Positional Label for Self-Supervised Vision Transformer

by Zhemin Zhang and Xun Gong

MI

Filip Kołodziejczyk

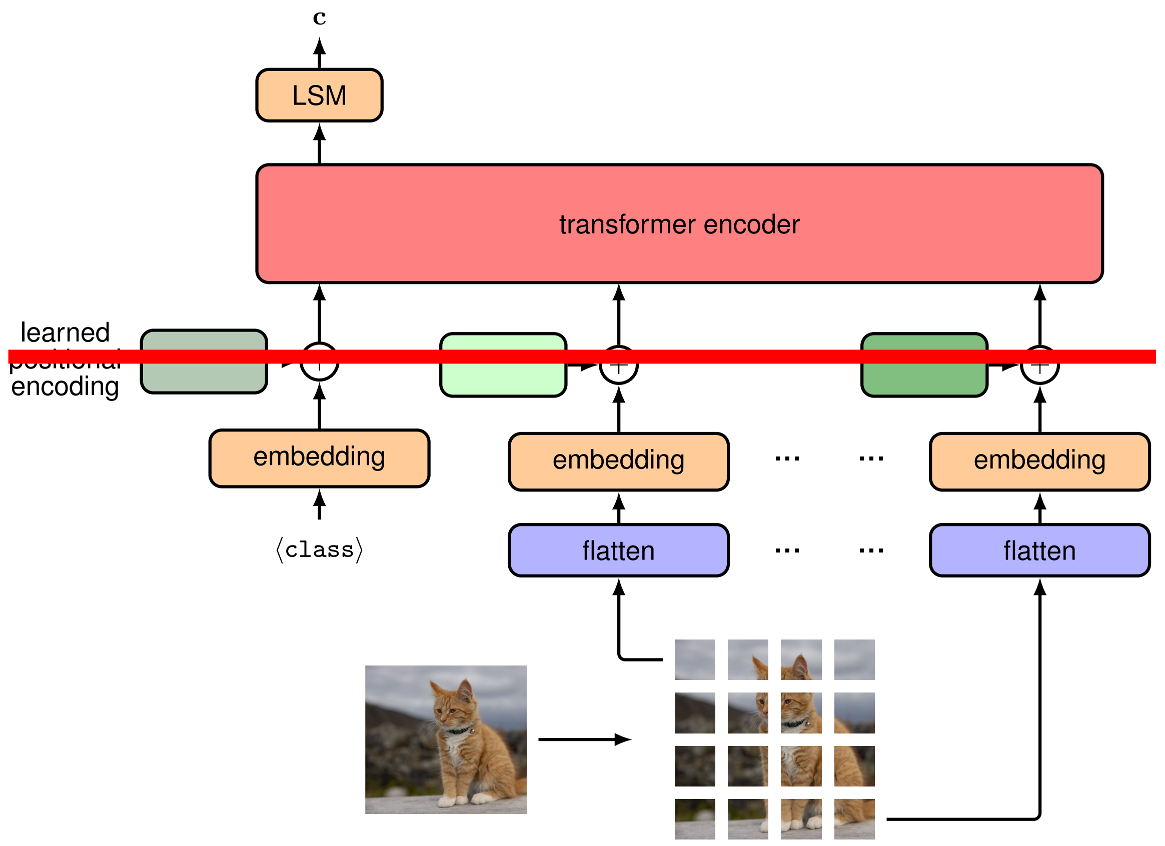
MI2.AI Seminar, Warsaw, October 21th, 2024

Agenda

Note
This presentation is a commentary/discussion on a paper titled *Positional Label for Self-Supervised Vision Transformer*, authored by Zhemin Zhang and Xun Gong. Accordingly, all information and graphics have their source in this article if not stated otherwise.

Vision Transformers (ViT)

- Transformers make very few assumptions about input data. Hence, they can be used with images.
- Instead of words (tokens), we have pixels. The single-pixel approach would be too expensive (computationally). A common approach is to split an image into a set of equally sized, non-overlapping patches projected into a one-dimensional vector.
- The classification task introduces a unique token, *CLS*. The corresponding transformer output is then passed to the classifier.
- Self-attention is permutation-invariant. It does not capture the spatial arrangement of input. **Thus, important information needs to be recovered.**

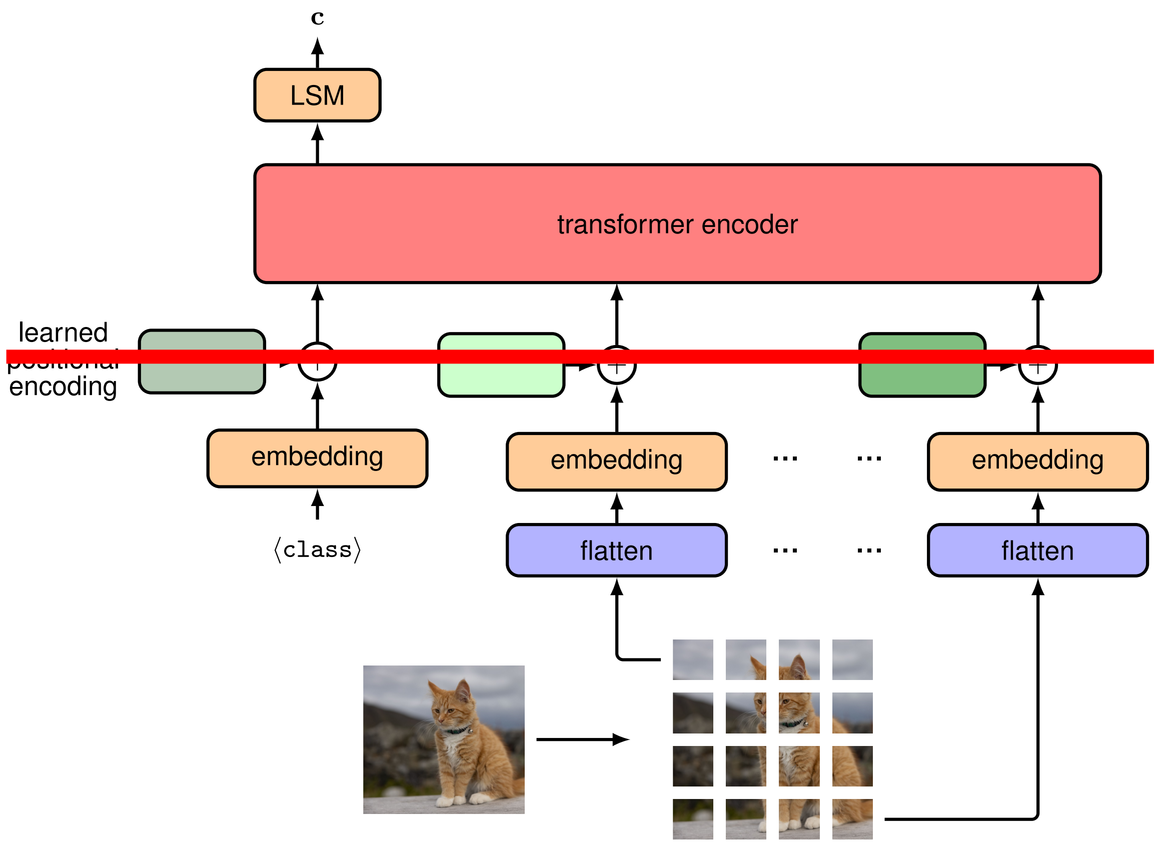


Source: Deep Learning: Foundations and Concepts by C.M. Bishop and H. Bishop

ViT w/o positional information

$$[y_{class}; y_1; y_2; \dots; y_N] = [x_{class}; x_m^1 E; x_m^2 E; \dots; x_m^N E] \quad (3.)$$
$$[z_{class}; z_1; z_2; \dots; z_N] = \mathrm{ViT}_\theta([y_{class}; y_1; y_2; \dots; y_N]) \quad (4.)$$

