Article

Translution: Unifying Transformer and Convolution for Adaptive and Relative Modeling

Hehe Fan [*] , Yi Yang , Fei Wu

Posted Date: 8 May 2025

doi: 10.20944/preprints202505.0597.v1

Keywords: Deep neural network; Transformer; Convolution

Article

Translution: Unifying Transformer and Convolution for Adaptive and Relative Modeling

Hehe Fan *, Yi Yang and Fei Wu

College of Computer Science and Technology, Zhejiang University
* Correspondence: hehefan@zju.edu.cn

Abstract: When referring to modeling, we consider it to involve two steps: 1) identifying relevant data elements or regions and 2) encoding them effectively. Transformer, leveraging self-attention, can adaptively identify these elements or regions but rely on absolute position encoding for their representation. In contrast, Convolution encodes elements or regions in a relative manner, yet their fixed kernel size limits their ability to adaptively select the relevant regions. We introduce Translution, a new neural network module that unifies the adaptive identification capability of Transformer and the relative encoding advantage of Convolution. However, this integration results in a substantial increase in parameters and memory consumption, exceeding our available computational resources. Therefore, we evaluate Translution on small-scale datasets, i.e., MNIST and CIFAR. Experiments demonstrate that Translution achieves higher accuracy than Transformer. We encourage the community to further evaluate Translution using larger-scale datasets across more diverse scenarios and to develop optimized variants for broader applicability.

Keywords: deep neural network; transformer; convolution

1. Introduction

When employing deep neural networks to model a specific type of data (e.g., images or text), it can be decoupled into two steps: 1) identifying relevant data elements (e.g., patches in images or words in text) or regions, and 2) encoding these elements into effective representations.

In Convolution [1–5], as shown in Figure 1a, the relevant region identification step is handled by convolutional filters with a fixed local receptive field (kernel). This fixed kernel defines a neighborhood of elements considered relevant. For visual data like images, such local focus is often effective because spatially adjacent pixels or patches tend to be related (e.g., forming parts of the same object). However, the rigid nature of a fixed-size kernel makes Convolution inevitably covers some irrelevant pixels or regions — especially near object boundaries or in background areas that fall inside the window. In contrast, as shown in Figure 1b, Transformer [6–10] uses a self-attention mechanism to adaptively identify relevant regions. Instead of being limited to a predetermined locality, self-attention allows the model to dynamically attend to relevant regions. This means a Transformer can focus on important features regardless of their distance, potentially ignoring irrelevant context more flexibly than a Convolution's fixed receptive field, which makes it highly suitable for processing long texts.

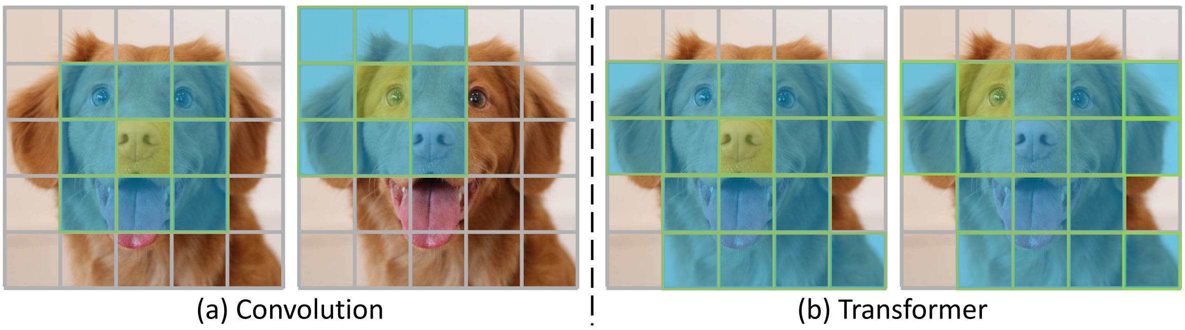


Figure 1. Comparison of Convolution and Transformer in identifying relevant regions (blue patches) for the kernel center or query area (yellow patch). 1) Convolution utilizes a fixed kernel size to define a neighborhood of regions considered relevant, inadvertently including some irrelevant patches or regions, particularly near object boundaries or within background areas inside the window. 2) Transformer employs a self-attention mechanism to adaptively and flexibly identify relevant regions, thereby avoiding including noisy or irrelevant areas.

