Research Article

Translution: Unifying Self-Attention and Convolution for Adaptive and Relative Modelling

Hehe Fan[1], Yi Yang[1], Fei Wu[1]

1. College of Computer Science and Technology, Zhejiang University, China

When modeling a given type of data, we consider it to involve two key aspects: 1) identifying relevant elements (*e.g.*, image pixels or textual words) to a central element, as in a convolutional receptive field, or to a query element, as in self-attention, and (2) encoding these tokens effectively. Self-attention can adaptively identify these elements but relies on absolute positional embedding for structural representation learning. In contrast, convolution encodes elements in a relative manner, yet their fixed kernel size limits their ability to adaptively select the relevant elements. In this paper, we introduce Translution, an operation that unifies the adaptive identification capability of self-attention and the relative encoding advantage of convolution. However, this integration leads to a substantial increase in the number of parameters, exceeding most currently available computational resources. Therefore, we propose a lightweight variant of Translution, named α-Translution. Experiments on computer vision and natural language processing tasks show that Translution (including α-Translution) achieves superior accuracy compared to self-attention, demonstrating its potential to build the next generation of deep neural networks. The code is available at https://github.com/hehefan/Translution.

Corresponding author: Hehe Fan, hehefan@zju.edu.cn

1. Introduction

Recent evidence suggests that directly scaling up deep neural networks, particularly Transformers[1][2][3][4], with additional data and parameters is encountering diminishing returns. Leading Artificial Intelligence (AI) labs have similarly noted slower-than-anticipated improvements in next-generation

 doi.org/10.32388/Q6KCoC.2

models, despite extensive training efforts. Given the saturation of available data and limitations imposed by current scaling laws, it is crucial now to reflect on past successes and pursue the design of innovative neural networks to sustain future progress in deep learning.

When employing deep neural networks to model a specific type of data, the process can be decomposed into two key aspects: 1) identifying relevant data elements and 2) encoding these elements into effective representations. When using convolutional neural networks[5][6][7][8][9] to process images, the basic element is pixel. When using Transformers, the element is word for natural language processing and patch for visual tasks.

1.1. Identification of Relevant Elements

In convolution, as shown in Figure 1 (a), the relevant element identification step is handled by convolutional filters (kernels) with a fixed local receptive field. This fixed kernel defines a neighborhood that is considered relevant to the center. For visual data like images, such local focus is often effective because spatially adjacent pixels tend to be related (*e.g.*, forming parts of the same object). However, the rigid nature of a fixed-size kernel makes convolution inevitably cover irrelevant pixels, especially near object boundaries or in background areas that fall inside the window.

In contrast, as shown in Figure 1 (b), self-attention[1] can adaptively identify relevant regions. Instead of being limited to a predetermined locality, it allows the model to dynamically attend to relevant regions. This means that self-attention can focus on important features regardless of their physical distance. This capability provides greater flexibility compared to the convolution's fixed receptive field.

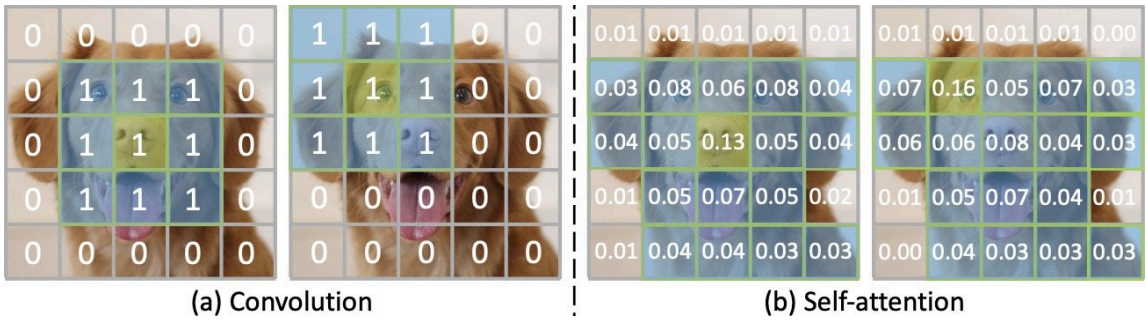

Figure 1. Difference between convolution and self-attention in identifying relevant elements (blue patches) for the kernel center or query element (yellow patch). Here, convolution is assumed to operate on image patches. 1) Convolution utilizes a fixed kernel size to define a neighborhood of elements considered relevant, inevitably including some irrelevant regions, particularly near object boundaries or within background areas inside the window. The fixed receptive field in convolution can be interpreted as a special case of attention, where the attention score is set to 1 within the receptive field and 0 outside it. 2) Self-attention adaptively identifies relevant elements by assigning greater attention scores to areas with higher relevance, thereby mitigating the inclusion of noisy or irrelevant information.

1.2. Encoding of Relevant Elements

When it comes to encoding the structure from these relevant elements, convolution and self-attention employ different strategies. As shown in Figure 2 (a), a convolutional kernel learns distinct parameters $\{\mathbf{W}_{\delta_x,\delta_y}\}$ for each relative direction and distance within its receptive field. In other words, the filter has separate parameters $\mathbf{W}_{\delta_x,\delta_y}$ for each offset δ_x, δ_y from the center. This design enables convolution to encode local structure relatively — capturing orientation and distance relationships.

In contrast, as shown in Figure 2 (b), self-attention uses three shared sets of parameters $\mathbf{W}^q$, $\mathbf{W}^k$ and $\mathbf{W}^v$ to process inputs for all positions. Consequently, the query, key and value of self-attention do not encode whether one patch is to the left or right of another. To introduce positional information, Transformer incorporates absolute positional embeddings into the input features at the outset. Although these embeddings enable Transformer to infer order or spatial relationships, they introduce noise into each token's representation. The absolute position information becomes part of the input features. Consequently, when the same object moves to a different location, Transformer may struggle to recognize it.

doi.org/10.32388/Q6KCoC.2

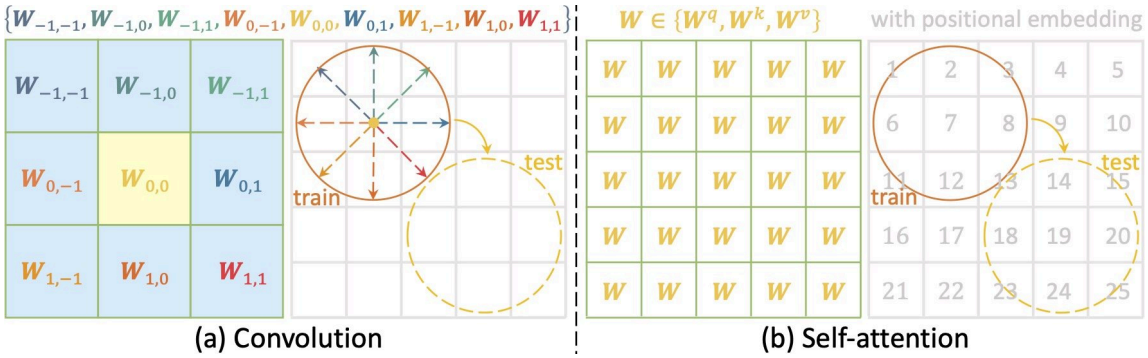

Figure 2. Difference between convolution and self-attention in encoding relevant elements: consider the scenario where convolution and self-attention are capturing the structure of a circle. 1) Convolution learns separate parameters $\{\mathbf{W}_{\delta_x,\delta_y}\}$ for each offset, where $\delta_x, \delta_y \in [-1, 1]$, from the kernel center, allowing it to effectively encode relative local structures. Thus, when the circle appears in a different location, it is still readily recognized due to this relative awareness. 2) Self-attention incorporates absolute position into each token's representation and uses position-irrelevant parameters $\mathbf{W} \in \{\mathbf{W}^q, \mathbf{W}^k, \mathbf{W}^v\}$ across all tokens for computing query, key and value, respectively. While this method facilitates general processing, the inclusion of absolute positional embeddings makes it more challenging to recognize the circle when it is moved to a different location.

1.3. Unification of Convolution and Transformer

In summary, convolution encodes structure through fixed local filters with position-specific weights, whereas self-attention relies on adaptive global attention and requires absolute positional encoding to capture order or spatial structures.

In this paper, we introduce Translution, a new type of operation that unifies the adaptive identification capability of self-attention with the relative encoding advantage of convolution. Specifically, Translution employs a convolution-style approach that assigns separate parameters (matrices) to each distance and direction when computing the query, key and value. This design enables Translution to effectively encode relative structures.

However, this unification leads to a significant increase in the number of parameters and exceeds most currently available computational resources. Therefore, we propose a lightweight variant of Translution, named α-Translution, which significantly reduces the number of parameters. This variant achieves lower accuracy than the "ideal" (original) Translution but better accuracy than self-attention.

doi.org/10.32388/Q6KCoC.2

As a fundamental operation, we investigate whether Translution can outperform self-attention. We conduct experiments on two widely-used Transformer architectures: Vision Transformer (ViT)[4] for computer vision tasks and Generative Pre-trained Transformer (GPT)[2][10][11] for natural language processing tasks. Experiments demonstrate that Translution and α-Translution surpass self-attention in terms of accuracy.

2. Related Work

Transformer[1][2][3][4][12][13] eschews recurrence (as used in recurrent neural networks) and kernel size (as used in convolutional neural networks), instead employing self-attention for relevant region identification. Because it has no built-in notion of order, Transformer incorporates explicit absolute positional embeddings into token embeddings, enabling the model to utilize sequence order. Subsequent work has explored "relative attention"[14][15][16][17][18][19][20], which integrates relative position information into self-attention. They can be categorized into three families: 1) *Relative positional vector.* Shaw *et al*.enhanced Transformer for language modeling by adding learnable relative positional vectors into the key and value computations, respectively[14]. BoTNet[21] and HaloNet[22] extended this approach to two dimensions for image processing by adding learnable relative positional vectors into key. 2) *Relative positional scalar.* Swin Transformer[12], CoAtNet[20], and ConViT d'Ascoli et al.[23] incorporate a learnable relative positional bias (a scalar) into the attention score. In these methods, the original self-attention can be regarded as content attention, which measures relationships from the token-feature perspective, while the additional relative positional bias can be regarded as position attention, which measures relationships from the token-position perspective. 3) *Rotary position embedding.* RoFormer[24] introduces a rotary position embedding mechanism, which encodes relative positional information by applying a rotation operation in the Query and Key representation space. Unlike these existing methods, Translution employs a convolution-style approach that uses relative positional matrices for query, key and value computation. Section D provides a formal comparison of these methods.

