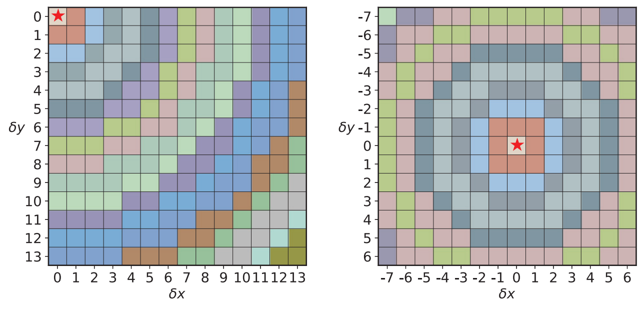
Rethinking and Improving Relative Position Encoding for Vision Transformer —— Supplementary Material ——

This supplementary material presents additional details of Section 3.2, 4.2, 4.3 and 4.4. Besides, two extra experiments are added to demonstrate the effectiveness and generality of the proposed iRPE. We also provide comparisons on the inference time.

- **Visualization of 2D relative position.** To provide an intuitive understanding, we visualize the proposed 2D relative position in Section 3.2, including Euclidean, Quantization, Cross and Product methods.
- **Weight initialization.** We elaborate the weight setting of the proposed relative position encoding methods, including the weight initialization and whether to equip with weight decay.
- **Computation complexity.** We provide a detailed explanation of why the computational costs are the same for shared and unshared relative position encodings across attention heads in Tab. 2 of Section 4.2.
- **Injecting previous RPE methods into DeiT.** We elaborate how to inject previous relative position encoding methods into DeiT [11] in Tab. 5 of Section 4.3.
- **Training and test settings of DETR.** We provide the details of training and test settings of DETR [1] in Section 4.4.
- **The effectiveness on other vision transformers.** We show the effectiveness of the proposed iRPE on the recent Swin transformer [7].
- **Transfer learning on fine-grained datasets.** To verify the generalizability, we evaluate our models on fine-grained datasets, including Stanford Cars and CUB200_2011 datasets.
- **Inference performance.** We compare the proposed iRPE with previous methods in terms of inference time and memory cost.

1. Visualization of 2D Relative Position

We visualize the proposed four relative position methods, *i.e.*, Euclidean, Quantization, Cross and Product, and present their difference. In DeiT [11], an image is split into 14×14 non-overlapping patches, so the number of tokens is 14×14 (except for the classification token). Therefore, in theory, each token has 14×14 relative positions. For visualization, we select the top-left position $(0, 0)$ and the center position $(7, 7)$ as the reference positions (presented by a red star ★ in the following figures), and then compute the relative offsets $\delta x = x_i - x_j$ and $\delta y = y_i - y_j$ between the reference position and the remaining $14 \times 14 - 1$ positions. Let $(\delta x, \delta y)$ denote a 2D relative position. We plot the map of the relative encoding $\mathbf{r}_{ij}$, defined in Eq. (19), Eq. (21), Eq. (22) and Eq. (25), where i is the reference position and j is one of the 14×14 positions. Notice that $\mathbf{r}_{ij}$ is either a learnable scalar in bias mode or a vector in contextual mode. Multiple $\mathbf{r}_{ij}$ may share an identical bucket, which is presented by the same color in Fig. 6 - 9. Different bucket is presented by different color.



(a) top-left (b) center

Figure 6: Visualization of Euclidean method. The red star ★ presents the reference position. Different color means different bucket. The relative positions with the same color share the same encoding.

Euclidean method. Fig. 6 shows Euclidean method. It is an undirected method, since the relative position encodings only depend on relative Euclidean distance. For example, in Fig. 6b since the relative positions $(-1, 0)$ and $(1, 0)$ have the same relative Euclidean distance of 1, they are mapped into the same bucket (the grids with orange color).

Quantization method. Fig. 7 presents Quantization method, another undirected method. It is an improved version of Euclidean method, and addresses the problem that the close two neighbors with different relative distances might be mapped into the same bucket (*e.g.*, he relative position $(1, 0)$ and $(1, 1)$ are both mapped into the same bucket in Euclidean method). Besides, the number of buckets in Quantization method is larger than that in Euclidean

