc). The first two datasets are word level. The PTB dataset consists of 929K training words, 73K validation words, and 82K test words. It has 10k words in its vocabulary. Wikitext-2 is roughly twice the size of PTB dataset, with 2 million training words, 218k validation words, 245k test words and a vocab size of 33k. The last dataset is character level. The PTBc dataset consists of 5017k training character, 393k validation character, and 442k test character. It has 50 alphabets in its vocabulary.

For all tasks, we train an n($n \in \{1, 3, 6\}$) layer Transformer model with attention embedding size 256 and MLP layer embedding size 512. The attention heads are 4. The sequence length is 1024 tokens. we apply the dropout trick with probability 0.1. We train each model 250 epochs. We decay the learning rate at 100 and 200 epochs by factor 0.1. We construct SI space for the model and optimize the model in its SI space directly. We choose SGD in weight space as a baseline. We search the learning rate from { 5 4, 3, 2, 1, 0.75, 0.5, 0.25, 0.1 } for both baseline and our method.

We use the Perplexity(PPL) as the evaluation metric for word-level datasets(PTB and Wikitext-2) and use bits-per-character(BPC), which is log_2 of the Perplexity, of the character-level datasets(PTBc).

We run every experiment for 4 times with different random seeds. The mean values and the standard deviation values are shown in Table 2 before and after symbol ±. The lower number means better performance. As we can see, our method performs better compared with the baseline in about all tasks in all settings which demonstrate that optimize the attention layer in its SI space is better than in weight space.

Table 2: Performance of Language Modeling Tasks

		SGD	SI-SGD
Wikitext-2	1	123.23±0.36	**121.85±0.46**
	3	112.82±1.18	**109.65±2.07**
	6	103.33±1.11	**100.28±0.80**
PTB	1	138.54±0.77	**137.20±0.73**
	3	123.22±1.22	**121.30±0.71**
	6	123.45±1.74	**121.13±0.48**
PTBc	1	2.22±0.002	**2.17±0.02**
	3	1.88±0.01	**1.87±0.01**
	6	1.80±0.01	**1.75±0.01**

Considering the variance of the experiments, we also implement the 2-sample T-test[33, 34] to verify the significance of our results by giving a quantitative measure[1]. The alternative hypothesis is that our result is significantly better than the baseline method(PPL/PBC is significantly less than the baseline method). We show P-values in the Table 3. As shown in the table, almost all the P-values are less than significant level 0.05 which means we can accept the alternative hypothesis and our improvement are statistically significant.

Table 3: Performance of Language Modeling Tasks

		P-Values
Wikitext-2	1	0.003
	3	0.036
	6	0.005
PTB	1	0.035
	3	0.034
	6	0.050
PTBc	1	0.009
	3	0.226
	6	0.005

5.3. Machine Translation

In this section, we demonstrate the effectiveness of our method on four datasets with different scale: German to English, Spanish to English, Hebrew to English in IWSLT2014 and English to German in WMT2014.

Machine Translation task aims to seek a target language translation y' given source sentence x. Specifically, the data is sentences pairs (x, y) from two different languages and we need to estimate a probabilistic model $G(y|s)$ so that we can sample the translation result $y' \sim G(\cdot|x)$ from the model. In this experiment, we use Transformer models with attention layers to model the probabilistic model.

[1]T-test is most commonly used to determine if the means of two sets of data are significantly different(greater or less) from each other. We set the null hypothesis as the means are not significantly different from each other and then calculate the P-value. The P-value is the probability of obtaining the test result assuming that the null hypothesis. If the P-value is less than a significant level(0.05 is commonly used), we can reject the null hypothesis and accept the alternative hypothesis.