When it comes to encoding the structure from these relevant regions, Convolution and Transformer employ different strategies. As shown in Figure 2a, a convolutional kernel learns distinct weights for each relative position within its receptive field. In other words, the filter has separate parameters for each offset (direction and distance) from the center. This design enables Convolution to encode local structure relatively — capturing orientation and distance relationships. On the other hand, as shown in Figure 2b, Transformer uses a shared set of weights to process inputs from all positions. The same value-projection weights are applied universally across positions. Consequently, the Value of Transformer does not encode whether one patch is to the left or right of another. To introduce positional information, Transformer incorporates absolute positional embeddings (either fixed sinusoidal or learned positional vectors) 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. Note that as the dataset scale increases, Transformer will become effective because the data might include scenarios where the object appears in different locations.

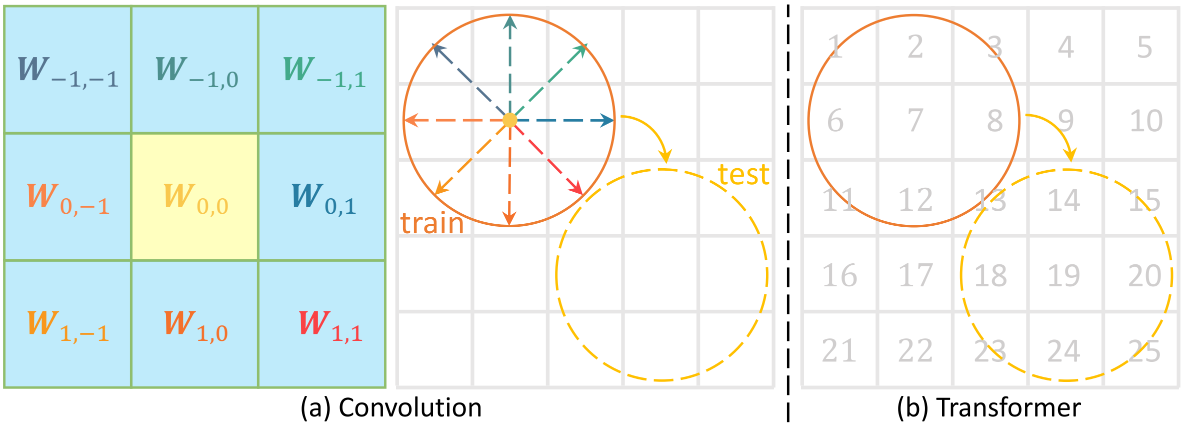


Figure 2. Comparison of Convolution and Transformer in encoding relevant regions: consider the scenario where Convolution and Transformer are tasked with recognizing a circle. 1) Convolution learns separate parameters for each offset (direction and distance) from the kernel center, allowing it to effectively encode relative local structures. Thus, when the circle appears in a different location, it is still readily recognized due to this relative awareness. 2) Transformer incorporates absolute position information into each token's representation and uses a shared set of weights across all tokens for computing Value. While this method facilitates general processing, the inclusion of absolute position embeddings makes it more challenging to recognize the circle when it is moved to a different location.

In summary, Convolution encodes structure through fixed local filters with position-specific weights, whereas Transformer relies on adaptive global attention and requires absolute positional encoding to capture order or spatial patterns. In this article, we introduce Translution, a new neural network module that combines the adaptive identification capabilities of the Transformer with the relative encoding advantages of Convolution. Translution retains the self-attention mechanism of Transformer but utilizes separate parameters to compute Value. This design enables Translution to effectively encode relative structures. However, this integration leads to a significant increase in the number of parameters and memory consumption. Specifically, for an input with N tokens, Transformer requires only 1 set of shared parameters to calculate Value, whereas Translution consumes $(2N-1)^2$ sets of parameters. This exceeds our available computational resources for high-resolution datasets. Consequently, we restricted our evaluation of Translution to low-resolution datasets, specifically MNIST and CIFAR. Experiments demonstrate that Translution outperforms Transformer in terms of accuracy. However, these experiments are not comprehensive. We encourage the community to further evaluate Translution on larger-scale datasets in a variety of scenarios and to develop optimized versions for wider applicability.