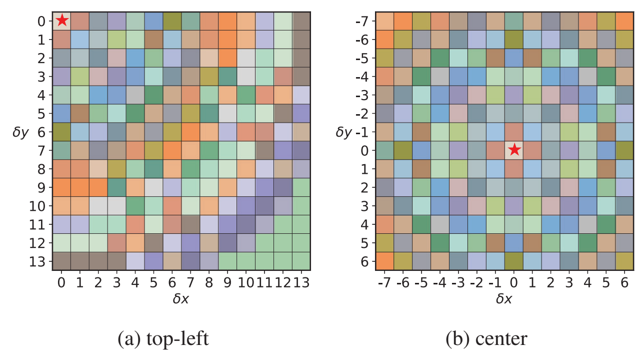

(a) top-left (b) center

Figure 7: Visualization of Quantization method. The red star ★ presents the reference position. Different color means different bucket. The relative positions with the same color share the same encoding.

method. The reason is that Quantization method quantize Euclidean distance from a set of real numbers {0, 1, 1.41, 2, 2.24, ...} to a set of integers {0, 1, 2, 3, 4, ...}, increasing the number of buckets for adjacent positions.

Cross method. Fig. 8 shows Cross method. It is a directed method, in which the relative position encoding depends on relative distances and relative directions simultaneously. It computes the encodings on horizontal and vertical directions separately, then summarizes them. The same offsets along x-axis (or y-axis) direction share the same horizontal (or vertical) encoding. For example, the two relative positions $(-1, 0)$ and $(1, 0)$ share the same encoding on horizontal in Fig. 8c, but not on vertical in Fig. 8d.

Product method. Fig. 9 shows Product method, which is also a directed method. Unlike Cross method, Product method does not share the same encoding even if the offsets are the same along x-axis or y-axis direction. For example, in Fig. 9b, the two relative positions $(-1, 0)$ and $(1, 0)$ have independent encodings. Moreover, it is more efficient than Cross method, since there is no extra addition operation in Eq. (22).

2. Weight Initialization

The relative position weight $\mathbf{r}_{ij}$ in Eq. (14) and Eq. (15) is initialized with zero. We found that there is no difference between zero and normal-distribution initialization. Besides, we do not impose weight decay on the weight of relative position encodings, because its effects on the final performance is negligible.

3. Computation Complexity

As shown in Tab. 2 (in the main manuscript), the computational costs MACs of shared and unshared relative position encodings across attention heads are the same. Here, we provide the detailed explanation. Let h, n, d, k denote the number of heads, the length of a sequence, the number of channels and the number of buckets, respectively. For bias mode, in Eq. (13), the broadcast addition on the dot-product attention $(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^T$ with the shape of $h \times n \times n$ and the encoding b_{ij} with the shape of $n \times n$ in shared scheme or $h \times n \times n$ in unshared scheme takes the computational cost of $\mathcal{O}(hn^2)$. For contextual mode, in Eq. (27), the broadcast multiplication on the input embedding $\mathbf{x}_i\mathbf{W}$ with the shape $h \times n \times d$ and the relative position weight $\mathbf{p}$ with the shape of $d \times k$ in shared scheme or $h \times d \times k$ in unshared scheme takes the computational cost of $\mathcal{O}(hndk)$. Due to the broadcast operations, the com-

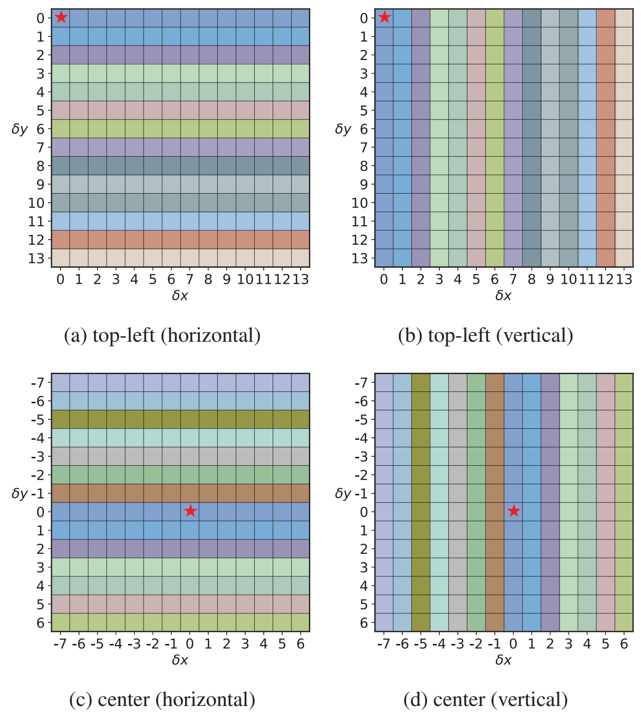

(a) top-left (horizontal) (b) top-left (vertical)
(c) center (horizontal) (d) center (vertical)

Figure 8: Visualization of Cross method. The red star ★ presents the reference position. Different color means different bucket. The relative positions with the same color share the same encoding.