Convolutional neural networks[5][6][7][8][9] have been the backbone of deep learning for years. By using small, shared kernels and pooling, convolutional neural networks efficiently capture local patterns. Recent architectural developments integrate self-attention with convolution. For instance, Conformer[25] combines convolution layers and self-attention layers to capture both local and global

dependencies in audio sequences. Similarly, CeiT[26] uses convolutions to extract low-level features and self-attention to model long-range dependencies. Unlike these architectural methods, Translution operates at the basic module or layer level, blending the advantages of self-attention and convolution into a unified fundamental operation.

3. Preliminary: Convolution and Self-attention

3.1. Convolution

Suppose $\mathbf{f}_{x,y} \in \mathbb{R}^{1\times C}$ denotes the feature or representation at location (x, y) in an image of height H and width W, where C is the number of the input feature channels. Convolution is designed to capture the local structure centered at (x, y) with a fixed kernel size $h \times w$,

$$\mathbf{f}'_{x,y} = \sum_{\delta_x=-\lfloor h/2 \rfloor}^{\lfloor h/2 \rfloor} \sum_{\delta_y=-\lfloor w/2 \rfloor}^{\lfloor w/2 \rfloor} \mathbf{f}_{x+\delta_x,y+\delta_y} \cdot \mathbf{W}_{\delta_x,\delta_y},$$

where $\mathbf{W}_{\delta_x,\delta_y} \in \mathbb{R}^{C\times C'}$ denotes the learnable parameters corresponding to the displacement (δ_x, δ_y), C' indicates the output feature dimension, and $\cdot$ denotes matrix multiplication. By assigning a set of parameters for each offset within the receptive field, convolution is able to discern direction and distance, and capture the local structure relatively. This means that when the absolute location of an object changes, it can still capture the same structure. However, convolution employs a rigid method to identify relevant regions, *i.e.*, using a fixed-size window, making it inevitably include irrelevant pixels or regions — particularly near object boundaries or in background areas within the window.

3.2. Self-attention

Suppose $\mathbf{x}_i \in \mathbb{R}^{1\times C}$ represents the feature or representation of the i-th patch at location (x_i, y_i). Transformer[1] first incorporates the embedding of absolute position into the input $\mathbf{x}_i$, as follows,

$$\text{input positional embedding: } \mathbf{f}_i = \mathbf{x}_i + \text{Embed}(x_i, y_i).$$

Then, self-attention performs two separate linear projections on the feature to generate query $\mathbf{q}_i \in \mathbb{R}^{1\times C'}$ and key $\mathbf{k}_j \in \mathbb{R}^{1\times C'}$, where C' is the dimension for query or key,

$$\begin{aligned} \text{query encoding: } \mathbf{q}_i &= \mathbf{f}_i \cdot \mathbf{W}^q, \\ \text{key encoding: } \mathbf{k}_j &= \mathbf{f}_j \cdot \mathbf{W}^k, \end{aligned}$$

where $\mathbf{W}^q/\mathbf{W}^k \in \mathbb{R}^{C\times C'}$. Subsequently, scaled dot-product attention is computed for each query, and a softmax function is applied to normalize the attention weights for a query across all positions,

$$\text{attention: } a_{i,j} = \frac{\mathbf{q}_i \cdot \mathbf{k}_j^T}{\sqrt{C'}}, \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}},$$

where $N = H \times W$. Next, self-attention conducts another linear projection on the input feature to generate value $\mathbf{v}_i \in \mathbb{R}^{1\times C'}$, as follows,

$$\text{value encoding: } \mathbf{v}_j = \mathbf{f}_j \cdot \mathbf{W}^v,$$

where $\mathbf{W}_v \in \mathbb{R}^{C\times C'}$. Finally, the output is computed as a weighted sum of the values, *i.e.*,

$$\text{weighted sum: } \mathbf{f}_i' = \sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{v}_j,$$

where $\mathbf{f}_i' \in \mathbb{R}^{1\times C'}$. In this way, self-attention can adaptively search for related regions, providing greater flexibility than methods that use local fixed-size windows. However, unlike convolution, which learns a feature encoding for every direction and distance, self-attention does not encode the structure in a relative manner.

3.3. Translution

Translution is designed to integrate the adaptive related region identification capabilities of self-attention with the relative encoding strengths of convolution. Specifically, as shown in Figure 3, Translution employs a convolution-style formulation by assigning different parameters to compute query, key, and value, respectively, as follows:

$$\text{Translution} \begin{cases} \text{relative query encoding: } \mathbf{q}_{i,j} = \mathbf{f}_i \cdot \mathbf{W}^q_{\delta_x,\delta_y}, \delta_x = x_i - x_j, \delta_y = y_i - y_j, \\ \text{relative key encoding: } \mathbf{k}_{j,i} = \mathbf{f}_j \cdot \mathbf{W}^k_{-\delta_x,-\delta_y}, \\ \text{relative attention: } a_{i,j} = \frac{\mathbf{q}_{i,j} \cdot \mathbf{k}_{j,i}^T}{\sqrt{C'}}, \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}}, \\ \text{relative value encoding: } \mathbf{v}_{i,j} = \mathbf{f}_j \cdot \mathbf{W}^v_{\delta_x,\delta_y}, \\ \text{weighted sum: } \mathbf{f}_i' = \sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{v}_{i,j}, \end{cases} \tag{1}$$

where $\mathbf{W}^q_{\delta_x,\delta_y}/\mathbf{W}^k_{\delta_x,\delta_y}/\mathbf{W}^v_{\delta_x,\delta_y} \in \mathbb{R}^{C\times C'}$, represent the learnable parameter matrices for the query, key, and value corresponding to the displacement (δ_x, δ_y).

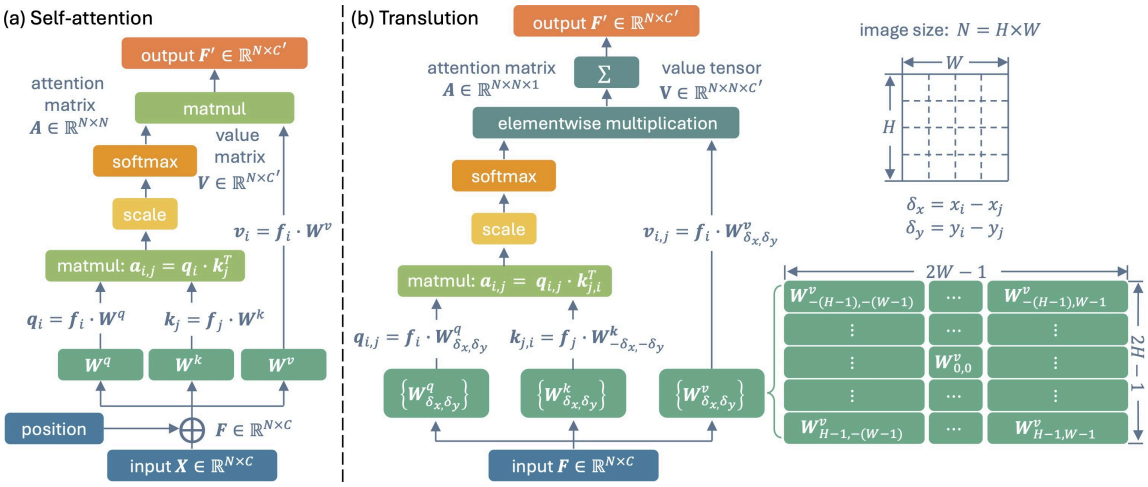

Figure 3. Comparison of self-attention and Translution. 1) Self-attention employs three shared sets of weights, *i.e.*, $\mathbf{W}^q$, $\mathbf{W}^k$, and $\mathbf{W}^v$, across all patches to compute query, key, and value, respectively. 2) Translution uses separate parameters for each offset (direction and distance), *i.e.*, $\{\mathbf{W}^q_{\delta_x,\delta_y}\}$, $\{\mathbf{W}^k_{\delta_x,\delta_y}\}$ and $\{\mathbf{W}^v_{\delta_x,\delta_y}\}$, to encode relative structures.

