# Supplementary Material for "LaPE: Layer-adaptive Position Embedding for Vision Transformers with Independent Layer Normalization"

## A. Limitation of Default Absolute PE Joining Method

In order to decouple the position information for each Transformer layer, we reparameterize the input of each encoder layer and each Multi-Head Self-Attention (MSA) module, and find the defect of the default absolute PE joining method. In Vision Transformers, the absolute position embedding (PE) is added to the patch embedding $\boldsymbol{\alpha}$ at the beginning and propagated to deeper layers through the skip connections. The input of each encoder layer can be rewritten as:

$$\begin{aligned}
\boldsymbol{x}_l &= \boldsymbol{x}_{l-1} + \boldsymbol{x}'_{l-1} + \boldsymbol{x}''_{l-1} \\
&= \boldsymbol{x}_{l-2} + \boldsymbol{x}'_{l-2} + \boldsymbol{x}''_{l-2} + \boldsymbol{x}'_{l-1} + \boldsymbol{x}''_{l-1} \\
&= \boldsymbol{x}_0 + \boldsymbol{x}'_0 + \boldsymbol{x}''_0 + ... + \boldsymbol{x}'_{l-1} + \boldsymbol{x}''_{l-1} \\
&= \boldsymbol{\alpha} + \boldsymbol{\omega} + \sum_{k=0}^{l-1} (\boldsymbol{x}'_k + \boldsymbol{x}''_k) \\
&= \tilde{\boldsymbol{x}}_l + \boldsymbol{\omega},
\end{aligned} \tag{1}$$

where $l$ is the index of layer, and $\boldsymbol{x}'$, $\boldsymbol{x}'' \in \mathbb{R}^{N \times D}$ (N is the token number, D is the dimension) represent the output of each MSA and MLP module (refer to Fig. 2 of our paper for more details). We use $\tilde{\boldsymbol{x}}_l$ to represent $\boldsymbol{\alpha} + \sum_{k=0}^{l-1} (\boldsymbol{x}'_k + \boldsymbol{x}''_k)$. In this way, we separate the input of each layer into two parts, PE and token embeddings.

We can further rewrite the input of each MSA module.

$$\begin{aligned}
\boldsymbol{x}_l{}' &= \mathrm{MSA}_l(\mathrm{LN}_l(\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega})) \\
&= \mathrm{MSA}_l(\boldsymbol{\gamma}_l * \frac{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega} - \mathrm{E}[\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}]}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} + \boldsymbol{\beta}_l) \\
&= \mathrm{MSA}_l(\boldsymbol{\gamma}_l * \frac{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega} - \mathrm{E}[\tilde{\boldsymbol{x}}_l] - \mathrm{E}[\boldsymbol{\omega}]}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} + \boldsymbol{\beta}_l) \\
&= \mathrm{MSA}_l(\boldsymbol{\gamma}_l * \frac{\tilde{\boldsymbol{x}}_l - \mathrm{E}[\tilde{\boldsymbol{x}}_l]}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} + \boldsymbol{\gamma}_l * \frac{\boldsymbol{\omega} - \mathrm{E}[\boldsymbol{\omega}]}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} + \boldsymbol{\beta}_l) \\
&= \mathrm{MSA}_l((\frac{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l}}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} \boldsymbol{\gamma}_l * \frac{\tilde{\boldsymbol{x}}_l - \mathrm{E}[\tilde{\boldsymbol{x}}_l]}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l}}) + (\frac{\boldsymbol{\sigma}_{\boldsymbol{\omega}}}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} \boldsymbol{\gamma}_l * \frac{\boldsymbol{\omega} - \mathrm{E}[\boldsymbol{\omega}]}{\boldsymbol{\sigma}_{\boldsymbol{\omega}}}) + \boldsymbol{\beta}_l) \\
&= \mathrm{MSA}_l(\frac{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l}}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} (\boldsymbol{\gamma}_l * \frac{\tilde{\boldsymbol{x}}_l - \mathrm{E}[\tilde{\boldsymbol{x}}_l]}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l}} + \boldsymbol{\beta}_l) + \frac{\boldsymbol{\sigma}_{\boldsymbol{\omega}}}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} (\boldsymbol{\gamma}_l * \frac{\boldsymbol{\omega} - \mathrm{E}[\boldsymbol{\omega}]}{\boldsymbol{\sigma}_{\boldsymbol{\omega}}} + \boldsymbol{\beta}_l) + \frac{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}} - \boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l} - \boldsymbol{\sigma}_{\boldsymbol{\omega}}}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} \boldsymbol{\beta}_l) \\
&= \mathrm{MSA}_l(\frac{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l}}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} \mathrm{LN}_l(\tilde{\boldsymbol{x}}_l) + \frac{\boldsymbol{\sigma}_{\boldsymbol{\omega}}}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} \mathrm{LN}_l(\boldsymbol{\omega}) + \frac{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}} - \boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l} - \boldsymbol{\sigma}_{\boldsymbol{\omega}}}{\boldsymbol{\sigma}_{\tilde{\boldsymbol{x}}_l + \boldsymbol{\omega}}} \boldsymbol{\beta}_l) \\
&= \mathrm{MSA}_l(\boldsymbol{\lambda_1} \mathrm{LN}_l(\tilde{\boldsymbol{x}}_l) + \boldsymbol{\lambda_2} \mathrm{LN}_l(\boldsymbol{\omega}) + \boldsymbol{\lambda_3} \boldsymbol{\beta}_l),
\end{aligned} \tag{2}$$