2. Related Work

Transformer [6] eschews recurrence and kernel size, relying on self-attention for relevant regin identification. Because it has no built-in notion of order, Transformer layers add explicit positional encodings to token embeddings so the model can use sequence order. Transformer employs sinusoidal (fixed) position vectors and note that learned embeddings produce nearly identical performance, with the fixed sinusoidal encodings chosen because they may allow better extrapolation to longer sequences. Subsequent work has explored "relative attention" [11–17]. For example, Shaw *et al*.enhanced Transformer by adding learnable relative positional vectors into the key and value computation for language modeling [11]. Bello *et al*.expanded this approach to two dimensions for image processing [18]. HaloNet employs learnable relative positional vectors to modify the query, making it aware of relative positions [19]. CoAtNet [17] adds a learnable scalar for each direction and distance into attention. ConViT also incorporates relative position embedding into the attention mechanism, employing a more complex strategy [20]. Unlike these existing relative attention methods, Translution divides the modeling process into two distinct phases: relevant region identification and relative structure encoding. In the first phase, Translution utilizes attention to only identify related regions. In the second phase, it employs a convolution-style method that uses separate weights (matrices) for each relative position within its receptive field to encode the relative structure.

Convolutional neural networks (CNNs) [1,2] have been the backbone of vision and sequence models for years. By using small, shared kernels and pooling, CNNs efficiently capture local spatial or temporal patterns with translation-invariant filters. Recent architectural developments increasingly seek to integrate self-attention with convolution. A prevalent strategy, particularly in vision applications, involves combining convolutional modules and Transformer-like blocks. For instance, BoTNet substitutes the spatial convolutions with global self-attention in the final three bottleneck blocks of ResNet [18], while CeiT combines convolutions for extracting low-level features with Transformers to manage long-range dependencies [21]. In contrast to these approaches, Translution functions at the module or layer level, blending the the advantages of Transformer and Convolution into a fundamental and unified operation.

3. Proposed Method

3.1. Convolution

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

$$\boldsymbol{F}'_{x,y} = \sum_{\delta_x=-\lfloor h/2\rfloor}^{\lfloor h/2\rfloor} \sum_{\delta_y=-\lfloor w/2\rfloor}^{\lfloor w/2\rfloor} \boldsymbol{F}_{x+\delta_x,y+\delta_y} \cdot \boldsymbol{W}_{\delta_x,\delta_y}, \tag{1}$$

where $\boldsymbol{W}_{\delta_x,\delta_y} \in \mathbb{R}^{C\times C'}$ represents the learnable weights for displacement (δ_x, δ_y), C' represents the output feature dimension, and $\cdot$ is matrix multiplication.

By assigning a set of weights 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, 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. Transformer

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

$$\boldsymbol{F}_i = \boldsymbol{X}_i + \mathrm{Embed}(i), \tag{2}$$

and then performs two separate linear projections on the feature map to generate queries $\boldsymbol{Q} \in \mathbb{R}^{N\times C''}$ and keys $\boldsymbol{K} \in \mathbb{R}^{N\times C''}$, where C'' is the dimension for queries or keys. Subsequently, scaled dot-product attention $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ (where $N = H \times W$) is computed for each query, and a softmax function is applied to normalize the attention weights for a query across all tokens,

$$\boldsymbol{Q} = \boldsymbol{F} \cdot \boldsymbol{W}_q, \quad \boldsymbol{K} = \boldsymbol{F} \cdot \boldsymbol{W}_k, \quad \boldsymbol{A} = \frac{\boldsymbol{Q} \cdot \boldsymbol{K}^T}{\sqrt{C''}}, \quad \alpha_{i,j} = \frac{e^{A_{i,j}}}{\sum_{n=1}^{N} e^{A_{i,n}}}, \tag{3}$$