Translution unifies convolution and self-attention

The fixed receptive field in convolution can be interpreted as a special case of attention, where the attention score is set to 1 within the receptive field and 0 outside it, as shown in Figure 2. The weights $\mathbf{W}^q$, $\mathbf{W}^k$, and $\mathbf{W}^v$ in self-attention serve as shared linear projections that are uniformly applied across all spatial directions and distances. Consequently, Translution integrates the functionalities of convolution and self-attention, as follows,

$$\begin{aligned}
&\text{Convolution:} && \mathbf{f}'_i = \textstyle\sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{f}_j \cdot \mathbf{W}_{\delta_x,\delta_y}, && \text{where } \alpha_{i,j} = \begin{cases} 1, & (\delta_i,\delta_j) \in \text{kernel}, \\ 0, & \text{otherwise}. \end{cases} \\
&\text{Self - attention:} && \mathbf{f}'_i = \textstyle\sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{f}_j \cdot \mathbf{W}^v, && \text{where } a_{i,j} = \frac{\mathbf{q}_i \mathbf{k}_j^T}{\sqrt{C'}}, \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}}. \\
&\text{Translution:} && \mathbf{f}'_i = \textstyle\sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{f}_j \cdot \mathbf{W}^v_{\delta_x,\delta_y}, && \text{where } a_{i,j} = \frac{\mathbf{q}_{i,j} \cdot \mathbf{k}^T_{j,i}}{\sqrt{C'}}, \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}}.
\end{aligned}$$

In other words, convolution and self-attention can be viewed as specific instances of Translution, where convolution simplifies the attention mechanism and self-attention omits the encoding of direction and distance.

doi.org/10.32388/Q6KCoC.2

3.4. α-Translution

Suppose there are $H \times W$ input image patches. The relative encoding method in Translution requires $(2H-1) \times (2W-1) \times C \times C'$ parameters. Specifically, it requires one parameter matrix $\mathbf{W}^q_{\delta_x,\delta_y}$, $\mathbf{W}^k_{\delta_x,\delta_y}$ or $\mathbf{W}^v_{\delta_x,\delta_y} \in \mathbb{R}^{C \times C'}$ for each relative position (δ_x, δ_y), where $\delta_x \in \{-(H-1), \cdots, 0, \cdots, H-1\}$ and $\delta_y \in \{-(W-1), \cdots, 0, \cdots, W-1\}$. This approach leads to excessive parameter demands, making it impractical for most computational devices currently. For instance, in the ViT/16 architecture[4] with input resolution 224×224, we have $H = W = \frac{224}{16} = 14$, resulting in $(2H-1) \times (2W-1) = 729$ distinct weight matrices for query, key or value. To reduce the number of parameters, we propose a variant of Translution, *i.e.*, α-Translution, which decreases both the input dimension C and the output dimension C' of each $\mathbf{W}^q_{\delta_x,\delta_y}$, $\mathbf{W}^k_{\delta_x,\delta_y}$, and $\mathbf{W}^v_{\delta_x,\delta_y}$, as follows:

$$\mathbf{W}^q_{\delta_x,\delta_y} \Rightarrow \mathbf{W}^q_1 \cdot \mathbf{W}^q_{\delta_x,\delta_y}, \mathbf{W}^k_{\delta_x,\delta_y} \Rightarrow \mathbf{W}^k_1 \cdot \mathbf{W}^k_{\delta_x,\delta_y}, \mathbf{W}^v_{\delta_x,\delta_y} \Rightarrow \mathbf{W}^v_1 \cdot \mathbf{W}^v_{\delta_x,\delta_y} \cdot \mathbf{W}^v_2,$$

where $\mathbf{W}^q_1/\mathbf{W}^k_1/\mathbf{W}^v_1 \in \mathbb{R}^{C \times C^1}$, $\mathbf{W}^q_{\delta_x,\delta_y}/\mathbf{W}^k_{\delta_x,\delta_y}/\mathbf{W}^v_{\delta_x,\delta_y} \in \mathbb{R}^{C^1 \times C^2}$, $\mathbf{W}^v_2 \in \mathbb{R}^{C^2 \times C'}$, and $C^1 \ll C$, $C^2 \ll C'$. Smaller values of C^1 and C^2 will significantly reduce the number of parameters.

However, setting C^1 and C^2 too small may overly compress the query, key and value information, negatively impacting performance. To preserve the information, we incorporate the query, key and value computation mechanism of self-attention into α-Translution. Specifically, the updated computation is defined as follows:

$$\alpha - \text{Translution} \begin{cases} \text{query encoding: } \mathbf{q}_{i,j} = \mathbf{f}_i \cdot \mathbf{W}^q_1 \cdot \mathbf{W}^q_{\delta_x,\delta_y}, \mathbf{q}_i = \mathbf{f}_i \cdot \mathbf{W}^q, \\ \text{key encoding: } \mathbf{k}_{j,i} = \mathbf{f}_j \cdot \mathbf{W}^k_1 \cdot \mathbf{W}^k_{-\delta_x,-\delta_y}, \mathbf{k}_j = \mathbf{f}_j \cdot \mathbf{W}^k, \\ \text{attention: } a_{i,j} = \frac{\mathbf{q}_{i,j} \cdot \mathbf{k}^T_{j,i} + \mathbf{q}_i \cdot \mathbf{k}^T_j}{\sqrt{C'}}, \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}}, \\ \text{value } \textit{encoding} : \mathbf{v}_{i,j} = \mathbf{f}_j \cdot (\mathbf{W}^v_1 \cdot \mathbf{W}^v_{\delta_x,\delta_y} \cdot \mathbf{W}^v_2 + \mathbf{W}^v), \\ \text{weighted sum:} \mathbf{f}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{v}_{i,j}. \end{cases} \tag{2}$$

In this way, α-Translution not only possesses relative modeling capability but also reduces the number of parameters.

4. Experiment

In this section, as a fundamental operation, our primary objective is to compare Translution with self-attention, rather than to achieve state-of-the-art performance through specialized network architectures

doi.org/10.32388/Q6KCoC.2

or extensive training techniques. To this end, we conduct experiments using two widely adopted Transformer architectures:

- Vision Transformer (ViT)[4] for computer vision tasks.
- Generative Pre-trained Transformer (GPT)[2][10][11] for natural language processing tasks. Section C demonstrates how to apply Translution to text modeling.

Table 1 provides an overview of various architecture configures. We substitute self-attention in ViT and GPT with Translution, while maintaining the remaining architecture unchanged.

Architecture	Depth (#Layers)	Embedding Dim (Hidden size)	#Heads	MLP Dim (Feedforward)
A	6	192	3	768
B	12	192	3	768
C	12	384	6	1,536

Table 1. Specifics of architecture configures used in this paper.

Due to limited computational resources, our evaluation is primarily conducted on small- and medium-scale architectures. Large-scale evaluation can be performed when single-GPU memory capacities approach approximately $2 \sim 3$ TB. All training starts from scratch. The default compression dimensions for the relative encoding in α-Translution are set as $C^1 = C^2 = 8$.

4.1. Image Classification with ViT

4.1.1. Dynamic MNIST

To evaluate the capability of modeling relative structure, we synthesize a dynamic MNIST dataset[27][28], where digits (originally sized 28×28 pixels) move within a 84×84 pixel area, as illustrated in Figure 4. For comparison, we also create a static MNIST dataset of the same size, where digits remain fixed at the center of each image.

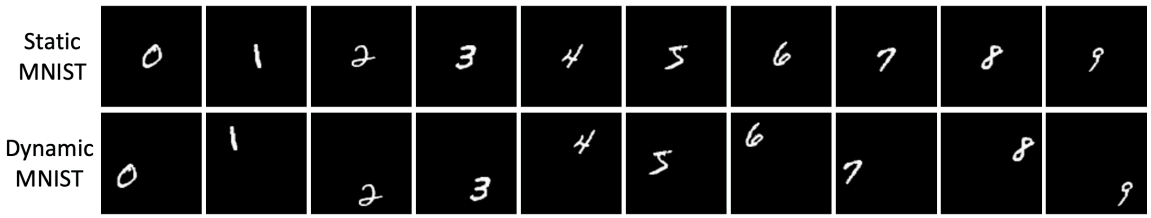

Figure 4. Examples of static and dynamic MNIST. Static MNIST digits are fixed at the center of images, whereas dynamic MNIST digits are randomly positioned within the images.

Arch.	Method	#Params	Static→Static	Dynamic→Dynamic	Static→Dynamic
ViT-A/12	Self-attention[1]	2.7 M	98.48	92.64	18.18
	α-Translution (relative dim = 8)	4.6 M	98.48	97.31	34.90
	Translution	116.2 M	**98.60**	**97.35**	**36.40**
ViT-A/7	Self-attention[1]	2.7 M	98.52	93.90	19.94
	α-Translution (relative dim = 8)	8.3 M	98.81	98.57	40.05
	Translution	355.0 M	**98.91**	**98.60**	**48.07**

Table 2. Top-1 accuracy (%) on different MNIST settings with the ViT-A architecture. $\mathcal{A} \rightarrow \mathcal{B}$ denotes that models are trained on dataset $\mathcal{A}$ and evaluated on dataset $\mathcal{B}$.

As shown in Table 2, all models achieve high accuracy when trained and evaluated on static MNIST. However, when digit locations vary, the self-attention's accuracy significantly decreases, whereas Translution (including α-Translution) still maintains high accuracy. This is because absolute positional embedding makes digit locations part of its representation. Consequently, when digits shift positions, networks may become confused and fail to recognize digits accurately. In contrast, Translution employs relative encoding, effectively capturing digit structures independently of their absolute locations. This significantly reduces sensitivity to location variability, demonstrating Translution's superior capability in modeling relative structures. However, when training on static MNIST, the uniformly black image

doi.org/10.32388/Q6KCoC.2

background causes some $\mathbf{W}_{\delta_x,\delta_y}$ not to be well trained. As a result, when evaluated on dynamic MNIST, Translution fails to achieve very high accuracy.

doi.org/10.32388/Q6KCoC.2

Architecture	Method	#Parameters	Top-1	Top-5
ViT-A/56	Self-attention[1]	4.7 M	46.28	71.17
	α-Translution (relative enc dim = 8)	5.3 M	48.36	73.31
	Translution	38.5 M	**52.41**	**76.50**
ViT-B/56	Self-attention[1]	7.4 M	53.75	77.59
	α-Translution (relative enc dim = 8)	8.7 M	55.87	79.16
	Translution	75.0 M	**59.51**	**81.97**
ViT-C/56	Self-attention[1]	25.3 M	64.15	84.95
	α-Translution (relative enc dim = 8)	30.5 M	66.54	86.49
	Translution	296.0 M	**68.05**	**88.62**
	Self-attention[1]	3.5 M	57.63	80.96
ViT-A/32	α-Translution (relative enc dim = 8)	5.3 M	60.26	83.07
	Translution	116.9 M	**66.03**	**86.01**
	Self-attention[1]	6.1 M	66.13	86.87
ViT-B/32	α-Translution (relative enc dim = 8)	9.9 M	67.63	87.96
	Translution	223.1 M	**70.63**	**90.10**
Translution runs out of memory under the following architectures.				
ViT-A/32	Self-attention[1]	22.9 M	73.62	91.12
	α-Translution (relative enc dim = 8)	38.0 M	**74.19**	**91.52**
ViT-A/16	Self-attention[1]	3.0 M	64.71	86.25
	α-Translution (relative enc dim = 8)	10.7 M	**69.28**	**89.24**
ViT-B/16	Self-attention[1]	5.7 M	73.51	91.89
	α-Translution (relative enc dim = 8)	21.1 M	**76.20**	**93.04**

Table 3. Accuracy (%) on the ImageNet-1K dataset with patch sizes of 56 and 32. Training is conducted from

doi.org/10.32388/Q6KCoC.2

scratch without pretraining on external datasets, with a batch size of 256.