where $l$ is the index of layer, LN represents the layer normalization operation, MSA represents the Multi-Head Self-Attention module, $\boldsymbol{\gamma} \in \mathbb{R}^{1 \times D}$ and $\boldsymbol{\beta} \in \mathbb{R}^{1 \times D}$ are the trainable affine transformation coefficients in LN, $\mathrm{E}(\cdot) \in \mathbb{R}^{N \times 1}$ and $\boldsymbol{\sigma}_{(\cdot)} \in \mathbb{R}^{N \times 1}$ are the token-wise mean and standard deviation. We use $\boldsymbol{\lambda_1}$, $\boldsymbol{\lambda_2}$, $\boldsymbol{\lambda_3} \in \mathbb{R}^{N \times 1}$ to represent the coefficients of $\mathrm{LN}(\tilde{\boldsymbol{x}}_l)$, $\mathrm{LN}(\boldsymbol{\omega})$, $\boldsymbol{\beta}_l$, respectively. In this way, we successfully decouple the position information $\boldsymbol{\lambda_2} \mathrm{LN}(\boldsymbol{\omega})$ for each Transformer layer (we ignore some minor couplings in $\boldsymbol{\omega}$).

From Eq. 2, we can see that the position embedding $\omega$ shares the same LN with the token embeddings $\tilde{x}_l$. However, the position and token embeddings represent different information and have different embedding distributions. When they are coupled together, the affine parameters in LN have to trade off between them, limiting the expressiveness of position embedding.

## B. PyTorch-Like Pseudo Code Implementation

We provide PyTorch-like codes here for easier understanding and better reproducibility of our proposed LaPE.

For the default absolute PE joining method, Vision Transformers (VTs) add the position embedding (PE) to the patch embedding before entering Transformer encoders, and deliver the same PE to each encoder layer. We take the framework of DeiT [12] for example:

```
# Attention: Multi-Head Self-Attention calculation
# MLP: Multi-Layer Perceptron: Linear+Gelu+Linear+Dropout
def Block(x):
    x = x + Attention(LayerNorm1(x))
    x = x + MLP(LayerNorm2(x))
    return x

# DeiT with PE joined by default
def VisionTransformer(x):
    x = Patch_embed(x)
    x = cat(cls_token, x)
    x = x + pos_embed # Eq.(2): add PE to the patch embedding
    for _ in range(depth):
        x = Block(x)
    x = Head(x[:, 0])
    return x
```

In contrast, LaPE provides independent LNs for PE and token embeddings in each layer, and adds the layer normalized PE and token embeddings together as the input of MSA, and the PE is delivered progressively among layers. The implementation is as follows:

```
# Attention: Multi-Head Self-Attention calculation
# MLP: Multi-Layer Perceptron: Linear+Gelu+Linear+Dropout
def LaPE_Block(x, pos_embed):
    # Eq.(9): use independt LN to PE and token embedding
    x = x + Attention(LayerNorm1(x)+LayerNorm2(pos_embed))
    x = x + MLP(LayerNorm3(x))
    pos_embed = LayerNorm2(pos_embed) # Eq.(11): pass PE progressively
    return x, pos_embed

# DeiT with LaPE
def VisionTransformer(x):
    x = Patch_embed(x)
    x = cat(cls_token, x)
    x = x # Eq.(8): pass the patch embedding to encoders
    for _ in range(depth):
        x, pos_embed = LaPE_Block(x,pos_embed)
    x = Head(x[:, 0])
    return x
```