where $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathbb{R}^{C\times C''}$. Next, Transformer conducts another feature encoding on the input feature map to generate values $\boldsymbol{V} \in \mathbb{R}^{N\times C'}$, where C' is the value dimension. Finally, the output is computed as a weighted sum of the values,

$$\boldsymbol{V} = \boldsymbol{F} \cdot \boldsymbol{W}_v, \quad \boldsymbol{F}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{V}_j, \tag{4}$$

where $\boldsymbol{W}_v \in \mathbb{R}^{C\times C'}$ and $\boldsymbol{F} \in \mathbb{R}^{N\times C'}$. With self-attention, Transformer 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, Transformer does not encode the structure in a relative manner.

3.3. Translution

Translution is designed to integrate the adaptive relative region identification capabilities of Transformer with the relative encoding strengths of Convolution. Specifically, Translution maintains the self-attention mechanism of Transformer but employs distinct parameters to compute Value,

$$\begin{aligned}
&\mathrm{self{-}attention}\colon \boldsymbol{Q} = \boldsymbol{F} \cdot \boldsymbol{W}_q, \quad \boldsymbol{K} = \boldsymbol{F} \cdot \boldsymbol{W}_k, \quad \boldsymbol{A} = \frac{\boldsymbol{Q} \cdot \boldsymbol{K}^T}{\sqrt{C'}}, \quad \alpha_{i,j} = \frac{e^{A_{i,j}}}{\sum_{n=1}^{N} e^{A_{i,n}}}, \\
&\mathrm{relative\ \ encoding}\colon \boldsymbol{V}_{i,j} = \boldsymbol{F}_j \cdot \boldsymbol{W}_{\delta_x,\delta_y}, \quad \mathrm{where} \quad \delta_x = x_i - x_j, \quad \delta_y = y_i - y_j, \\
&\mathrm{weighted\ \ sum}\colon \boldsymbol{F}'_i = \sum_{j=1}^{N} \alpha_{i,j} \times \boldsymbol{V}_{i,j},
\end{aligned} \tag{5}$$

where $W_{\delta_x,\delta_y} \in \mathbb{R}^{C\times C'}$ denotes the learnable weights for displacement (δ_x, δ_y).