4.1.2. ImageNet

ImageNet-1K Deng et al.[29] is a widely used dataset for computer vision research, particularly in the area of image classification. It contains 1,000 object categories (classes), each with approximately 1,300 training images and 50 validation images, amounting to about 1.28 million training images and 50,000 validation images in total. Images are resized to 224×224. As shown in Table 3, compared to self-attention[1], Translution and α-Translution effectively improve ImageNet classification.

We compare Translution with existing positional encoding strategies, which typically represent positional information by introducing additional positional biases, as scalars Liu et al.[12]; d'Ascoli et al.[23] or vectors[1][14]. The formal differences between these approaches are detailed in Section D. As shown in Table 4, compared to existing relative encoding methods, Translution achieves a notable improvement in accuracy.

Method	#Parameters	Top-1	Top-5
Self-attention w/o Pos Emb	4.69 M	42.49	67.39
Self-attention w/ Pos Emb[1]	4.69 M	46.28	71.17
Relative key vector[14]	4.74 M	46.39	71.25
Relative value vector[14]	4.74 M	46.35	71.04
Swin Transformer[12]	4.69 M	46.36	71.31
ConViT[23]	4.69 M	46.39	71.44
RoFormer[24]	4.69 M	46.65	71.51
α-Translution	5.33 M	48.36	73.31
Translution	38.53 M	52.41	76.50

Table 4. Comparison of different positional encoding strategies. Results are reported on ImageNet-1K with ViT-A/56, trained from scratch (no external pretraining) using a batch size of 256.

doi.org/10.32388/Q6KCoC.2

4.1.3. Ablation Study

1) Is the improvement of Translution (including α-Translution) caused by the introduction of additional parameters or the proposed modeling approach based on relative encoding?

Compared to self-attention, which employs three parameter matrices $\mathbf{W}^q$, $\mathbf{W}^k$, $\mathbf{W}^v$ to compute query, key and value, Translution uses three groups of parameter matrices $\{\mathbf{W}^q_{\delta_x,\delta_y}\}$, $\{\mathbf{W}^k_{\delta_x,\delta_y}\}$, $\{\mathbf{W}^v_{\delta_x,\delta_y}\}$ for relative encoding, thus introducing more parameters.

To investigate whether the improvement arises from the increased parameter count or from the relative encoding method itself, we conducted the following experiment:

$$\text{relative encoding: } \mathbf{W}^q_{\delta_x,\delta_y}, \mathbf{W}^k_{\delta_x,\delta_y}, \mathbf{W}^v_{\delta_x,\delta_y} \Rightarrow \text{absolute encoding: } \mathbf{W}^q_{i,j}, \mathbf{W}^k_{i,j}, \mathbf{W}^v_{i,j},$$

where $\delta_x \in \{-(H-1), \cdots, 0, \cdots, H-1\}$, $\delta_y \in \{-(W-1), \cdots, 0, \cdots, W-1\}$, and indices $i \in [1, H \times W]$ and $y \in [1, H \times W]$. Specifically, for each pair of patches (i, j), a distinct parameter matrix is employed to calculate query, key or value, rather than using the shared offset-based matrices. Under this modification, Translution transitions to absolute modeling. Moreover, this adjustment significantly increases the number of parameter matrices from $(2H-1) \times (2W-1)$ to $(H \times W)^2$.

Method	Encoding	#Parameters	Static→Static	Dynamic→Dynamic	Static→Dynamic
α-Translution	relative	4.6 M	**98.48**	**97.31**	**34.90**
	absolute	28.7 M	98.42	96.18	25.37
Translution	relative	116.2 M	**98.60**	**97.35**	**36.24**
	absolute	1660.9 M	98.55	53.79	11.23

Table 5. Investigation of whether the improvement of Translution arises from the additional parameters or the proposed relative encoding method ($\mathbf{W}^q_{\delta_x,\delta_y}$, $\mathbf{W}^k_{\delta_x,\delta_y}$, $\mathbf{W}^v_{\delta_x,\delta_y}$). Because the absolute encoding method ($\mathbf{W}^q_{i,j}$, $\mathbf{W}^k_{i,j}$, $\mathbf{W}^v_{i,j}$) consumes a large number of parameters, Translation with ViT-A/7 encounters the out-of-memory issue. Therefore, experiments are conducted using ViT-A/12.

As shown in Table 5, although absolute encoding involves significantly more parameters, it achieves

lower accuracy than relative encoding. Therefore, simply increasing the number of parameters does not lead to performance improvements.

2) Impact of relative encoding dimension on the performance of α-Translution.

To reduce parameter usage, α-Translution employs smaller input (C^1) and output (C^2) dimensions for $\{\mathbf{W}^q_{\delta_x,\delta_y}\}$, $\{\mathbf{W}^k_{\delta_x,\delta_y}\}$ and $\{\mathbf{W}^v_{\delta_x,\delta_y}\}$. In our experiments, we set the relative encoding dimensions as $C^1 = C^2 = 8$. This section investigates the impact of varying C^1 and C^2 on performance. As shown in Table 6, increasing the relative encoding dimension improves accuracy but results in more parameters. Therefore, the relative encoding dimension presents a trade-off between efficiency and effectiveness for α-Translution. (When $C^1 = C^2 = 0$, it reduces to self-attention without positional embedding.)

Relative Enc Dim	#Params	Top-1	Top-5	Relative Enc Dim	#Params	Top-1	Top-5
$C^1 = C^2 = 0$	4.7 M	42.49	67.39	$C^1 = C^2 = 8$	5.3 M	48.36	73.31
$C^1 = C^2 = 2$	4.8 M	46.10	71.29	$C^1 = C^2 = 16$	7.0 M	48.91	73.65
$C^1 = C^2 = 4$	4.9 M	47.61	72.18	$C^1 = C^2 = 32$	13.8 M	50.07	74.84

Table 6. Impact of relative encoding dimension on the performance of α-Translution with ViT-A/56.

4.2. Natural Language Modeling with GPT

To compare Translution and Transformer for natural language processing, we conduct experiments using the OpenWebText dataset[30], an openly available reproduction of OpenAI's proprietary WebText dataset used for GPT-2[10]. OpenWebText contains 9 billion training tokens and 4 million validation tokens, with a vocabulary size of 50K. We use perplexity, defined as the exponentiation of the cross-entropy loss, as the evaluation metric, where a lower perplexity indicates stronger language modeling performance. Since the most powerful GPU available to us has 80GB memeory, Translution can handle at most a text sequence of length 160 with the GPT-A architecture. Therefore, we conduct the Translution experiment with sequences of length 160. As shown in Table 7, Translution achieves lower perplexity compared to Transformer, demonstrating its effectiveness in natural language modeling.

Architecture	Method	#Parameters	Perplexity ↓
GPT-A-160	Self-attention[1]	22.0 M	60.40
	α-Translution (relative enc dim = 8)	23.7 M	57.97
	Translution	127.5 M	**56.26**
Translution runs out of memory under the following architectures.			
GPT-B-160	Self-attention[1]	24.7 M	54.82
	α-Translution (relative enc dim = 8)	28.2 M	**52.72**
GPT-C-160	Self-attention[1]	60.0 M	39.88
	α-Translution (relative enc dim = 8)	74.0 M	**39.25**

Table 7. Perplexity on OpenWebText using a batch size of 8 and a sequence length of 160.

5. Conclusion

In this paper, we introduce Translution, a new operation that unifies self-attention and convolution for adaptive and relative modeling. Experiments on computer vision and natural language processing tasks demonstrate the effectiveness of Translution.

However, due to current limited computational resources, the validation in this paper is preliminary. We encourage the community to further evaluate Translution using larger-scale frameworks and datasets in diverse scenarios to verify its broader applicability, particularly when single GPUs equipped with over $2 \sim 3$ TB of memory are available.

Given Translution's substantial parameter consumption, it is worthwhile to explore optimized variants, such as α-Translution. For instance, certain relative positions may share the same parameter, especially when the distance between elements is too long. At the same time, extending Translution to 3D, video, molecule, and other modalities of processing holds significant promise.

As a fundamental operation, Translution can be employed beyond the ViT and GPT architectures. More effective and efficient architectures for Translution merit further exploration in future.

doi.org/10.32388/Q6KCoC.2

Appendix A. Default Notation

a, A	A scalar	$\mathbf{a}$	A vector
$\mathbf{A}$	A matrix	$\mathbf{A}$	A tensor
$\times$	Scalar multiplication	$\cdot$	Matrix multiplication

Appendix B. General Translution

The calculation of the query, key and value in Translution, *i.e.*, Eq. (1), assumes that element positions (*e.g.*, image patches or textual words) are discrete. In this setting, it is feasible to assign a different set of parameters for each direction and distance. However, if the positions are continuous variables, *e.g.*, in point clouds, it becomes impractical to assign individual weights for each direction and distance, as there are infinitely many possible variations in continuous space. In this case, it may be necessary to design new functions for the relative encoding.