| Dataset | Model | Learning Rate | Learning Rate Scheduler | Weight Decay | Batch Size | Epochs | Warm-up Epochs |
|---|---|---|---|---|---|---|---|
| ImageNet [2] | DeiT [12] | 5e-4 | cosine, min_lr=1e-5 | 0.05 | 1024 | 300 | 5 |
| | T2T-ViT [16] | 1e-3 | cosine, min_lr=1e-5 | 0.03 | 1024 | 300+10 (cool_down epochs) | 10 |
| | Swin [8] | 5e-4 | cosine, min_lr=5e-6 | 0.05 | 512 | 300 | 20 |
| | CeiT [15] | 5e-4 | cosine, min_lr=1e-5 | 0.05 | 1024 | 300 | 5 |
| Cifar-10 [6] | ViT_Lite [4] | 55e-5 | cosine, min_lr=1e-5 | 0.06 | 128 | 300+10 (cool_down epochs) | 10 |
| | CVT [4] | | | | | | |
| | CCT [4] | | | | | | |
| Cifar-100 [6] | ViT_Lite [4] | 6e-4 | cosine, min_lr=1e-5 | 0.06 | 128 | 300+10 (cool_down epochs) | 10 |
| | CVT [4] | | | | | | |
| | CCT [4] | | | | | | |

Table I. **Image classification experimental settings on ImageNet, CIFAR10 and CIFAR100.**

## C. Analysis on Position Correlation

Many works studying how PE works in Transformers reveal that PE works as a position inductive bias [3, 13, 14, 5]. Specifically, PE provides its position correlation to the Query-Key product in Multi-Head Self-Attention (MSA) calculation, and such correlation guides tokens to attend more to the adjacent tokens. For a better understanding, we can explain it with equations of the Query-Key product in LaPE-based VTs. The Query-Key product of MSA in $l$th layer is:

$$\begin{aligned} QK^{\mathrm{T}} &= (\mathrm{LN}_{\boldsymbol{x}|l}(\boldsymbol{x}_l) + \mathrm{LN}_{\boldsymbol{\omega}|l}(\boldsymbol{\omega}_l))W_Q{W_K}^{\mathrm{T}}(\mathrm{LN}_{\boldsymbol{x}|l}(\boldsymbol{x}_l) + \mathrm{LN}_{\boldsymbol{\omega}|l}(\boldsymbol{\omega}_l))^{\mathrm{T}} \\ &= \mathrm{LN}_{\boldsymbol{x}|l}(\boldsymbol{x}_l)W_Q{W_K}^{\mathrm{T}}\mathrm{LN}_{\boldsymbol{x}|l}(\boldsymbol{x}_l)^{\mathrm{T}} + \mathrm{LN}_{\boldsymbol{x}|l}(\boldsymbol{x}_l)W_Q{W_K}^{\mathrm{T}}\mathrm{LN}_{\boldsymbol{\omega}|l}(\boldsymbol{\omega}_l)^{\mathrm{T}} \\ &+ \mathrm{LN}_{\boldsymbol{\omega}|l}(\boldsymbol{\omega}_l)W_Q{W_K}^{\mathrm{T}}\mathrm{LN}_{\boldsymbol{x}|l}(\boldsymbol{x}_l)^{\mathrm{T}} + \mathrm{LN}_{\boldsymbol{\omega}|l}(\boldsymbol{\omega}_l)W_Q{W_K}^{\mathrm{T}}\mathrm{LN}_{\boldsymbol{\omega}|l}(\boldsymbol{\omega}_l)^{\mathrm{T}} \end{aligned} \tag{3}$$

where $W_Q$ and $W_K$ represent the linear transformation matrix.

In Eq. (3), we split the Query-Key product into four items, which represent token-token correlation, token-position correlation, position-token correlation, and position-position correlation. The 2nd and 3rd items are proved to be meaningless [5], as the image semantics (represented by token embedding) has little correlation with its position (represented by PE). The 4th item are correlations between projected $\mathrm{LN}_{\boldsymbol{\omega}|l}(\boldsymbol{\omega}_l)$, which is essentially the position correlation. Moreover, such correlation can be measured with the cosine similarity of $\mathrm{LN}_{\boldsymbol{\omega}|l}(\boldsymbol{\omega}_l)$.