1. Image $x \in \mathbb{R}^{H \times W \times C}$ is split into fixes-size patches $x_m \in \mathbb{R}^{N \times (m^2 C)}$. (m, m) is a patch resolution and $N = \frac{HW}{m^2}$ is the number of generated patches. Alternatively to raw image patches, feature maps from CNN can be used.
2. x_{class} token is prepened – its output state z_{class} is later used in classification task.
3. Linear projection $E \in \mathbb{R}^{(m^2 C) \times D}$ is applied to patches, resulting in D dimensional embeddings.
4. Embeddings sequence is feed to a ViT encoder.
5. z_{class} is converted into logit vector, softmax function convertes further into probabilites and then cross-entropy loss L_s is calculated.



Source: Deep Learning: Foundations and Concepts by C.M. Bishop and H. Bishop

Positional encoding (PE) – related work

There are two standard approaches to adding positional information for ViT:

- **Absolute PE** - each absolute position from the input sequence (1 to max sequence length) has an associated encoding vector. The encoding vector is combined with related tokens from the sequence, usually by element-wise addition. Encoding vectors can be
    - Fixed (Attention Is All You Need)
    - Learned via training parameters (Convolutional sequence to sequence learning)
- **Relative PE** - relative position between tokens (based on their absolute positions) is calculated pairwise. Then attention score between pairs accounts for this distance (thus formula is adjusted). Some of such methods:
    - Initial proposition (Self-Attention with Relative Position Representations)
    - Modification capturing longer-term dependency and resolves context fragmentation problem (Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context)
    - ViT dedicated (Rethinking and Improving Relative Position Encoding for Vision Transformer, Stand-Alone Self-Attention in Vision Model, Bottleneck Transformers for Visual Recognition)

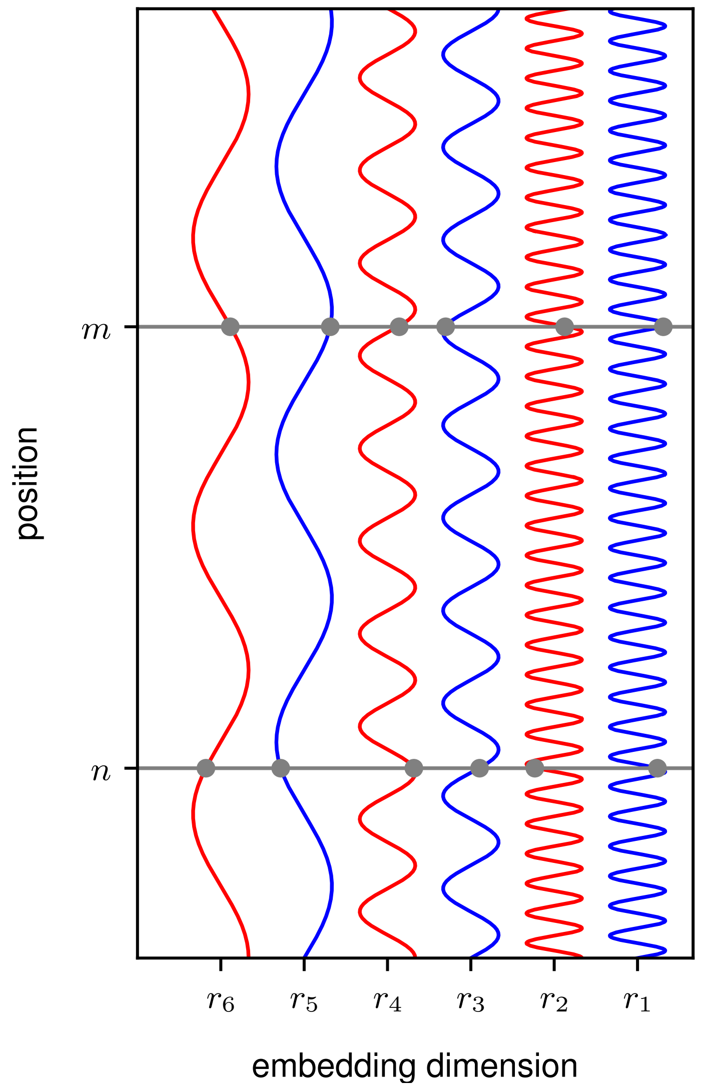
PE – absolute and fixed

Introduced in Attention Is All You Need. Thanks to the use of sinusoidal functions, it is invariant to the sequence length and unique for each position. i – position, L – input sequence length

$$r_{ni} = \begin{cases} \sin\left(\frac{n}{L^{i/D}}\right), & \text{if } i \text{ is even} \\ \cos\left(\frac{n}{L^{(i-1)/D}}\right), & \text{if } i \text{ is odd} \end{cases}$$

3. […] Linear projection $E \in \mathbb{R}^{(m^2C)\times D}$ is applied to patches, resulting in D dimensional embeddings.
4. PE is added to embeddings by element-wise addition.
5. Embeddings sequence is feed to a ViT encoder […]

$$[y_{class}; y_1; y_2; \dots; y_N] = [x_{class}; x_m^1E; x_m^2E; \dots; x_m^NE] \quad (3.)$$
$$[\hat{y}_{class}; \hat{y}_1; \hat{y}_2; \dots; \hat{y}_N] = [y_{class}; y_1 + r_1; y_2 + r_2; \dots; y_N + r_N] \quad (4.)$$
$$[z_{class}; z_1; z_2; \dots; z_N] = \text{ViT}_\theta([\hat{y}_{class}; \hat{y}_1; \hat{y}_2; \dots; \hat{y}_N]) \quad (5.)$$