Suppose $\mathbf{p}_i$ denotes the position of the i-th element. For language, $\mathbf{p}_i$ can represent the index of the i-th word in the text. For images, $\mathbf{p}_i$ corresponds to the row and column indices of the i-th patch. For point clouds, $\mathbf{p}_i$ refers to the 3D coordinates of the i-th point. A more general version of Translution can be formulated as follows,

$$\text{General Translution: } \mathbf{f}_i' = \sum_{j=1}^{N} \alpha(\mathbf{p}_i - \mathbf{p}_j, \mathbf{f}_i, \mathbf{f}_j,) \times v(\mathbf{p}_i - \mathbf{p}_j, \mathbf{f}_j),$$

where $\alpha \in [0, 1]$ denotes the attention score measuring the relevance of the j-th element to the i-th element, and $v : \mathbb{R}^{d+C} \to \mathbb{R}^{C'}$ is a function that encodes relative positional information into the element features (d denotes the dimensionality of the position, C is the number of input feature channels, and C' is the number of output feature channels). When applying Translution to a new type of data, the key is to develop effective α and v functions.

Appendix C. 1D Translution for Natural Language Processing

In the main text, we demonstrate how to apply Translution for image modeling. That Translution can be viewed as a 2D operation because the relative encoding involves two spatial directions. However, in

natural language, relative encoding operates along a single dimension, which makes Translution a one-dimensional model when applied to text.

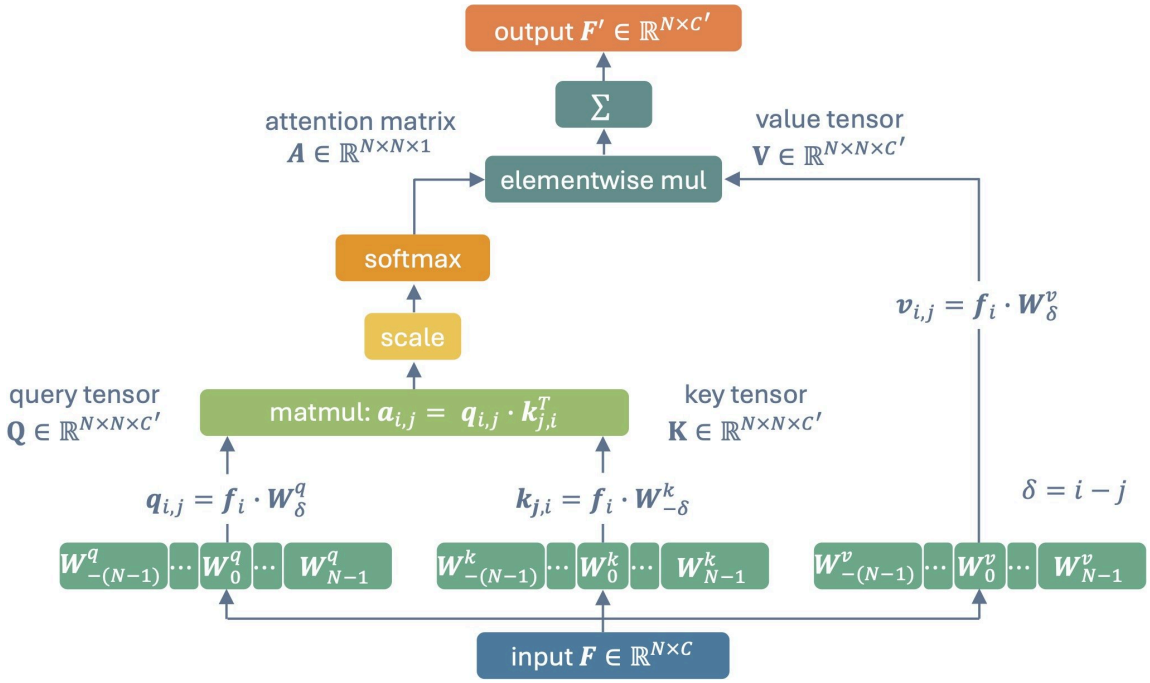

Figure 5. When modeling text, Translution operates in a 1D setting. For a sequence of length N, it employs separate parameters for each positional offset (considering both direction and distance), *i.e.*, $\{\mathbf{W}^q_{-(N-1)}, \cdots, \mathbf{W}^q_0, \cdots, \mathbf{W}^q_{N-1}\}$, $\{\mathbf{W}^k_{-(N-1)}, \cdots, \mathbf{W}^k_0, \cdots, \mathbf{W}^k_{N-1}\}$ and $\{\mathbf{W}^v_{-(N-1)}, \cdots, \mathbf{W}^v_0, \cdots, \mathbf{W}^v_{N-1}\}$, to encode relative language structure.

Suppose $\mathbf{f}_i \in \mathbb{R}^{1\times C}$ denotes the embedding (or representation) of the i-th token within a text sequence of length N, where C represents the embedding dimension. As shown in Figure 5, 1D Translution is designed to integrate adaptive identification of related tokens with relative structural encoding for language modeling. Specifically, Translution retains the self-attention mechanism of the Transformer but employs distinct parameters for computing the Query, Key and Value representations, as follows,

$$\text{1D Translution} \begin{cases} \text{relative query encoding: } \mathbf{q}_{i,j} = \mathbf{f}_i \cdot \mathbf{W}^q_\delta, \delta = i - j, \\ \text{relative key encoding: } \mathbf{k}_{j,i} = \mathbf{f}_j \cdot \mathbf{W}^k_{-\delta}, \\ \text{relative attention: } a_{i,j} = \frac{\mathbf{q}_{i,j} \cdot \mathbf{k}^T_{j,i}}{\sqrt{C'}}, \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}}, \\ \text{relative value encoding: } \mathbf{v}_{i,j} = \mathbf{f}_j \cdot \mathbf{W}^v_\delta, \\ \text{weighted sum: } \mathbf{f}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{v}_{i,j}, \end{cases}$$

doi.org/10.32388/Q6KCoC.2

where $\mathbf{W}_{\delta}^{q}/\mathbf{W}_{\delta}^{k}/\mathbf{W}_{\delta}^{v} \in \mathbb{R}^{C\times C'}$ denotes the learnable parameters for displacement δ.

Causal 1D Translution

For autoregressive tasks, such as language modeling in GPT, a causal variant is typically required to ensure future tokens remain unseen during inference. In causal 1D Translution, each token attends only to itself and preceding tokens, guaranteeing that predictions rely exclusively on past context, as follows,

$$
\text{Causal 1D Translution}\begin{cases}
\text{relative query encoding: } \mathbf{q}_{i,j} = \mathbf{f}_i \cdot \mathbf{W}_{\delta}^{q}, \delta = i-j, \\
\text{relative key encoding: } \mathbf{k}_{j,i} = \mathbf{f}_j \cdot \mathbf{W}_{-\delta}^{k}, \\
\text{relative attention: } a_{i,j} = \frac{\mathbf{q}_{i,j}\cdot\mathbf{k}_{j,i}^{T}}{\sqrt{C'}}, \\
\text{causal attention: } a'_{i,j} = \begin{cases} a_{i,j}, & i \geq j, \\ -\infty, & \text{otherwise}, \end{cases} \alpha_{i,j} = \frac{e^{a'_{i,j}}}{\sum_{n=1}^{N} e^{a'_{i,n}}}, \\
\text{relative value encoding: } \mathbf{v}_{i,j} = \begin{cases} \mathbf{f}_j \cdot \mathbf{W}_{\delta}, & \delta = i-j \geq 0, \\ \forall, & \text{otherwise}, \end{cases} \\
\text{weighted sum: } \mathbf{f}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{v}_{i,j}.
\end{cases}
$$

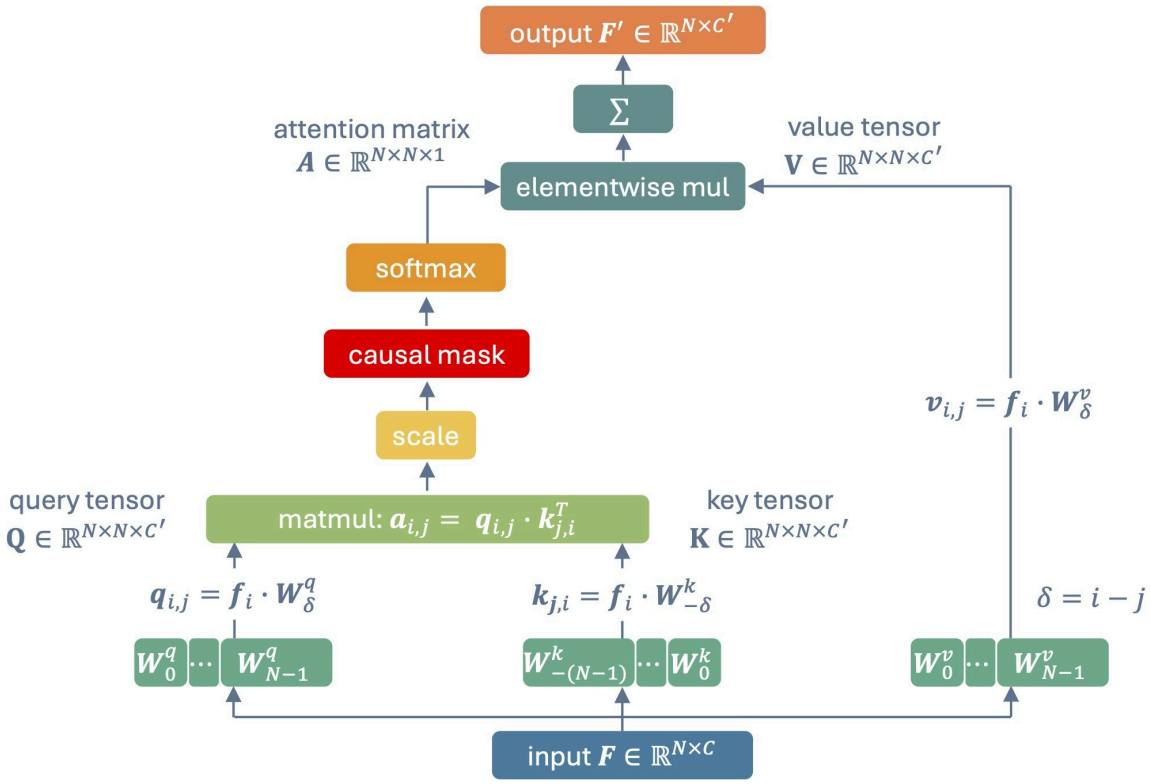

Figure 6. Illustration of causal 1D Translution. For a sequence of length N, it employs N parameter matrices to encode relative language structure. Compared to the original 1D Translution, the causal variant reduces the number of parameters required to compute Query, key and Value by half.

doi.org/10.32388/Q6KCoC.2

As shown in Figure 6, compared to the original variant, causal 1D Translution reduces by half the number of parameters needed to compute the query, key and value representations.