German to English. There are 160k and 7k sentence pairs in the training and validation datasets. We follow the same setup in [35] to combine *tst2010, tst2011, tst2012, dev2010 and dev 2012* to be our test data. As a vocabulary, we use a joint source and target vocabulary with 10k sub-word types based on byte-pair-encoding (BPE)[36].
Spanish to English There are 181k parallel sentence pairs in the training data. *tst2013* is used as the validation data and *tst2014* is used as the test data. The vocabulary is the same as the previous one.
Hebrew to English There are 181k parallel sentence pairs in the training data. *tst2013* is used as the validation data and *tst2014* is used as the test data. The vocabulary is the same as the previous one.
English to German The training data contains 4.5M sentences pairs. We combine *newstest2012* and *newstest2013* as the validation data and combine *newstest2014* as test data. We use 40k sub-word types based on BPE.

For the former three IWSLT2014 datasets, we use the Transformer architecture with 6 layers for both encoder and decoder. The hidden size of attention and MLP layer is 512 and 1024 respectively. The head for the multi-head attention is 4. For the last WMT2014 datasets, we use the default $transformer\_big$ configuration with 6 layers for both encoder and decoder. the hidden size of attention and MLP layer is 1024 and 4096 respectively. The evaluation metric is BLEU score [37].

We choose commonly used Adam optimizer in these tasks as our baseline[38, 1]. To accelerate training, we heuristically modify our optimization algorithm in SI space into the Adam-like algorithms(Denoted as SI-ADAM). Specifically, we accumulate the gradient of basis path in the SI space and apply Adam formulations in the SI space when updating the basis path value.

We apply our SI-ADAM method on the attention layer in encoder, decoder and both encoder-decoder. The result is shown in Table 4.

Table 4: BLEU score of machine translation tasks

	IWSLT			WMT
	de → en	es → en	he → en	en → de
Adam	34.79	41.58	33.64	28.40
SI-ADAM(decoder-attention)	**35.33±0.12**	**41.84±0.12**	**34.21±0.37**	**29.05±0.11**
SI-ADAM(encoder-attention)	34.28	40.96	33.30	26.55
SI-ADAM(all-attention)	34.73	41.05	33.41	26.86

The baseline we list in the Table 4(results of Adam method in the first line) is from the previous paper [39, 1]. We can observe in the Table 4, optimize the decoder of the Transformer model in the SI space can achieve better performance compared to traditional optimization algorithms in all datasets. Specifically, in large size WMT datasets, we improve the BLEU score by 0.65 points on average. In middle size IWSLT datasets, we improve the BLEU score by 0.57, 0.26, 0.54 points on average respectively. Note that optimizing the encoder of the Transformer model will not lead to a good result. This may because the SI-ADAM algorithm is just heuristic, a more appropriate ADAM-like algorithm in SI space is needed for better performance. Since this work is focused on the SI space for the attention layer rather than designing the optimization algorithms in SI space, we leave it for future work.

To verify the significance of our improvements, we repeat the SI-ADAM algorithms for 4 times to calculate the variance of the experiment results. The standard deviation is also shown in the table after the ± symbol.

Besides, we apply the one-sample T-test[33, 34] to give a quantitative measure of how our improvement is significant. The alternative hypothesis is our improvement is significantly larger than 0. P-Values is shown in the table 5. We can see that all the P-Values are less than 0.05 which means we can accept the alternative hypothesis. In other words, our results are statistically significant.

6. Conclusion

In this paper, we investigate the scale-invariant property of the attention layer and demonstrate the advantage of optimizing the attention layer in the new space. First, we prove that there exists the Scale

Table 5: P-value of machine translation tasks

	IWSLT			WMT
	de → en	es → en	he → en	en → de
P-Value	0.003	0.018	0.038	0.0009

Invariant (SI for short) property in the attention layer. Then, we introduce the path representation for the attention layer and formally investigate the path representation. After that, we define the basis path and propose an efficient *Attention Basis Path Identification Method* to identify the basis path. Using the definition of basis path, we construct the SI space for the attention layer. Finally, we propose to optimize the attention layer in the SI space by using the SI-SGD algorithms and conduct several experiments on benchmark datasets. The results demonstrate the correctness of the proposed SI space and the effectiveness of optimizing the attention layer in its SI space. In the feature, we plan to investigate other properties of the attention layer by using the path view such as the interpretability of the attention layer.