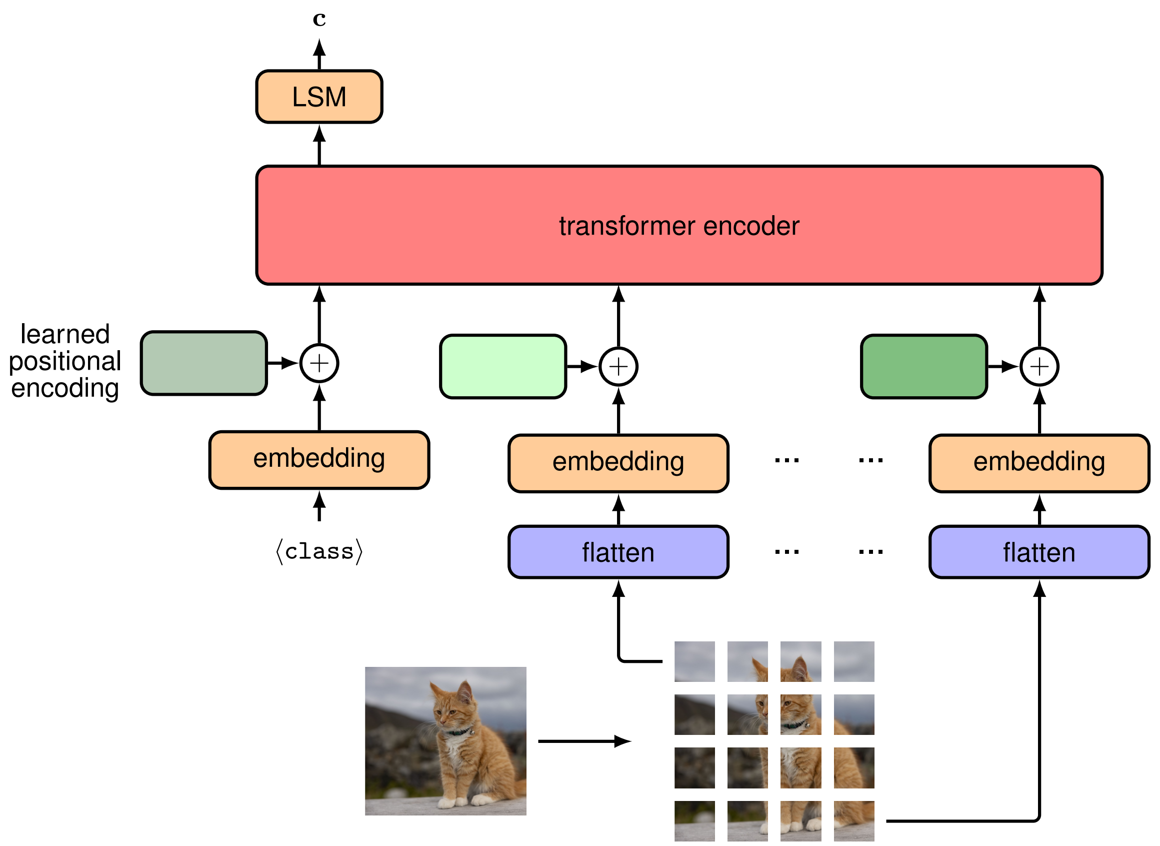


Source: Deep Learning: Foundations and Concepts by C.M. Bishop and H. Bishop

PE – absolute and parametried

Introduced in Convolutional sequence to sequence learning. Compared to the fixed values, they are initialized randomly and learned via backpropagation.

3. […] Linear projection $E \in \mathbb{R}^{(m^2C)\times D}$ is applied to patches, resulting in D dimensional embeddings.
4. PE is added to embeddings by element-wise addition. Values are updated during the proces.
5. Embeddings sequence is feed to a ViT encoder […]



Source: Deep Learning: Foundations and Concepts by C.M. Bishop and H. Bishop

$$[y_{class}; y_1; y_2; \dots; y_N] = [x_{class}; x_m^1 E; x_m^2 E; \dots; x_m^N E] \quad (3.)$$
$$[\hat{y}_{class}; \hat{y}_1; \hat{y}_2; \dots; \hat{y}_N] = [y_{class}; y_1 + r_1; y_2 + r_2; \dots; y_N + r_N] \quad (4.)$$
$$[z_{class}; z_1; z_2; \dots; z_N] = \mathrm{ViT}_\theta([\hat{y}_{class}; \hat{y}_1; \hat{y}_2; \dots; \hat{y}_N]) \quad (5.)$$

PE – relative (simplified example)

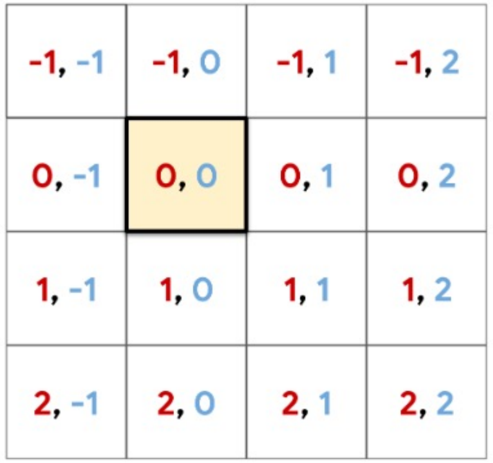
Standard self-attention formula
($W^Q, W^K, W^V \in \mathbb{R}^{d_x \times d_z}$ - query, key, and, value weights):

$$z_i = \sum_{j=1}^{N} \operatorname{softmax} e_{ij} \; (y_j W^V), \qquad e_{ij} = \frac{(y_i W^Q)(y_j W^K)^T}{\sqrt{d_z}}$$

Modified attention score to contain relative position
(r – learnable parameter, $i - j$ – relative distance between y_i and y_j):

$$e_{ij} = \frac{(y_i W^Q)(y_j W^K)^T + (y_i W^Q)(r_{i-j})^T}{\sqrt{d_z}}$$

$i - j$ might be based e.g. on spatial offset or eculidean distance.



-1, -1	-1, 0	-1, 1	-1, 2
0, -1	0, 0	0, 1	0, 2
1, -1	1, 0	1, 1	1, 2
2, -1	2, 0	2, 1	2, 2

Figure 4: An example of relative distance computation. The relative distances are computed with respect to the position of the highlighted pixel. The format of distances is *row offset*, *column offset*.

$$[z_{class}; z_1; z_2; \dots; z_N] = \mathrm{ViT}_\theta([y_{class}; y_1; y_2; \dots; y_N]) \quad (4.)$$

Positional labels (PL) – alternative to PE

- Instead of encoding, added before or inside the ViT, an **auxiliary training task** is added. **A lightweight MLP** block processes the ViT output and **outputs positional labels for each image patch** (classification).
- A new loss function - **position loss** - is introduced for this task. It is joined with standard softmax loss from the primary classification task.
- As a result, position information is **implicitly** contained with the input token itself.
- The additional goal is to **enhance the model performance** compared to standard encoding.
- Both **absolute and relative variants** are proposed and implemented.

Absolute positional label

4. [...] Embeddings sequence is feed to a ViT encoder.
5. z_{class} is "classified" (loss function L_S).
6. z_i is representation of the image patch x_m^i. Half of it is intercepted and named $\frac{1}{2}z_i$. Trainable lightweight MLP block projects $\frac{1}{2}z_i$ as a positional feature vector p_i.
7. Softmax converts it to probability that p_i belongs to y_i.
8. Cross-entropy loss $\boldsymbol{L_P}$ is calculated, called **position loss**.
9. Final loss is a **joint $\boldsymbol{L_S}$ and $\boldsymbol{L_P}$** (balanced by hyperparameter λ).