Appendix D. Memory-Efficient Implementation of α-Translution: Optimizing Runtime Memory Usage

Recall that α-Transformer is defined as follows,

$$\alpha - \text{Translution: } \mathbf{f}_i' = \sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{f}_j \cdot (\mathbf{W}^v + \mathbf{W}^{v1} \cdot \mathbf{W}^v_{\delta_x,\delta_y} \cdot \mathbf{W}^{v2}),$$

where $\mathbf{W}^v \in \mathbb{R}^{C \times C'}$, $\mathbf{W}^{v1} \in \mathbb{R}^{C \times C^1}$, $\mathbf{W}^v_{\delta_x,\delta_y} \in \mathbb{R}^{C^1 \times C^2}$, $\mathbf{W}^{v2} \in \mathbb{R}^{C^2 \times C'}$, and $C^1 \ll C$, $C^2 \ll C'$. Although this variant significantly reduces the number of parameters, it still demands considerable runtime memory. Specifically, as shown in Figure 3, the resulting value tensor of Translution is $\mathbf{V} \in \mathbb{R}^{N \times N \times C'}$, which is considerably larger than the Transformer's value matrix $\mathbf{V} \in \mathbb{R}^{N \times C'}$. To address this issue, we implement α-Translution as follows,

$$\mathbf{f}_i' = \sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{f}_j \cdot \mathbf{W}^v + \left(\sum_{j=1}^{N} \alpha_{i,j} \times \mathbf{f}_j \cdot (\mathbf{W}^{v1} \cdot \mathbf{W}^v_{\delta_x,\delta_y}) \right) \cdot \mathbf{W}^{v2}.$$

This reformulation reduces the peak runtime memory usage from $N \times N \times C'$ to $N \times C' + N \times N \times C^2$, where $C^2 \ll C'$, thus significantly alleviating memory demands during computation.

Appendix E. Comparison with Existing Position Modeling Methods

Existing methods typically encode positional information by introducing additional positional biases (either scalars or vectors). In this paper, inspired by convolution, we propose an alternative approach that employs offset-based matrices for relative encoding. In this section, we provide a detailed comparison between these approaches. Suppose $\mathbf{x}_i \in \mathbb{R}^{1 \times C}$ represents the feature or representation of the i-th patch, located at (x_i, y_i) in an image composed of $N = H \times W$ patches.

1. Baseline (Self-attention w/o Positional Embedding)

We consider the self-attention without position embedding as the baseline, formulated as follows:

doi.org/10.32388/Q6KCoC.2

$$
\begin{aligned}
&\text{w/o input position embedding:}\quad \boldsymbol{f}_i = \boldsymbol{x}_i,\\
&\text{query encoding:}\quad \boldsymbol{q}_i = \boldsymbol{f}_i \cdot \boldsymbol{W}_q,\\
&\text{key encoding:}\quad \boldsymbol{k}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_k,\\
&\text{attention:}\quad a_{i,j} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j^T}{\sqrt{C'}},\quad \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}},\\
&\text{value encoding:}\quad \boldsymbol{v}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_v,\\
&\text{weighted sum:}\quad \boldsymbol{f}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{v}_j.
\end{aligned}
$$

2. Transformer (Self-attention with Positional Embedding)

Most Transformers, including the original Transformer[1], employ position embedding to incorporate positional information. Specifically, they integrate absolute positions into element representations, formulated as follows:

$$
\begin{aligned}
&\text{w/ input position embedding:}\ \boldsymbol{f}_i = \boldsymbol{x}_i + \mathrm{Embed}(x_i, y_i),\\
&\text{query encoding:}\quad \boldsymbol{q}_i = \boldsymbol{f}_i \cdot \boldsymbol{W}_q,\\
&\text{key encoding:}\quad \boldsymbol{k}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_k,\\
&\text{attention:}\quad a_{i,j} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j^T}{\sqrt{C'}},\quad \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}},\\
&\text{value encoding:}\quad \boldsymbol{v}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_v,\\
&\text{weighted sum:}\quad \boldsymbol{f}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{v}_j.
\end{aligned}
$$

3. Relative Key Vector

Shaw et al.[14] enhanced Transformer for language modeling by adding learnable relative positional vectors into the key computations. BoTNet[21] and HaloNet[22] extended this approach to two dimensions for image processing by adding learnable relative positional vectors into the key computation. This can be formulated as follows,

doi.org/10.32388/Q6KCoC.2

$$
\begin{aligned}
&\text{w/o input position embedding:}\quad \boldsymbol{f}_i = \boldsymbol{x}_i,\\
&\text{query encoding:}\quad \boldsymbol{q}_i = \boldsymbol{f}_i \cdot \boldsymbol{W}_q,\\
&\text{key encoding:}\quad \boldsymbol{k}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_k + \boldsymbol{r}_{\delta_x,\delta_y},\\
&\text{attention:}\quad a_{i,j} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j^T}{\sqrt{C'}},\quad \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}},\\
&\text{value encoding:}\quad \boldsymbol{v}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_v,\\
&\text{weighted sum:}\quad \boldsymbol{f}_i' = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{v}_j,
\end{aligned}
$$

where $\mathbf{r}_{\delta_x,\delta_y} \in \mathbb{R}^{1\times C'}$.

4. Relative Value Vector

Shaw et al.[14] also extended the above relative vector method to the value computations, as follows:

$$
\begin{aligned}
&\text{w/o input position embedding:}\quad \boldsymbol{f}_i = \boldsymbol{x}_i,\\
&\text{query encoding:}\quad \boldsymbol{q}_i = \boldsymbol{f}_i \cdot \boldsymbol{W}_q,\\
&\text{key encoding:}\quad \boldsymbol{k}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_k,\\
&\text{self}-\text{attention:}\quad a_{i,j} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j^T}{\sqrt{C'}},\quad \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}},\\
&\text{value encoding:}\quad \boldsymbol{v}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_v + \boldsymbol{r}_{\delta_x,\delta_y},\\
&\text{weighted sum:}\quad \boldsymbol{f}_i' = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{v}_j.
\end{aligned}
$$

5. Relative Positional Scalar

Swin Transformer[12] and CoAtNet[20] incorporate a learnable relative positional bias (a scalar) into the attention score. In these methods, the original self-attention can be regarded as content attention, which measures relationships from the token-feature perspective, while the additional relative positional bias can be regarded as position attention, which measures relationships from the token-position perspective. Formally, this can be expressed as follows:

$$
\begin{aligned}
&\text{w/o input position embedding:}\ \ \boldsymbol{f}_i = \boldsymbol{x}_i, \\
&\text{query encoding:}\ \ \boldsymbol{q}_i = \boldsymbol{f}_i \cdot \boldsymbol{W}_q, \\
&\text{key encoding:}\ \ \boldsymbol{k}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_k, \\
&\text{attention:}\ \ a_{i,j} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j^T}{\sqrt{C'}} + b_{\delta_x,\delta_y},\ \ \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}}, \\
&\text{value encoding:}\ \ \boldsymbol{v}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_v, \\
&\text{weighted sum:}\ \ \boldsymbol{f}_i' = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{v}_j,
\end{aligned}
$$

where $b_{\delta_x,\delta_y} \in \mathbb{R}$. ConViT[23] introduces Gated Positional Self-Attention (GPSA), a variant of self-attention that incorporates a positional inductive bias. Moreover, a learnable gating parameter in each attention head controls the balance between positional and content-based attention, as follows,

$$
\begin{aligned}
&\text{w/o input position embedding:}\ \ \boldsymbol{f}_i = \boldsymbol{x}_i, \\
&\text{query encoding:}\ \ \boldsymbol{q}_i = \boldsymbol{f}_i \cdot \boldsymbol{W}_q, \\
&\text{key encoding:}\ \ \boldsymbol{k}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_k, \\
&\text{patch attention:}\ \ a_{i,j} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j^T}{\sqrt{C'}},\ \ \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}}, \\
&\text{position attention:}\ \ b_{i,j} = \boldsymbol{w} \cdot \boldsymbol{r}_{\|\delta\|},\ \ \beta_{i,j} = \frac{e^{b_{i,j}}}{\sum_{n=1}^{N} e^{b_{i,n}}}, \\
&\text{gated attention:}\ \ c_{i,j} = \big(1 - \sigma(\lambda)\big) \times \alpha_{i,j} + \sigma(\lambda) \times \beta_{i,j},\ \ \xi_{i,j} = \frac{c_{i,j}}{\sum_{n=1}^{N} c_{i,n}}, \\
&\text{value encoding:}\ \ \boldsymbol{v}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_v, \\
&\text{weighted sum:}\ \ \boldsymbol{f}_i' = \sum_{j=1}^{N} \xi_{i,j} \times \boldsymbol{v}_j,
\end{aligned}
$$

where $\mathbf{w}$ is a trainable vector for embedding, $\mathbf{r}_{\|\delta\|}$ is the relative positional encoding, λ is a learnable gate and σ is the Sigmoid function.