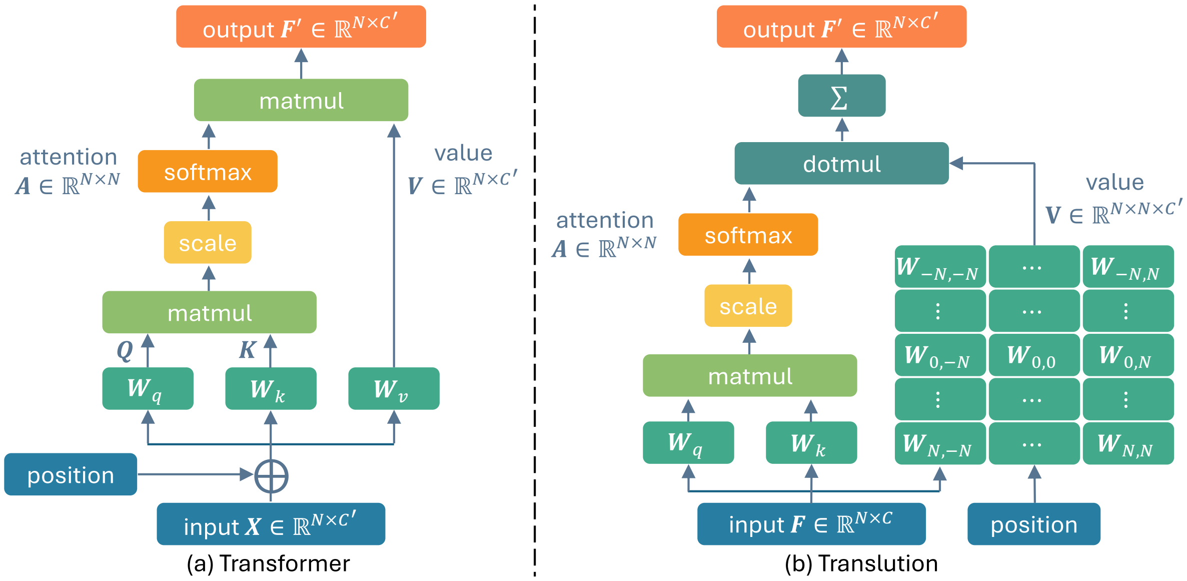

output $F' \in \mathbb{R}^{N\times C'}$
matmul
attention
$A \in \mathbb{R}^{N\times N}$
softmax
value
$V \in \mathbb{R}^{N\times C'}$
scale
matmul
Q
K
W_q
W_k
W_v
position
input $X \in \mathbb{R}^{N\times C'}$
(a) Transformer
output $F' \in \mathbb{R}^{N\times C'}$
$\sum$
dotmul
value
$V \in \mathbb{R}^{N\times N\times C'}$
attention
$A \in \mathbb{R}^{N\times N}$
softmax
$W_{-N,-N}$
$W_{-N,N}$
scale
$W_{0,-N}$
$W_{0,0}$
$W_{0,N}$
matmul
W_q
W_k
$W_{N,-N}$
$W_{N,N}$
input $F \in \mathbb{R}^{N\times C}$
position
(b) Translution

Figure 3. Comparison of Transformer and Translution in computing Value. 1) Transformer uses a shared set of weights, i.e., W_v, across all patches for computing Value. 2) Translution employs separate parameters for each offset (direction and distance), i.e., $\{W_{-N,-N}, \cdots, W_{0,0}, \cdots, W_{N,N}\}$, to encode relative structures. Suppose there are N tokens, Translution consumes $(2N-1)^2$ more parameters than Transformer. This may be one of the major issues with Translution. However, with the upgrading of computational resources and the increase in GPU memory, this will not be a problem in the near future.

4. Discussion

4.1. Translution unifies Transformer and Convolution.

The fixed receptive field in Convolution can be viewed as a form of attention, where the weight is set to 1 within the receptive field and 0 elsewhere. The weights W_v for computing Value in Transformer act as a shared linear projection across all directions and distances. Consequently, Translution represents an operation that integrates the functionalities of Convolution and Transformer, as follows,

$$\begin{aligned}
&\text{Convolution:} \quad && F'_i = \textstyle\sum_{j=1}^{N} \alpha_{i,j} \times F_j \cdot W_{\delta_x,\delta_y}, \quad && \text{where} \quad \alpha_{i,j} = \begin{cases} 1, & (\delta_i, \delta_j) \in \text{kernel}, \\ 0, & \text{otherwise.} \end{cases} \\
&\text{Transformer:} \quad && F'_i = \textstyle\sum_{j=1}^{N} \alpha_{i,j} \times F_j \cdot W_v, \quad && \text{where} \quad \alpha_{i,j} = \frac{e^{A_{i,j}}}{\sum_{n=1}^{N} e^{A_{i,n}}}. \\
&\text{Translution:} \quad && F'_i = \textstyle\sum_{j=1}^{N} \alpha_{i,j} \times F_j \cdot W_{\delta_x,\delta_y}, \quad && \text{where} \quad \alpha_{i,j} = \frac{e^{A_{i,j}}}{\sum_{n=1}^{N} e^{A_{i,n}}}.
\end{aligned}$$

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

4.2. Why not integrate relative encoding into self-attention rather than into the computation of Value?

The attention weight $\alpha_{i,j}$ between two tokens is a scalar ranging from 0 to 1. As a scalar, it cannot convey extensive information such as displacement. However, being a scalar within the (0,1) range, it effectively reflects relationships. If two tokens are highly related, the attention weight tends towards 1. Conversely, if two tokens are unrelated, the attention weight approaches 0. In contrast, the computation of Value involves vectors, which can encapsulate sufficient information to represent direction and distance. This aligns with the motivation to decouple modeling into identification and encoding of relevant regions. Here, self-attention is responsible for identifying relevant regions, while the computation of Value handles the encoding of these regions.

4.3. General Translution: a more general version of Translution.

The calculation of the Value in Translution, i.e., Eq. (5), assumes that positions in the data (e.g., images or text) are discrete. In this setting, it is feasible to assign a different set of weights 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 of relevant regions.