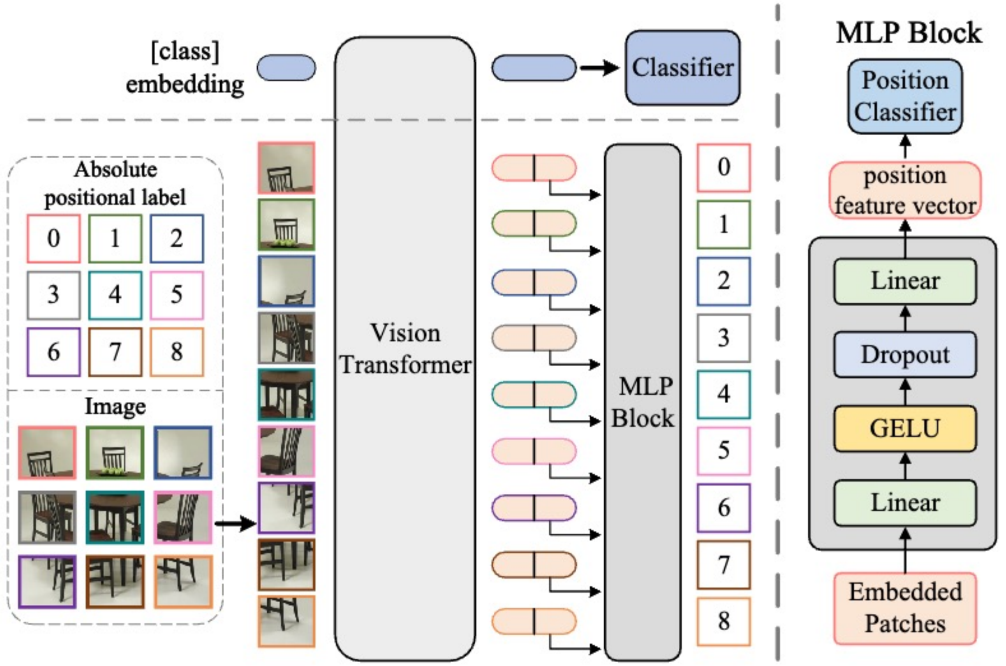
$$[z_{class}; z_1; z_2; \dots; z_N] = \mathrm{ViT}_\theta([y_{class}; y_1; y_2; \dots; y_N]) \quad (4.)$$

$$[p_1; p_2; \dots p_N] = \mathrm{MLP}\left(\left[\frac{1}{2}z_1; \frac{1}{2}x_2; \dots; \frac{1}{2}z_N\right]\right) \quad (6.)$$

$$P_{p_i,y_i} = \frac{\exp(w_{y_i}^T p_i)}{\sum_{j=1}^{c} \exp(w_j^T p_i)} \quad (7.)$$

$$L_P = -\frac{1}{N}\sum_{i=1}^{N} \log(P_{p_i,y_i}) \quad (8.)$$

$$L = L_S + \lambda L_P \quad (9.)$$



Relative positional label

4. […] Embeddings sequence is feed to a ViT encoder.
5. z_{class} is "classified" (loss function L_S).
6. To establish pairwise positional relationship between patches, intercepted halves z_i are combined in pairs $z_{ij} = \text{concat}(\frac{1}{2}z_i, \frac{1}{2}z_j)$. Trainable lightweight MLP block projects z_{ij} as a relative positional feature vector p_{ij}^r.
7. Softmax converts it to relatve that p_{ij}^r belongs to y_{ij} (relative position calculated with relative PE based on Swin-VIT method).
8. Cross-entropy loss L_P^r is calculated, called **relative position loss**.
9. Final loss is a **joint L_S and L_P^r** (balanced by hyperparameter λ).

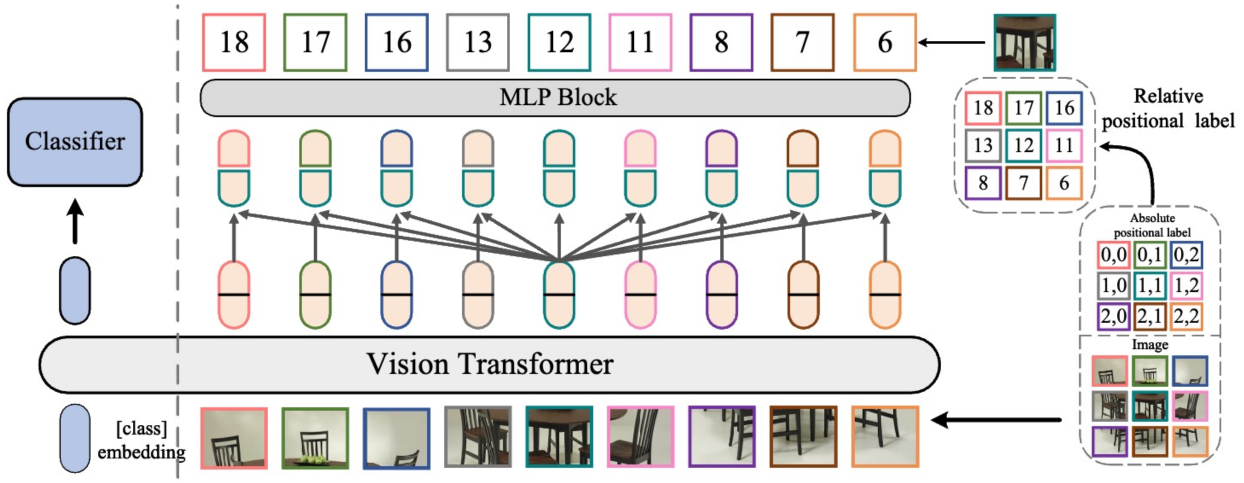
$$[z_{class}; z_1; z_2; \dots; z_N] = \text{ViT}_\theta([y_{class}; y_1; y_2; \dots; y_N]) \quad (4.)$$

$$p_{ij}^r = \text{MLP}\left(\left(\text{z}_{\text{ij}} = \text{concat}(z_i, z_j)\right)\right) \quad (6.)$$

$$P_{p_{ij}^r, y_{ij}} = \frac{\exp\left(w_{y_{ij}}^T p_{ij}^r\right)}{\sum_{k=1}^{c} \exp(w_k^T p_{ij}^r)} \quad (7.)$$

$$L_P^r = -\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \log\left(P_{p_{ij}^r, y_{ij}}\right) \quad (8.)$$