6. Rotary Position Embedding

Unlike the above vector- and scalar-based methods, RoFormer[24] proposes a rotation-based positional encoding method that is applied directly to queries and keys. As a result, attention scores depend solely on relative distances, eliminating the need to explicitly store a positional vector or scalar, as follows,

doi.org/10.32388/Q6KCoC.2

$$
\begin{aligned}
&\text{w/o input position embedding:}\quad \boldsymbol{f}_i = \boldsymbol{x}_i,\\
&\text{query encoding:}\quad \boldsymbol{q}_i = \boldsymbol{f}_i \cdot \boldsymbol{W}_q,\\
&\text{key encoding:}\quad \boldsymbol{k}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_k,\\
&\text{attention:}\quad \boldsymbol{q}'_i,\ \boldsymbol{k}'_j = \text{rotary}(\boldsymbol{q}_i,\ \boldsymbol{k}_j),\quad a_{i,j} = \frac{\boldsymbol{q}'_i \cdot \boldsymbol{k}'^{T}_j}{\sqrt{C'}},\quad \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}},\\
&\text{value encoding:}\quad \boldsymbol{v}_j = \boldsymbol{f}_j \cdot \boldsymbol{W}_v,\\
&\text{weighted sum:}\quad \boldsymbol{f}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{v}_j,
\end{aligned}
$$

where $rotary(\cdot)$ is a rotary position embedding function.

7. Relative Positional Matrix (Translution)

Inspired by convolution, we propose Translution that performs matrix multiplication to produce a vector output that encodes displacement or offset information, defined as follows:

$$
\begin{aligned}
&\text{w/o input position embedding:}\quad \boldsymbol{f}_i = \boldsymbol{x}_i,\\
&\text{relative query encoding:}\quad \boldsymbol{q}_{i,j} = \boldsymbol{f}_i \cdot \boldsymbol{W}^{q}_{\delta_x,\delta_y},\\
&\text{relative key encoding:}\quad \boldsymbol{k}_{j,i} = \boldsymbol{f}_j \cdot \boldsymbol{W}^{k}_{-\delta_x,-\delta_y},\\
&\text{relative attention:}\quad a_{i,j} = \frac{\boldsymbol{q}_{i,j} \cdot \boldsymbol{k}^{T}_{j,i}}{\sqrt{C'}},\quad \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}},\\
&\text{relative value encoding:}\quad \boldsymbol{v}_{i,j} = \boldsymbol{f}_j \cdot \boldsymbol{W}^{v}_{\delta_x,\delta_y},\\
&\text{weighted sum:}\quad \boldsymbol{f}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{v}_{i,j}.
\end{aligned}
$$

Table 8 provides a summary of various positional encoding strategies.

doi.org/10.32388/Q6KCoC.2

Method		
w/o Pos Emb	$\mathbf{f}_i = \mathbf{x}_i$	Baseline
w/ Pos Emb	$\mathbf{f}_i = \mathbf{x}_i + Embed(x_i, y_i)$	Transformer[1]
Relative Positional Vector	Key	Shaw et al.[14], BoTNet[21], HaloNet[22], *etc*
	Value	Shaw et al.[14]
Relative Positional Scalar	w/o gating	Swin Transformer[12], CoAtNet[20], *etc*
	w/ gating	ConViT[23]
Rotary Position Embedding	RoFormer[24]	
Relative Positional Matrix	α-Translution	
	Translution	

Table 8. Summary of different position encoding strategies.

Appendix F. Translution with Input Positional Embedding

In this section, we examine whether incorporating the input positional embedding method from Transformer can further improve Translution. To this end, we implement Translution as follows:

$$
\begin{aligned}
&\text{w/ input position embedding:}\quad \boldsymbol{f}_i = \boldsymbol{x}_i + \mathrm{Embed}(x_i, y_i),\\
&\text{relative query encoding:}\quad \boldsymbol{q}_{i,j} = \boldsymbol{f}_i \cdot \boldsymbol{W}^{q}_{\delta_x,\delta_y},\\
&\text{relative key encoding:}\quad \boldsymbol{k}_{j,i} = \boldsymbol{f}_j \cdot \boldsymbol{W}^{k}_{-\delta_x,-\delta_y},\\
&\text{relative attention:}\quad a_{i,j} = \frac{\boldsymbol{q}_{i,j} \cdot \boldsymbol{k}^T_{j,i}}{\sqrt{C'}},\quad \alpha_{i,j} = \frac{e^{a_{i,j}}}{\sum_{n=1}^{N} e^{a_{i,n}}},\\
&\text{relative value encoding:}\quad \boldsymbol{v}_{i,j} = \boldsymbol{f}_j \cdot \boldsymbol{W}^{v}_{\delta_x,\delta_y},\\
&\text{weighted sum:}\quad \boldsymbol{f}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{v}_{i,j}.
\end{aligned}
$$

doi.org/10.32388/Q6KCoC.2

As shown in Table 9, incorporating the Transformer's absolute positional embedding does not yield a clear performance gain for Translution in the static-to-static setting, leads to a slight drop in the dynamic-to-dynamic setting, and results in a substantial drop in the static-to-dynamic setting.

Method	$Embed\ (x_i,\ y_i)$	**#Parameters**	**Static→Static**	**Dynamic→Dynamic**	**Static→Dynamic**
α-Translution	✗	4.6 M	98.48	**97.31**	**34.90**
	✓	4.6 M	**98.72**	96.81	17.20
Translution	✗	116.2 M	**98.60**	**97.35**	**36.24**
	✓	116.2 M	98.47	96.31	16.50

Table 9. Accuracy (%) of Translution w/o and w/ the absolute positional embedding method from Transformer. Results are reported on Static and Dynamic MNIST with ViT-A/12.

Appendix G. Impact of W^q, W^k and W^v on α-Translution

Recall that: To reduce the number of parameters, we propose α-Translution, which decreases both the input dimension C^1 and the output dimension C^2 of each $\mathbf{W}^q_{\delta_x,\delta_y}$, $\mathbf{W}^k_{\delta_x,\delta_y}$, and $\mathbf{W}^v_{\delta_x,\delta_y}$. However, setting C^1 and C^2 too small can overly compress the query, key, and value representations, thereby degrading performance. To address this issue, we integrate the $\mathbf{W}^q$, $\mathbf{W}^k$, and $\mathbf{W}^v$ of Transformer into α-Translution to better preserve essential information.

In this section, we analyze the impact of $\mathbf{W}^q$, $\mathbf{W}^k$, and $\mathbf{W}^v$ by systematically removing them from Eq. (2) as follows:

$$
\alpha-\text{Translution}\begin{cases}
\text{query encoding: } \mathbf{q}_{i,j}=\mathbf{f}_i\cdot\mathbf{W}^q_1\cdot\mathbf{W}^q_{\delta_x,\delta_y},\mathbf{q}_i=\mathbf{f}_i\cdot\mathbf{W}^q,\\
\text{key encoding: } \mathbf{k}_{j,i}=\mathbf{f}_j\cdot\mathbf{W}^k_1\cdot\mathbf{W}^k_{-\delta_x,-\delta_y},\mathbf{k}_j=\mathbf{f}_j\cdot\mathbf{W}^k,\\
\text{self - attention: } a_{i,j}=\frac{\mathbf{q}_{i,j}\cdot\mathbf{k}^T_{j,i}+\mathbf{q}_i\cdot\mathbf{k}^T_j}{\sqrt{C'}},\alpha_{i,j}=\frac{e^{a_{i,j}}}{\sum_{n=1}^{N}e^{a_{i,n}}},\\
\text{value encoding: } \mathbf{v}_{i,j}=\mathbf{f}_j\cdot(\mathbf{W}^v_1\cdot\mathbf{W}^v_{\delta_x,\delta_y}\cdot\mathbf{W}^v_2+\mathbf{W}^v),\\
\text{weighted sum: } \mathbf{f}'_i=\sum_{j=1}^{N}\alpha_{i,j}\times\mathbf{v}_{i,j}.
\end{cases}
$$

As shown in Table 10, incorporating $\mathbf{W}^q$, $\mathbf{W}^k$, and $\mathbf{W}^v$ significantly enhances the performance of α-Translution, particularly when C^1 and C^2 are small. As C^1 and C^2 grow larger, the improvement decreases because the information is no longer overly compressed. In this case, $\mathbf{W}^q$, $\mathbf{W}^k$, and $\mathbf{W}^v$ become less critical.

Relative Encoding Dimension	$\mathbf{W}^q, \mathbf{W}^k, \mathbf{W}^v$	#Parameters	Top-1	Top-5
$C^1 = C^2 = 0$	✓	4.68 M	42.49	67.39
$C^1 = C^2 = 2$	✗	4.08 M	31.77	56.66
	✓	4.75 M	46.10	71.29
$C^1 = C^2 = 4$	✗	4.21 M	37.46	62.72
	✓	4.89 M	47.61	72.18
$C^1 = C^2 = 8$	✗	4.67 M	41.81	67.23
	✓	5.33 M	48.36	73.31
$C^1 = C^2 = 16$	✗	6.40 M	44.87	69.91
	✓	7.06 M	48.91	73.65
$C^1 = C^2 = 32$	✗	13.09 M	47.27	72.20
	✓	13.75 M	50.07	74.84

Table 10. Impact of $\mathbf{W}^q$, $\mathbf{W}^k$ and $\mathbf{W}^v$ on α-Transformer. Results are reported on ImageNet-1K with ViT-A/56, trained from scratch (no external pretraining) using a batch size of 256.