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

$$\text{General Translution:}\ \boldsymbol{F}'_i = \sum_{j=1}^{L} \alpha_{i,j} \times f(\boldsymbol{p}_i - \boldsymbol{p}_j, \boldsymbol{F}_j), \tag{6}$$

where $\alpha_{i,j} \in [0,1]$ denotes the attention weight measuring the relevance of the j-th token to the i-th token, and $f : \mathbb{R}^{d+C} \to \mathbb{R}^{C'}$ is a function that encodes relative positional information into the token 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 an effective attention mechanism α and structure encoding function f. Point Spatio-Temporal Transformer [22] can be viewed as a specific instance of general Translution, which has been demonstrated to be effective for 3D point cloud video classification tasks, particularly on small-scale datasets.

5. Experiment

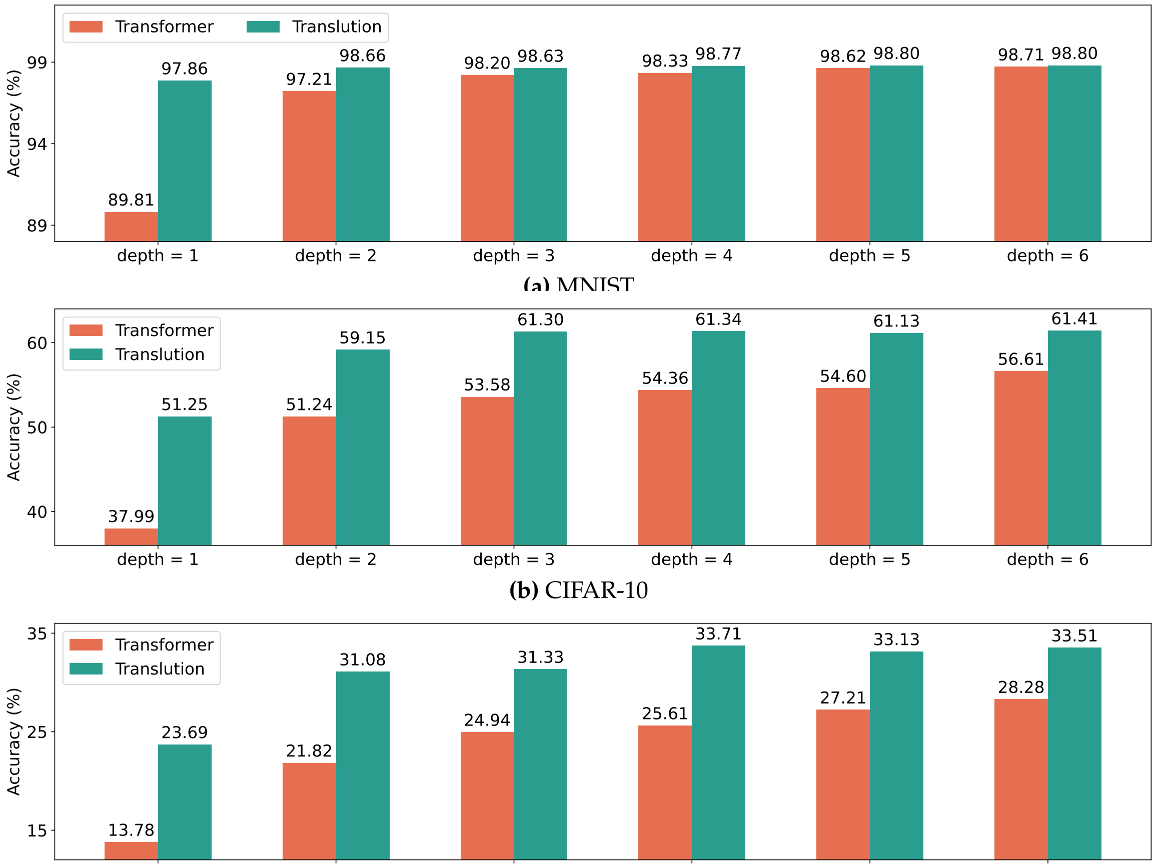
We compare Translution and Transformer using the same Vision Transformer (ViT)[1] [9] architecture. For our method, we substitute the Transformer component in ViT with Translution, while maintaining the remaining architecture unchanged.