$$L = L_S + \lambda L_P \quad (9.)$$

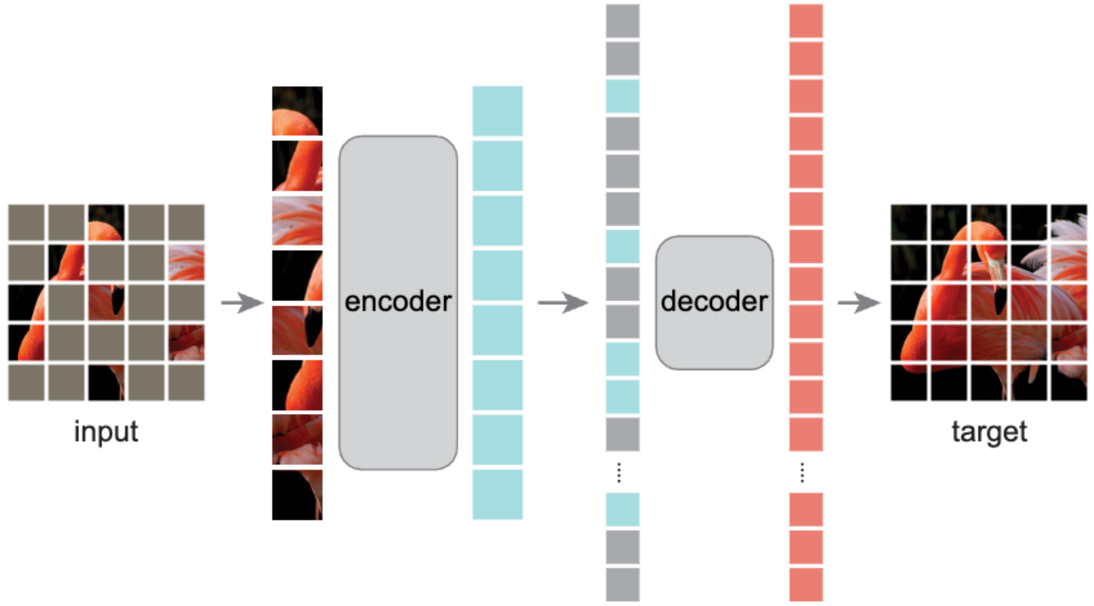


Additional use case for positional encodings – self-supervised training signal

- **Positional encoding can be used as a self-supervised signal for a self-supervised task. The authors focus on enhancing the masked autoencoders (MAE) method.**
- Self-supervised learning is a machine learning approach where a model learns to understand and represent data by using the data as a form of supervision. It leverages the inherent structure and patterns within unlabeled data to create artificial labels, enabling the model to learn meaningful representations without explicit human annotation.
- Self-supervised learning allows leveraging large amounts of unlabelled data (no need for expensive labelling processes) . It is suitable for transformer architecture, which excels at capturing complex relations within data.
- Related work mentioned in context of self-supervised learning:
  - iGTP - masks and reconstructs pixels (Generative Pretraining From Pixels)
  - MAE - masks 75% patches at random and reconstructs them (Masked Autoencoders Are Scalable Vision Learners)
  - DINO - knowledge distillation with no labels (Emerging Properties in Self-Supervised Vision Transformers)
  - DILEMMA - representation learning by detecting incorrect location embeddings (Repesentation Learning by Detecting Incorrect Location Embeddings)
  - An Empirical Study of Training Self-Supervised Vision Transformers

Masked Autoencoders (MAE)

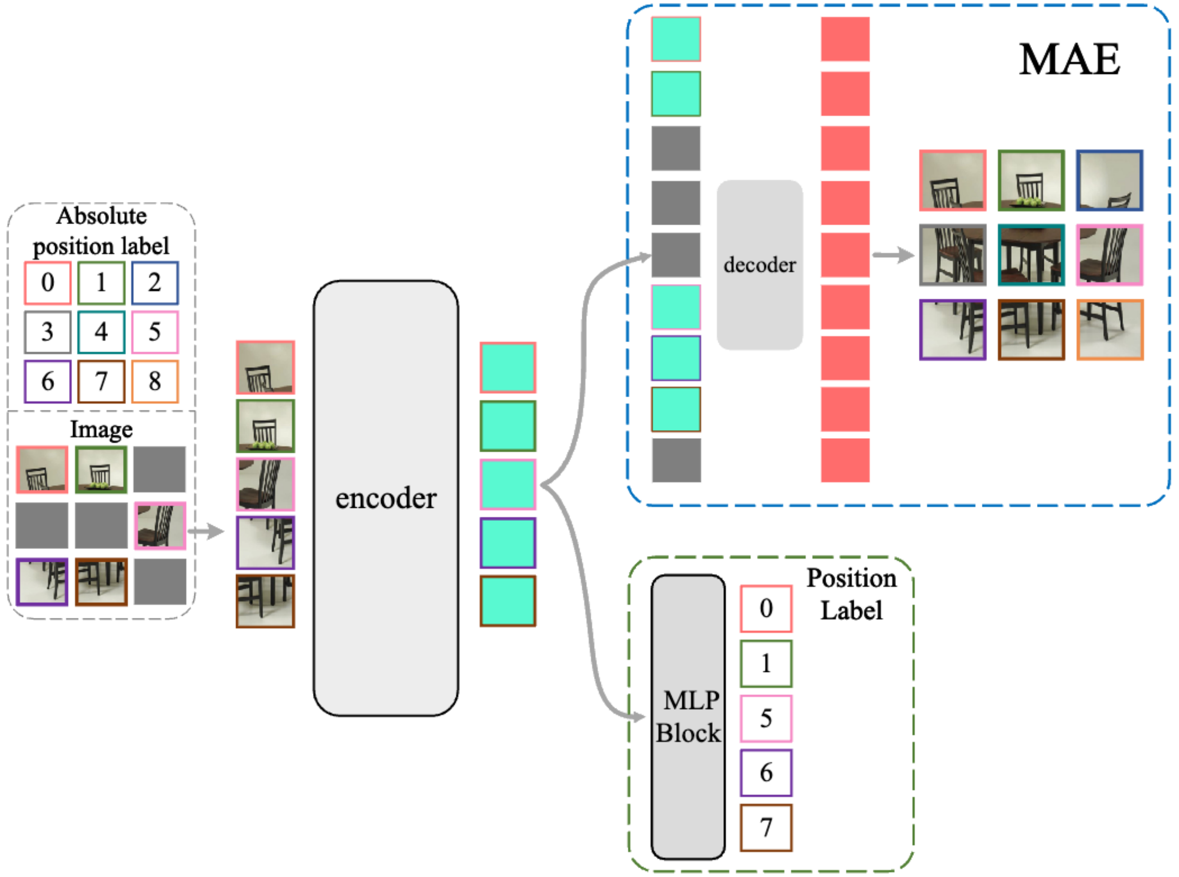
During pre-training, a large random subset of image patches (e.g., 75%) is masked out. The encoder is applied to the small subset of visible patches. Mask tokens are introduced after the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.



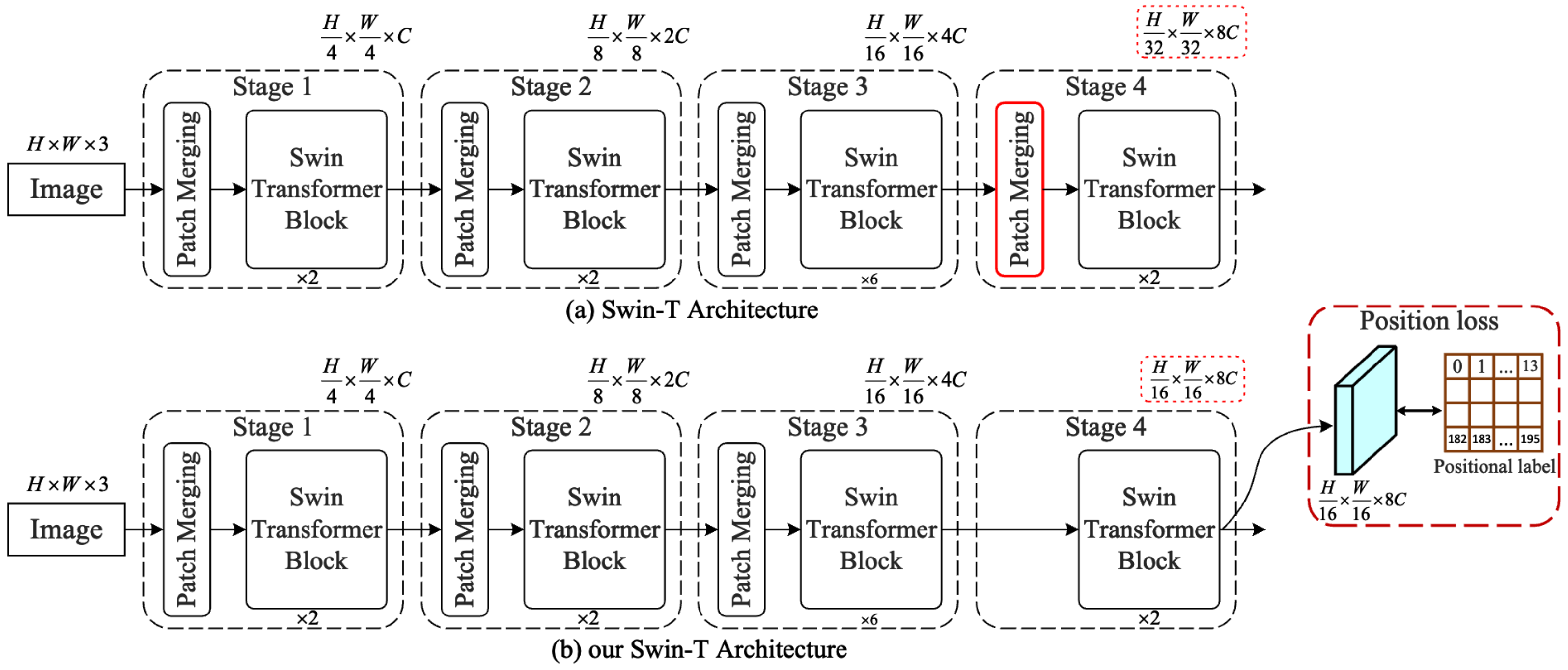
Source: [Masked Autoencoders Are Scalable Vision Learners]