Acknowledgment

This work is supported by the Fundamental Research Funds for the Central Universities(2018JBM320). Support by the German Research Foundation (DFG) through the CRC 1283 "Taming uncertainty and profiting from randomness and low regularity in analysis, stochastics and their applications" is acknowledged.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017.

[2] R. Al-Rfou, D. Choe, N. Constant, M. Guo, L. Jones, Character-level language modeling with deeper self-attention, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 3159–3166.

[3] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

[4] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, R. Salakhutdinov, Transformer-XL: Attentive language models beyond a fixed-length context, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 2978–2988. doi:10.18653/v1/P19-1285.

[5] Y. Lu, Z. Li, D. He, Z. Sun, B. Dong, T. Qin, L. Wang, T.-Y. Liu, Understanding and improving transformer from a multi-particle dynamic system point of view, arXiv preprint arXiv:1906.02762.

[6] Q. Meng, S. Zheng, H. Zhang, W. Chen, Z.-M. Ma, T.-Y. Liu, G-SGD: Optimizing reLU neural networks in its positively scale-invariant space, in: International Conference on Learning Representations, 2019.

[7] B. Neyshabur, R. R. Salakhutdinov, N. Srebro, Path-sgd: Path-normalized optimization in deep neural networks, in: Advances in Neural Information Processing Systems, 2015, pp. 2422–2430.

[8] B. Neyshabur, Y. Wu, R. R. Salakhutdinov, N. Srebro, Path-normalized optimization of recurrent neural networks with relu activations, in: Advances in Neural Information Processing Systems, 2016, pp. 3477–3485.

[9] S. Zheng, Q. Meng, H. Zhang, W. Chen, N. Yu, T.-Y. Liu, Capacity control of relu neural networks by basis-path norm, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 5925–5932.

[10] K. Gregor, I. Danihelka, A. Graves, D. Rezende, D. Wierstra, Draw: A recurrent neural network for image generation, in: International Conference on Machine Learning, 2015, pp. 1462–1471.

[11] T. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1412–1421. doi:10.18653/v1/D15-1166.

[12] A. W. Yu, D. Dohan, Q. Le, T. Luong, R. Zhao, K. Chen, Fast and accurate reading comprehension by combining self-attention and convolution, in: International Conference on Learning Representations, 2018.

[13] R. Paulus, C. Xiong, R. Socher, A deep reinforced model for abstractive summarization, in: International Conference on Learning Representations, 2018.

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

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners.

[16] B. Neyshabur, R. Salakhutdinov, N. Srebro, Path-sgd: Path-normalized optimization in deep neural networks, in: Twenty-ninth Conference on Neural Information Processing Systems, 2015.

[17] B. Neyshabur, Y. Wu, R. Salakhutdinov, N. Srebro, Path-normalized optimization of recurrent neural networks with relu activations, in: Thirtieth Conference on Neural Information Processing Systems, 2016.