In this paper, we decouple each layer's $\boldsymbol{\lambda_2}\mathrm{LN}_l(\boldsymbol{\omega})$ for default method and $\mathrm{LN}_{\boldsymbol{\omega}|l}(\boldsymbol{\omega}_l)$ for LaPE, and visualize their position correlations. The visualization results of Fig. 3 and Fig. 4 in our main text verify the superiority of our proposed LaPE.

## D. Experimental Settings

### D.1. Image Classification

For image classification, ViT_Lite, CVT, and CCT [4] on CIFAR use the SGD [10] as the optimizer, while DeiT [12], T2T-ViT [16], Swin [8] and CeiT [15] on ImageNet-1K all use the Adamw [9]. We list the hyper-parameters and settings used in our paper in TableI, which are the same as those used in the original papers.

### D.2. Object Detection

For object detection, we conduct experiments on ViT-Adapter[1]. The settings are slightly different from those used in the original paper, considering we conduct experiments on 4 GPUs with a batch size of 8, while the original paper uses 8 GPUs with a batch size of 16. The detailed training settings are shown in Table II.

| Dataset | Name | Method | Backbone | Pre-trained Model | Crop Size | Optimizer | LR | LR Scheduler | Weight Decay | Batch Size |
|---|---|---|---|---|---|---|---|---|---|---|
| COCO 2017 [7] | ViT-Adapter [1] | Mask R-CNN | ViT-Adapter-Ti | DeiT-Ti | 1024 | AdamW | 1e-4 | 3x+MS | 0.05 | 8 |
| | | | ViT-Adapter-S | DeiT-S | | | | | | |

Table II. **Object detection experimental settings on COCO val2017.**

| Dataset | Name | Method | Backbone | Pre-trained Model | Crop Size | Optimizer | LR | LR Scheduler | Weight Decay | Batch Size |
|---|---|---|---|---|---|---|---|---|---|---|
| ADE20K [17] | Segmenter [11] | Seg-Ti-Mask/16 | DeiT-Ti | DeiT-Ti | 512 | SGD | 1e-3 | 160k | 0. | 8 |
| | | Seg-S-Mask/16 | DeiT-S | DeiT-S | | | | | | |
| | ViT-Adapter [1] | UperNet | ViT-Adapter-Ti | DeiT-Ti | 512 | AdamW | 2e-5 | 160K | 0.05 | 8 |
| | | | ViT-Adapter-S | DeiT-S | | | | | | |

Table III. **Semantic segmentation experimental settings on ADE20K.** Seg-Ti/S-Mask/16 represents methods using mask transformer as the decoder with 16×16 input patch size.

### D.3. Semantic Segmentation

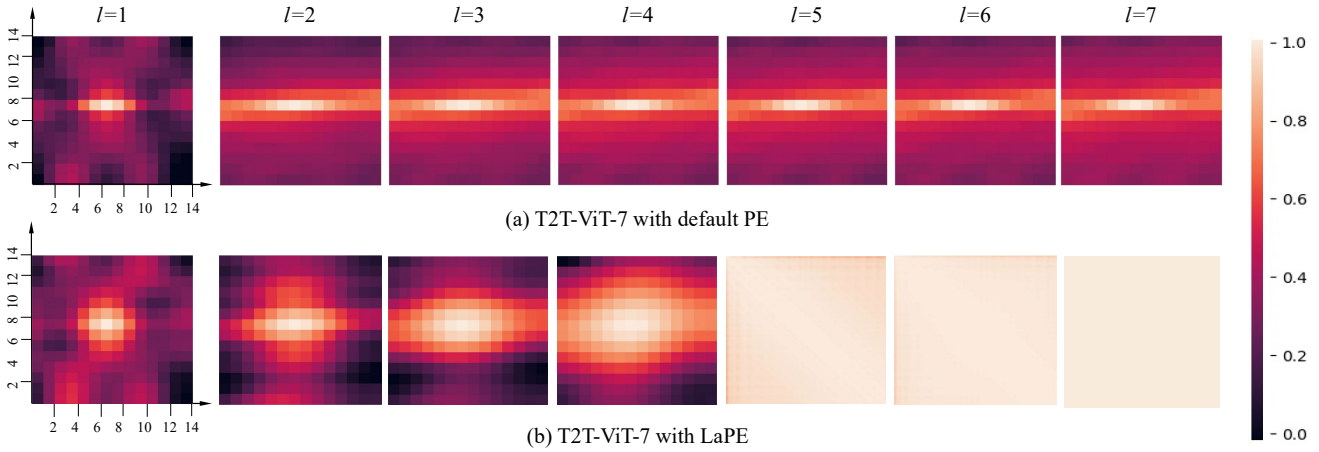
For semantic segmentation, we conduct experiments on Segmenter [11] and ViT-Adapter [1]. The training settings are the same as those introduced in original papers, except for the pre-trained model of Segmenter [11] and the batch size of ViT-Adapter. For Segmenter, we use the DeiT (pre-trained on ImageNet-1K) to initialize the model, different from ViT (pre-trained on ImageNet-22K) used in the original paper. For ViT-Adapter, we conduct experiments on 4 GPUs and with a batch size of 8, compared with 8 GPUs and a batch size of 16 in the original paper. The detailed training settings are shown in Table III.