5.1. MNIST, CIFAR-10 and CIFAR-100

For MNIST images of size 28 × 28, we set the patch size to 7, resulting in $\frac{28}{7} \times \frac{28}{7} + 1 = 17$ input tokens (the '1' represents the additional CLS token). For CIFAR images of size 32 × 32, we set the patch size to 8, also yielding $\frac{32}{8} \times \frac{32}{8} + 1 = 17$ input tokens. The dimension of the embedding vectors is set to 64, the MLP hidden layer dimension is set to 128, the number of head is set to 1 and the head dimension is 64. All training starts from scratch.

As shown in Figure 4, Translution outperforms Transformer in terms of accuracy. Moreover, the advantage is more pronounced in shallower networks, e.g., at a depth of 1. As the network depth increases, the differences diminish. This may indicate that as the network becomes deeper, the influence of absolute positioning embedding decreases because it is only used once at the initial input stage.

[1] https://github.com/lucidrains/vit-pytorch



(a) MNIST

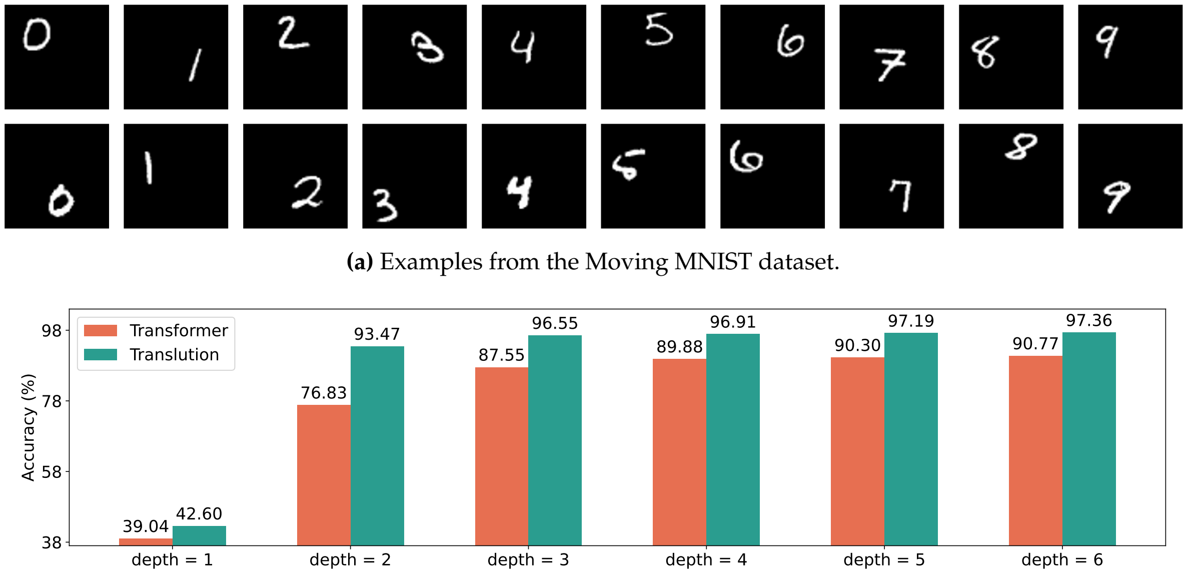
(b) CIFAR-10

(c) CIFAR-100

Figure 4. Accuracy on MNIST, CIFAR-10, and CIFAR-100 with different depths, respectively.

5.2. Moving MNIST

The location of digits in MNIST and objects in CIFAR is typically centered within the images. To assess the capability of modeling relative structures, we synthesize a Moving MNIST dataset [23,24] featuring MNIST digits that move within a 64×64 area, as shown in Figure 5a. We set the patch size to 8, resulting in $\frac{64}{8} \times \frac{64}{8} + 1 = 65$ input tokens.



(a) Examples from the Moving MNIST dataset.

(b) Accuracy on Moving MNIST with different depths.