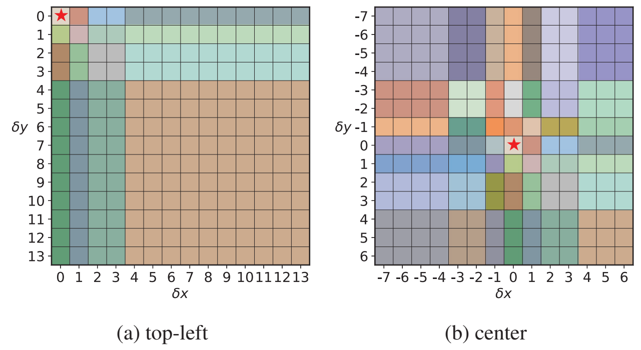

(a) top-left (b) center

Figure 9: Visualization of Product method. The red star ★ presents the reference position. Different color means different bucket. The relative positions with the same color share the same encoding.

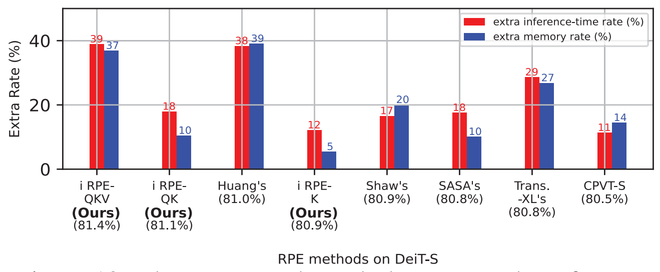


Figure 10: The extra cost brought by RPEs. The reference model is DeiT-S [11] without RPE, taking 1,096 images/s and 8,930 Mb memory.

putational cost of shared and unshared schemes is the same.

4. Injecting Previous RPE Methods into DeiT

In the Tab. 5, in order to compare with previous 1D relative position encoding methods, we utilize our Product method (defined in Sec. 3.2 in the main manuscript) to adapt 1D encoding methods for 2D images. We replace the piecewise function $g(x)$ with the clip function $h(x)$, which is matched with previous methods. The encoding weight is shared across attention heads. DeiT-S(Shaw's), DeiT-S(Trans.-XL's), DeiT-S(Huang's) are DeiT-S [11] models with Shaw's RPE [10], RPE in Transformer-XL [2] and Huang's RPE [6], respectively. Besides, the 2D RPE in SASA [9] is equipped on DeiT-S [11] directly.

5. Training and Test Settings of DETR

We follow the same training protocol and hyperparameter configurations as the original DETR [1]. The backbone model of DETR [1] is ResNet-50 [5], pretrained on ImageNet [3], and the BatchNorm layers are frozen during training. All transformer blocks are initialized with Xavier initialization [4]. The image is cropped such that the shortest side is at least 480 and at most 800 pixels while the longest at most 1333. When training, random horizontal flipping and random cropping are utilized. The initial learning rates of transformer and backbone are 10^{-4} and 10^{-5}, respectively. Learning rates are divided by 10 in the last 50 epochs in 150 epochs schedule, and the last 100 epochs in 300 epochs schedule. The optimizer is AdamW [8] with weight decay of 10^{-4} and a mini-batch size of 16. The number of queries is 100. We train the models for 150 epochs and 300 epochs.

6. The Effectiveness on Other Vision Transformers

We further verify the effectiveness of the proposed iRPE on the recent Swin transformer [7]. Specifically, the original Swin-T model without RPE obtains a top-1 accuracy of 80.5% (Tab. 4 in Swin transformer [7]), while using RPE bias mode gets +0.8% improvements. Our contextual RPE on QKV can further improve Swin-T to 81.9% on ImageNet.

7. Transfer Learning on Fine-grained Datasets

We finetune the pretrained models on Stanford Cars and CUB200_2011 datasets using the resolution 224x224 and 300 epochs. DeiT-B [11] with iRPE on keys obtains a top-1 accuracy of 93.4% and 84.9% on the two datasets respectively, outperforming the original DeiT-B (92.1% and 83.4%) by 1.3% and 1.5% points.

8. Inference Performance

The inference runtime and memory cost are reported in Fig. 10, tested on Nvidia V100 GPU with a batch size of 128. We can see that our iRPE on keys is more effective.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 3

[2] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, 2019. 3

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 3

[4] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010. 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[6] Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. Improve transformer models with better relative position embeddings. In *EMNLP*, 2020. 3

[7] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 1, 3

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 3

[9] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. 3

[10] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *ACL*, 2018. 3

[11] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1, 3