Appendix H. Relative CLS Token

For classification tasks, besides the image tokens, there is an additional CLS token (classification token) that serves as a global representation of the input image. Usually, the CLS token is a learnable embedding appended at the beginning of the input token sequence fed into Transformer. To apply the strategy of relative encoding to the CLS token, we introduce additional parameters: $\mathbf{W}^q_{CLS\_in}$, $\mathbf{W}^q_{CLS}$, $\mathbf{W}^q_{CLS\_out}$, $\mathbf{W}^k_{CLS\_in}$, $\mathbf{W}^k_{CLS}$, $\mathbf{W}^k_{CLS\_out}$, and $\mathbf{W}^v_{CLS\_in}$, $\mathbf{W}^v_{CLS}$, $\mathbf{W}^v_{CLS\_out}$, corresponding to the query, key, and value, respectively.

doi.org/10.32388/Q6KCoC.2

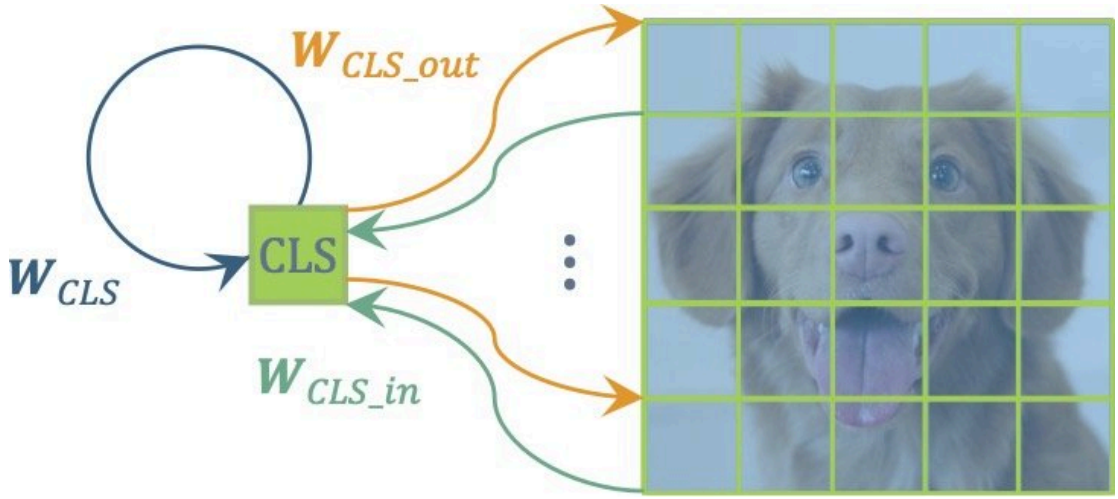

Figure 7. Illustration of relative encoding for the CLS token. For CLS, there are three encoding directions: in, in-place, and out. Correspondingly, three sets of weights, *i.e.*, $\mathbf{W}_{CLS\_in}$, $\mathbf{W}_{CLS}$, and $\mathbf{W}_{CLS\_out}$, are introduced for relative encoding in each respective direction.

As shown in Figure 7, $\mathbf{W}_{CLS\_in}$ is utilized when gathering information from the image tokens to update the CLS token; $\mathbf{W}_{CLS}$ is applied when updating the CLS token based on its own information; and $\mathbf{W}_{CLS\_out}$ is employed when image tokens gather information from the CLS token to update themselves.

Appendix I. Training Details

I.1. Dynamic MNIST

The training is conducted with an input image size of 84 for 10 epochs using a batch size of 24. Optimization employs the AdamW optimizer with an initial learning rate of 3×10^{-4} and a weight decay of 0.05. The learning rate is scheduled with CosineAnnealingLR over the training epochs. The input data is preprocessed using a transformation pipeline consisting of Normalize(0.1307,0.3081).

I.2. ImageNet

The model is trained over 300 epochs, using an input image size of 224 and a batch size of 256. Optimization is carried out using the AdamW optimizer, with an initial learning rate of 3×10^{-4} and a weight decay of 0.05. The learning rate follows a CosineAnnealingLR schedule, which gradually decreases it according to a smooth cosine curve throughout training to aid in stable convergence. Data

augmentation includes RandomResizedCrop (224), RandomHorizontalFlip, and RandAugment (with two random operations and magnitude 9), providing automated, robust augmentation policies that enhance generalization. The input data is preprocessed using a transformation pipeline consisting of Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]). Additionally, CutMix ($\alpha = 1.0$) and MixUp ($\alpha = 0.8$) are applied in a batch-level "random choice" manner for strong regularization—CutMix pastes patches between images while proportionally mixing labels; MixUp blends images and labels to improve generalization and training stability.

I.3. OpenWebText

The training runs for 9.6 million iterations with a mini-batch size of 8. Optimization is performed using the AdamW optimizer with an initial learning rate of 3×10^{-4} and a weight decay of 0.05. The learning rate is scheduled by CosineAnnealingLR over every 3.2k iterations. Gradient clipping is applied with a maximum norm of 1.0.

References

1. [a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v] *Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017). "Attention Is All You Need." Conference on Neural Information Processing Systems (NeurIPS). 5998–6008.*
2. [a, b, c, d] *Radford A, Narasimhan K, Salimans T, Sutskever I (2018). "Improving Language Understanding by Generative Pre-Training." OpenAI.*
3. [a, b] *Devlin J, Chang M, Lee K, Toutanova K (2019). "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). 4171–4186.*
4. [a, b, c, d, e] *Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N (2021). "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale." International Conference on Learning Representations (ICLR).*
5. [a, b] *LeCun Y, Bottou L, Bengio Y, Haffner P (1998). "Gradient-Based Learning Applied to Document Recognition." Proc IEEE. **86**(11):2278–2324. doi:10.1109/5.726791.*
6. [a, b] *Krizhevsky A, Sutskever I, Hinton GE (2012). "ImageNet Classification With Deep Convolutional Neural Networks." Conference on Neural Information Processing Systems (NeurIPS). 1106–1114.*

7. a, b *Simonyan K, Zisserman A (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognitio n." International Conference on Learning Representations (ICLR).*
8. a, b *Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015). "Go ing Deeper With Convolutions." IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1–9.*
9. a, b *He K, Zhang X, Ren S, Sun J (2016). "Deep Residual Learning for Image Recognition." IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 770–778.*
10. a, b, c *Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019). "Language Models Are Unsupervised M ultitask Learners." OpenAI blog. **1**(8):9.*
11. a, b *Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amod ei D (2020). "Language Models Are Few-Shot Learners." Conference on Neural Information Processing Syst ems (NeurIPS).*
12. a, b, c, d, e, f *Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021). "Swin Transformer: Hierarchical Visi on Transformer Using Shifted Windows." IEEE International Conference on Computer Vision (ICCV). 9992–10002.*
13. ^ *Touvron H, Cord M, Douze M, Massa F, Sablayrolles A, Jégou H (2021). "Training Data-Efficient Image Tra nsformers & Distillation Through Attention." International Conference on Machine Learning (ICML). **139**:10 347–10357.*
14. a, b, c, d, e, f, g, h, i *Shaw P, Uszkoreit J, Vaswani A (2018). "Self-Attention With Relative Position Representation s." Conference of the North American Chapter of the Association for Computational Linguistics: Human Lan guage Technologies (NAACL-HLT). 464–468.*
15. ^ *Huang CA, Vaswani A, Uszkoreit J, Simon I, Hawthorne C, Shazeer N, Dai AM, Hoffman MD, Dinculescu M, Eck D (2019). "Music Transformer: Generating Music With Long-Term Structure." International Conference on Learning Representations (ICLR).*
16. ^ *Parmar N, Ramachandran P, Vaswani A, Bello I, Levskaya A, Shlens J (2019). "Stand-Alone Self-Attention i n Vision Models." Conference on Neural Information Processing Systems (NeurIPS). 68–80.*
17. ^ *Dai Z, Yang Z, Yang Y, Carbonell JG, Le QV, Salakhutdinov R (2019). "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context." Conference of the Association for Computational Linguistics (AC L). 2978–2988.*

18. ^Tsai YH, Bai S, Yamada M, Morency L, Salakhutdinov R (2019). "Transformer Dissection: An Unified Understanding for Transformer's Attention Via the Lens of Kernel." Conference on Empirical Methods in Natural Language Processing (EMNLP). 4343–4352.

19. ^Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu PJ (2020). "Exploring the Limits of Transfer Learning With a Unified Text-to-Text Transformer." J Mach Learn Res. **21**:140:1–140:67.

20. a, b, c, dDai Z, Liu H, Le QV, Tan M (2021). "CoAtNet: Marrying Convolution and Attention for All Data Sizes." Conference on Neural Information Processing Systems (NeurIPS). 3965–3977.

21. a, b, cSrinivas A, Lin T, Parmar N, Shlens J, Abbeel P, Vaswani A (2021). "Bottleneck Transformers for Visual Recognition." IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 16519–16529.

22. a, b, cVaswani A, Ramachandran P, Srinivas A, Parmar N, Hechtman BA, Shlens J (2021). "Scaling Local Self-Attention for Parameter Efficient Visual Backbones." IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 12894–12904.

23. a, b, c, d, ed'Ascoli S, Touvron H, Leavitt ML, Morcos AS, Biroli G, Sagun L (2021). "ConViT: Improving Vision Transformers With Soft Convolutional Inductive Biases." International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research. **139**:2286–2296.

24. a, b, c, dSu J, Ahmed MH, Lu Y, Pan S, Bo W, Liu Y (2024). "RoFormer: Enhanced Transformer With Rotary Position Embedding." Neurocomputing. **568**:127063. doi:10.1016/J.NEUCOM.2023.127063.

25. ^Gulati A, Qin J, Chiu C, Parmar N, Zhang Y, Yu J, Han W, Wang S, Zhang Z, Wu Y, Pang R (2020). "Conformer: Convolution-Augmented Transformer for Speech Recognition." Interspeech. 5036–5040.

26. ^Yuan K, Guo S, Liu Z, Zhou A, Yu F, Wu W (2021). "Incorporating Convolution Designs Into Visual Transformers." IEEE International Conference on Computer Vision (ICCV). 559–568.

27. ^Srivastava N, Mansimov E, Salakhutdinov R (2015). "Unsupervised Learning of Video Representations Using LSTMs." International Conference on Machine Learning (ICML). **37**:843–852.

28. ^Fan H, Yang Y (2019). "PointRNN: Point Recurrent Neural Network for Moving Point Cloud Processing." arXiv. 1910.08287.

29. ^Deng J, Dong W, Socher R, Li L, Li K, Fei-Fei L (2009). "ImageNet: A Large-Scale Hierarchical Image Database." IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 248–255.

30. ^Gao L, Biderman S, Black S, Golding L, Hoppe T, Foster C, Phang J, He H, Thite A, Nabeshima N, Presser S, Leahy C (2020). "The Pile: An 800GB Dataset of Diverse Text for Language Modeling." arXiv. 2101.00027.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.