Positional labels with MAE

The encoder's output is fed into the MLP block (before the mask tokens are introduced) to output the absolute positions of visible patches.



Positional labels with hierarchical ViT



$H \times W \times 3$
Image
Stage 1
Patch Merging
Swin Transformer Block
×2
$\frac{H}{4} \times \frac{W}{4} \times C$
Stage 2
$\frac{H}{8} \times \frac{W}{8} \times 2C$
Stage 3
×6
$\frac{H}{16} \times \frac{W}{16} \times 4C$
Stage 4
$\frac{H}{32} \times \frac{W}{32} \times 8C$
(a) Swin-T Architecture
$\frac{H}{16} \times \frac{W}{16} \times 8C$
Position loss
0 1 ... 13
182 183 ... 195
Positional label
(b) our Swin-T Architecture

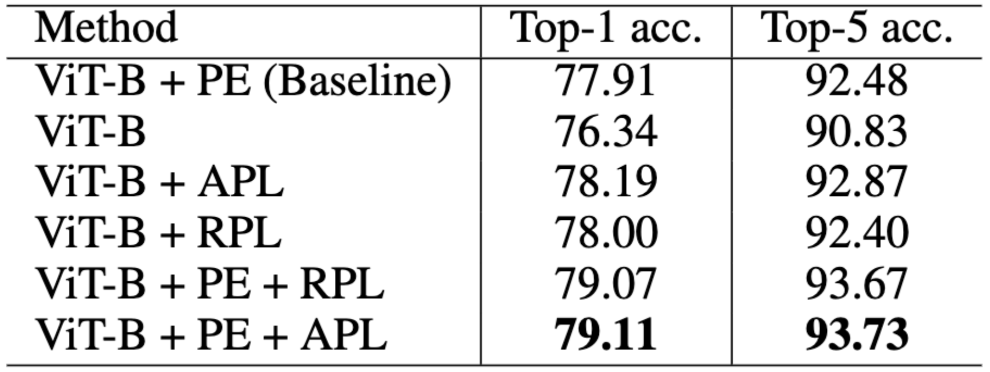
Positional labels require a specific number of output sequences (position classes). Thus, architectural adjustments are needed to combine hierarchical ViT with PL (e.g., change of patch merging blocks).

Experiments setup

- Datasets:
  - ImageNet-1K – 1.28M train, 50K validation, 100K test images, 1000 classes
  - Caltech-256 – 30K images, 256 classes (uneven)
  - Mini-ImageNet – 50K train, 10k test images, 100 classes
- Training:
  - AdamW optimizer
  - 300 epochs with LR scheduler
  - Batch size 256
  - Use of augmentation and regularization strategies
  - Hyperparameter λ set on 0.5 (if not specified otherwise)

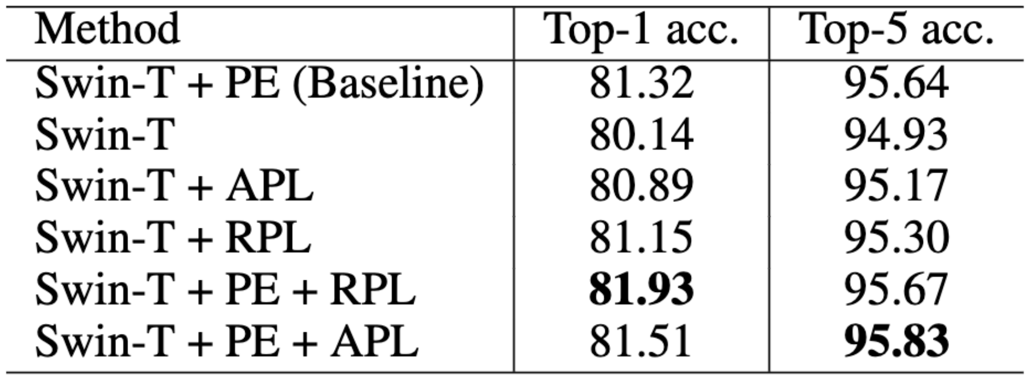
Ablation studies

Table 1: Top-1 and Top-5 accuracies (%) for ViT-B model, on the ImageNet dataset. PE: positional encoding term of ViT; APL: the proposed absolute positional label; RPL: the proposed relative positional label.



Method	Top-1 acc.	Top-5 acc.
ViT-B + PE (Baseline)	77.91	92.48
ViT-B	76.34	90.83
ViT-B + APL	78.19	92.87
ViT-B + RPL	78.00	92.40
ViT-B + PE + RPL	79.07	93.67
ViT-B + PE + APL	**79.11**	**93.73**

Table 2: Top-1 and Top-5 accuracies (%) for Swin-T model, on the ImageNet dataset.



Method	Top-1 acc.	Top-5 acc.
Swin-T + PE (Baseline)	81.32	95.64
Swin-T	80.14	94.93
Swin-T + APL	80.89	95.17
Swin-T + RPL	81.15	95.30
Swin-T + PE + RPL	**81.93**	95.67
Swin-T + PE + APL	81.51	**95.83**