[18] V. Badrinarayanan, B. Mishra, R. Cipolla, Symmetry-invariant optimization in deep networks, ArXiv abs/1511.01754.
[19] G. Nam, H. Choi, J. Cho, I.-J. Kim, Psi-cnn: A pyramid-based scale-invariant cnn architecture for face recognition robust to various image resolutions, Applied Sciences 8 (9) (2018) 1561.
[20] Y. Xu, T. Xiao, J. Zhang, K. Yang, Z. Zhang, Scale-invariant convolutional neural networks, ArXiv abs/1411.6369.
[21] G. Tang, M. Müller, A. Rios, R. Sennrich, Why self-attention? a targeted evaluation of neural machine translation architectures, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 4263–4272. doi:10.18653/v1/D18-1458.
[22] R. Li, K. Xian, C. Shen, Z. Cao, H. Lu, L. Hang, Deep attention-based classification network for robust depth prediction, in: Asian Conference on Computer Vision, Springer, 2018, pp. 663–678.
[23] A. H. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, J. Weston, Key-value memory networks for directly reading documents, in: EMNLP, 2016.
[24] J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, in: Advances in Neural Information Processing Systems 2016 Deep Learning Symposium, 2016.
[25] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: European conference on computer vision, Springer, 2016, pp. 630–645.
[26] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, A neural probabilistic language model, Journal of machine learning research 3 (Feb) (2003) 1137–1155.
[27] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, S. Khudanpur, Recurrent neural network based language model, in: Eleventh annual conference of the international speech communication association, 2010.
[28] A. Radford, R. Jozefowicz, I. Sutskever, Learning to generate reviews and discovering sentiment (2018).
[29] S. Merity, B. McCann, R. Socher, Revisiting activation regularization for language rnns, arXiv preprint arXiv:1708.01009.
[30] A. M. Dai, Q. V. Le, Semi-supervised sequence learning, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Twenty-ninth Conference on Neural Information Processing Systems, Curran Associates, Inc., 2015, pp. 3079–3087.
[31] M. P. Marcus, M. A. Marcinkiewicz, B. Santorini, Building a large annotated corpus of english: The penn treebank, Computational linguistics 19 (2) (1993) 313–330.
[32] S. Merity, C. Xiong, J. Bradbury, R. Socher, Pointer sentinel mixture models, in: International Conference on Learning Representations, 2017.
[33] E. L. Lehmann, J. P. Romano, Testing statistical hypotheses, Springer Science & Business Media, 2006.
[34] W. Haynes, Student's t-Test, Springer New York, New York, NY, 2013, pp. 2023–2025.
[35] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y. Dauphin, Convolutional sequence to sequence learning, in: International Conference on Machine Learning, 2017.
[36] R. Sennrich, B. Haddow, A. 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), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1715–1725. doi:10.18653/v1/P16-1162.
[37] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics, 2002, pp. 311–318.
[38] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2015.
[39] F. Gao, J. Zhu, L. Wu, Y. Xia, T. Qin, X. Cheng, W. Zhou, T.-Y. Liu, Soft contextual data augmentation for neural machine translation, in: Proceedings of the 57th Conference of the Association for Computational Linguistics, 2019, pp. 5539–5544.

7. Appendix

7.1. Proof of Proposition 3

Proof. We denote the structure vectors of basis paths as $C_{p_1}, \cdots C_{p_z}$. Note that these vectors compose the maximal linear independent group of the structure matrix according to the definition. Now, we need to verify the two conclusions.

(1) For arbitrary path $p \in \mathcal{P}$ it has the corresponding counting vecctor C_p. C_p can be represented by the maximal linear independent group vector as $C_p = \sum_{i=1}^{z} \alpha_i C_{p_i}$. So the path can be represented by the basis path as

$$p = \Pi_{j=1}^{m} w_j^{C_p,j} \tag{7.1}$$

$$= \Pi_{j=1}^{m} w_j^{\sum_{i=1}^{z} \alpha_i C_{p_{i,j}}} \tag{7.2}$$

$$= \Pi_{i=1}^{z} \left(\Pi_{j=1}^{m} w_j^{C_{p_{i,j}}} \right)^{\alpha_i} \tag{7.3}$$

$$= \Pi_{i=1}^{z} p_i^{\alpha_i} \tag{7.4}$$

(2) We proof by contradiction. Without loss of generality, we consider the path p_1 and its counting vector C_{p_1}. Assume there exist non-all-zero coefficients $\{\alpha_2 \cdots \alpha_z\}$ that can represent p_1 as $p_1 = \Pi_{i=2}^{z} p_i^{\alpha_i}$. We can get that

$$left = p_1 = \Pi_{j=1}^{m} w_j^{C_{p_1,j}} \tag{7.5}$$

$$right = \Pi_{i=2}^{z} p_i^{\alpha_i} = \Pi_{i=2}^{z} (\Pi_{j=1}^{m} w_j^{C_{p_i,j}})_i^{\alpha} = \Pi_{j=1}^{m} w_j^{\sum_{i=2}^{z} \alpha_i C_{p_i,j}} \tag{7.6}$$

Therefore, we can get $C_{p_1,j} = \sum_{i=2}^{z} \alpha_i C_{p_i,j}$. it is opposite to the assumption that the set of structure vectors compose the maximal linear independent group of the structure matrix.

□