## E. Visualization

In our main text, we provide partial layer's visualization of the position correlation of T2T-ViT [16] and DeiT [12]. Here we supply the complete visualization for them.

In Fig. I, the visualization results strongly support our analysis: (1) The shared LN in the default PE method trades off between token embedding and PE. (2) LaPE significantly enhances the expressiveness of PE. As shown in Fig. I (a), the position correlation in the first layer is obviously adjusted, because it no longer has the same 1-D correlation as sinusoidal PE, which means the affine parameters in LN learn to adjust the position correlation of PE. However, the position correlations in the latter layers are almost monotonic and unadjusted, which means the affine parameters do not learn to enhance the expressiveness of PE. This phenomenon verifies our analysis (1). As shown in Fig. I (b), the position correlations are adjusted into 2-D, as they have both horizontal and vertical perception. Moreover, the position correlations are layer adapted into hierarchical, which verifies the analysis (2).

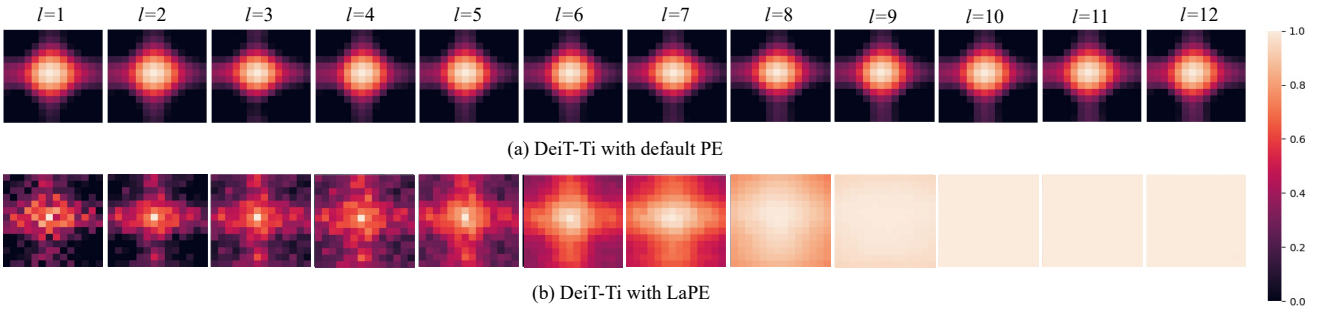
In Fig. II, the position correlations of default PE are monotonic, while the correlations of LaPE are hierarchical, which better fits the way Vision Transformers process images.




(a) T2T-ViT-7 with default PE

(b) T2T-ViT-7 with LaPE

Figure I. **Visualization of the position correlations in each layer for T2T-ViT-7.** (a) The position correlation is obviously changed into 2-D in the first layer, while the correlations are 1-D and monotonic in the latter layers. This phenomenon demonstrates that the affine parameters in LN trade off between token embedding and PE, which limits the expressiveness of PE. (b) The position correlations are 2-D and change from local to global, which means LaPE can improve the expressiveness of PE greatly.




(a) DeiT-Ti with default PE

(b) DeiT-Ti with LaPE

Figure II. **Visualization of the position correlations at different layers for DeiT-Ti.** (a) The default position correlations are monotonic. (b) The position correlations of LaPE change from local to global as the layer goes deeper.

## References


[1] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. 2022.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Philipp Dufter, Martin Schmitt, and Hinrich Schütze. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763, 2022.

[4] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

[5] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. *International Conference on Learning Representations*, 2021.

[6] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[8] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

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

[10] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[11] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7262–7272, 2021.

[12] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[13] Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. On position embeddings in bert. In *International Conference on Learning Representations*, 2020.

[14] Yu-An Wang and Yun-Nung Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. *arXiv preprint arXiv:2010.04903*, 2020.

[15] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 579–588, 2021.

[16] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021.

[17] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 2019.