Hyperparameter λ sensitivness

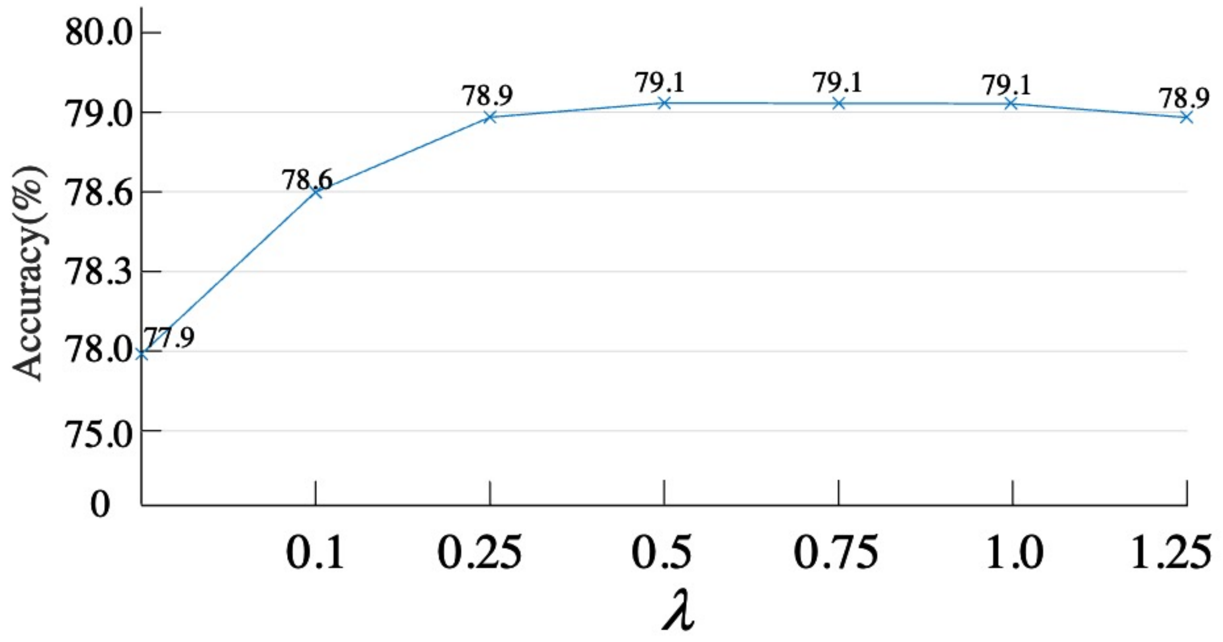


Figure 5: Illustration of position loss with different λ, using the ViT-B architecture.

Ablation studies – data augmentation

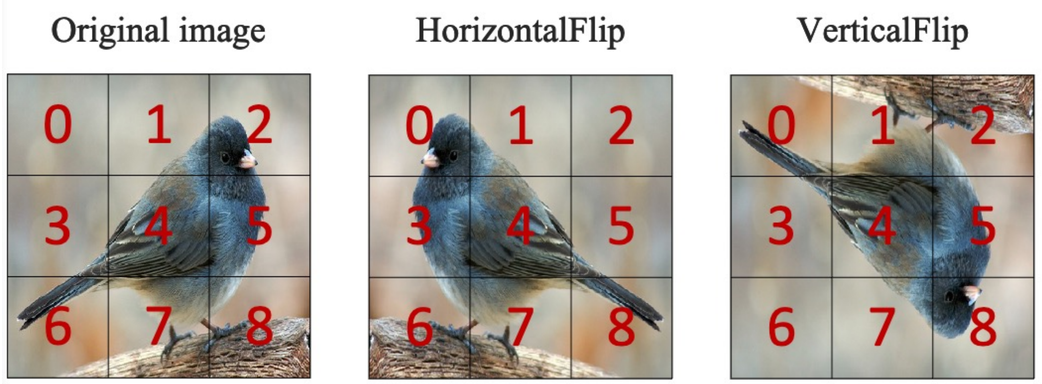


Figure 6: Data augmentation. Our positional label works with different image augmentations.

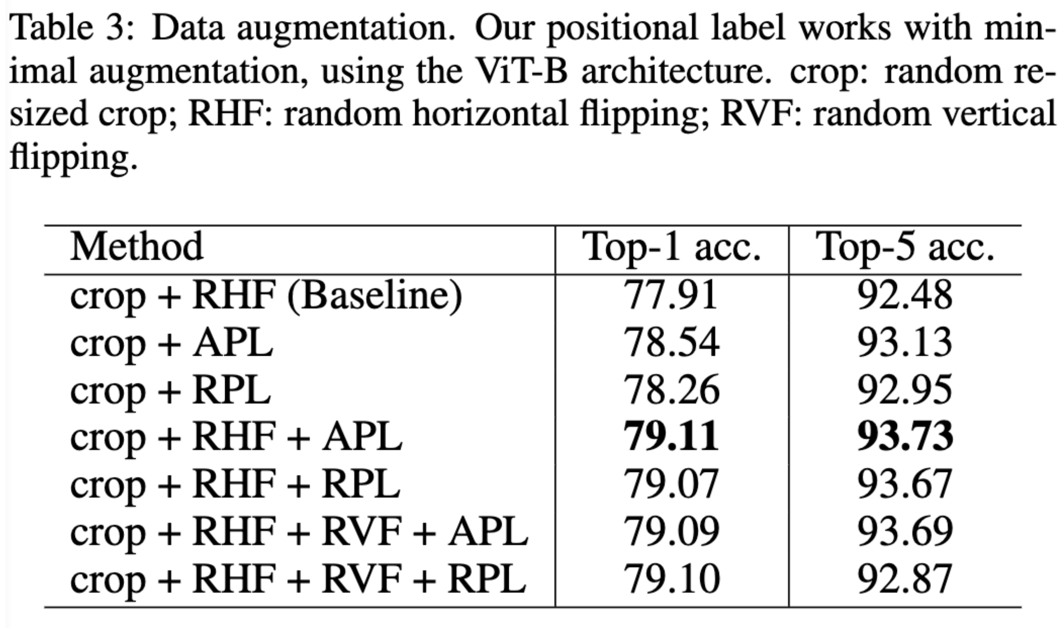


Table 3: Data augmentation. Our positional label works with minimal augmentation, using the ViT-B architecture. crop: random resized crop; RHF: random horizontal flipping; RVF: random vertical flipping.

Method	Top-1 acc.	Top-5 acc.
crop + RHF (Baseline)	77.91	92.48
crop + APL	78.54	93.13
crop + RPL	78.26	92.95
crop + RHF + APL	**79.11**	**93.73**
crop + RHF + RPL	79.07	93.67
crop + RHF + RVF + APL	79.09	93.69
crop + RHF + RVF + RPL	79.10	92.87

Image classification – ImageNet-1K

Table 4: Top-1 and Top-5 accuracies (%) of different ViT backbones, on the ImageNet dataset. Each ViT variant follows the approach of adding positional encoding in its paper.

Method	Top-1 acc.	Top-5 acc.
ViT-B [Dosovitskiy *et al.*, 2020]	77.91	92.48
DeiT-B [Touvron *et al.*, 2021]	81.87	93.92
Swin-B [Liu *et al.*, 2021]	83.35	**96.38**
NesT-B [Zhang *et al.*, 2022]	**83.67**	96.16
ViT-B + APL	79.11	93.73
DeiT-B + APL	82.49	94.34
Swin-B + APL	83.76	96.83
NesT-B + APL	**84.13**	**96.85**
ViT-B + RPL	79.07	93.67
DeiT-B + RPL	82.85	94.21
Swin-B + RPL	**84.09**	96.77
NesT-B + RPL	83.93	**96.81**

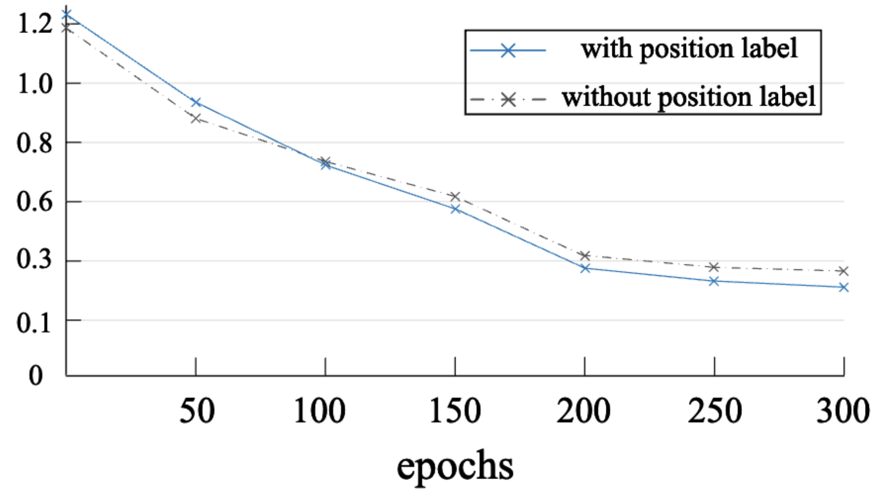


Figure 7: During training, the classification loss of ViT-B with the positional label and ViT-B without the positional label.