Figure 5. Moving MNIST.

As shown in Figure 5b, the relative modeling of Translution effectively improves accuracy on the Moving MNIST dataset. Furthermore, comparing Figure 4a with Figure 5b, it is evident that changes in absolute location in Moving MNIST significantly affect Transformer. In contrast, when the depth of the network increases, it does not significantly impact Translution.

6. Conclusion

In this article, we propose Translution, a novel neural network module that integrates the adaptive identification capabilities of Transformer with the advantageous relative encoding properties of Convolution. Preliminary experimental results demonstrate the effectiveness of the method. Nevertheless, further validation using larger-scale datasets across diverse scenarios is required to comprehensively assess its performance. Additionally, the increasing number of parameters involved in the relative Value computation poses potential learning challenges, suggesting that optimized variants should be explored in future work.

References

1. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. https://doi.org/10.1109/5.726791.
2. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS), 2012, pp. 1106–1114.
3. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations (ICLR), 2015.
4. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.
5. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
6. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS), 2017, pp. 5998–6008.
7. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.
8. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 2019, pp. 4171–4186.
9. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In Proceedings of the International Conference on Learning Representations (ICLR), 2021.
10. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2021, pp. 9992–10002.
11. Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-Attention with Relative Position Representations. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT); Walker, M.A.; Ji, H.; Stent, A., Eds., 2018, pp. 464–468.
12. Huang, C.A.; Vaswani, A.; Uszkoreit, J.; Simon, I.; Hawthorne, C.; Shazeer, N.; Dai, A.M.; Hoffman, M.D.; Dinculescu, M.; Eck, D. Music Transformer: Generating Music with Long-Term Structure. In Proceedings of the International Conference on Learning Representations (ICLR), 2019.
13. Parmar, N.; Ramachandran, P.; Vaswani, A.; Bello, I.; Levskaya, A.; Shlens, J. Stand-Alone Self-Attention in Vision Models. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS), 2019, pp. 68–80.
14. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.G.; Le, Q.V.; Salakhutdinov, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In Proceedings of the Conference of the Association for Computational Linguistics (ACL), 2019, pp. 2978–2988.

15. Tsai, Y.H.; Bai, S.; Yamada, M.; Morency, L.; Salakhutdinov, R. Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019, pp. 4343–4352.
16. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 140:1–140:67.
17. Dai, Z.; Liu, H.; Le, Q.V.; Tan, M. CoAtNet: Marrying Convolution and Attention for All Data Sizes. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS), 2021, pp. 3965–3977.
18. Srinivas, A.; Lin, T.; Parmar, N.; Shlens, J.; Abbeel, P.; Vaswani, A. Bottleneck Transformers for Visual Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 16519–16529.
19. Vaswani, A.; Ramachandran, P.; Srinivas, A.; Parmar, N.; Hechtman, B.A.; Shlens, J. Scaling Local Self-Attention for Parameter Efficient Visual Backbones. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 12894–12904.
20. d'Ascoli, S.; Touvron, H.; Leavitt, M.L.; Morcos, A.S.; Biroli, G.; Sagun, L. ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases. In Proceedings of the International Conference on Machine Learning (ICML), 2021, Vol. 139, *Proceedings of Machine Learning Research*, pp. 2286–2296.
21. Yuan, K.; Guo, S.; Liu, Z.; Zhou, A.; Yu, F.; Wu, W. Incorporating Convolution Designs into Visual Transformers. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2021, pp. 559–568.
22. Fan, H.; Yang, Y.; Kankanhalli, M.S. Point Spatio-Temporal Transformer Networks for Point Cloud Video Modeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 2181–2192. https://doi.org/10.1109/TPAMI.2022.3161735.
23. Srivastava, N.; Mansimov, E.; Salakhutdinov, R. Unsupervised Learning of Video Representations using LSTMs. In Proceedings of the International Conference on Machine Learning (ICML), 2015, Vol. 37, pp. 843–852.
24. Fan, H.; Yang, Y. PointRNN: Point Recurrent Neural Network for Moving Point Cloud Processing. *arXiv* **2019**, *1910.08287*.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.