Image classification – small datasets

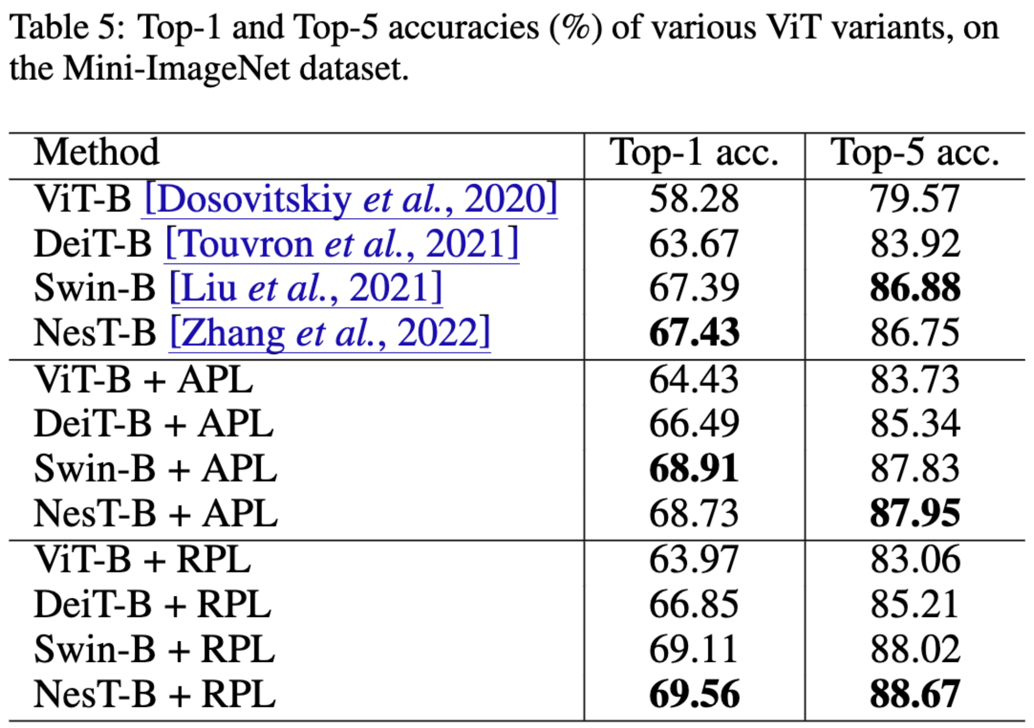


Table 5: Top-1 and Top-5 accuracies (%) of various ViT variants, on the Mini-ImageNet dataset.

Method	Top-1 acc.	Top-5 acc.
ViT-B [Dosovitskiy *et al.*, 2020]	58.28	79.57
DeiT-B [Touvron *et al.*, 2021]	63.67	83.92
Swin-B [Liu *et al.*, 2021]	67.39	**86.88**
NesT-B [Zhang *et al.*, 2022]	**67.43**	86.75
ViT-B + APL	64.43	83.73
DeiT-B + APL	66.49	85.34
Swin-B + APL	**68.91**	87.83
NesT-B + APL	68.73	**87.95**
ViT-B + RPL	63.97	83.06
DeiT-B + RPL	66.85	85.21
Swin-B + RPL	69.11	88.02
NesT-B + RPL	**69.56**	**88.67**

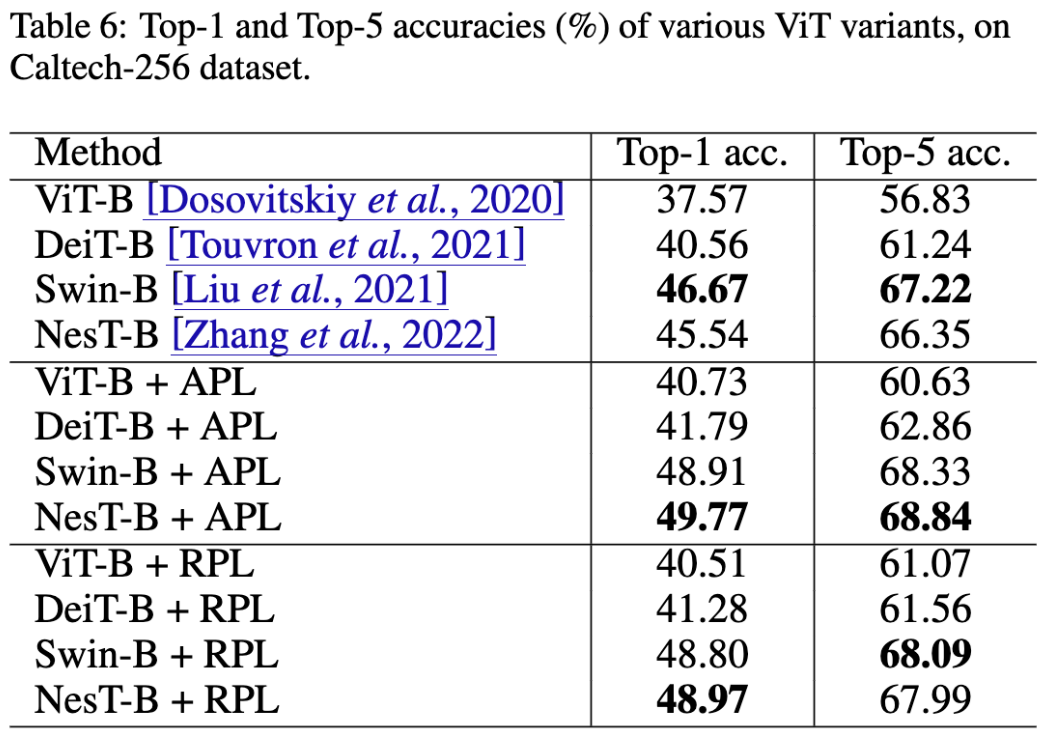


Table 6: Top-1 and Top-5 accuracies (%) of various ViT variants, on Caltech-256 dataset.

Method	Top-1 acc.	Top-5 acc.
ViT-B [Dosovitskiy *et al.*, 2020]	37.57	56.83
DeiT-B [Touvron *et al.*, 2021]	40.56	61.24
Swin-B [Liu *et al.*, 2021]	**46.67**	**67.22**
NesT-B [Zhang *et al.*, 2022]	45.54	66.35
ViT-B + APL	40.73	60.63
DeiT-B + APL	41.79	62.86
Swin-B + APL	48.91	68.33
NesT-B + APL	**49.77**	**68.84**
ViT-B + RPL	40.51	61.07
DeiT-B + RPL	41.28	61.56
Swin-B + RPL	48.80	**68.09**
NesT-B + RPL	**48.97**	67.99

Self-supervised training (MAE)

Table 7: Experimental results of positional label combined with MAE on the ImageNet-1K. Pre-trained for 400 epochs.

Method	Top-1 acc.
ViT-B [Dosovitskiy *et al.*, 2020]	77.91
ViT-B + MAE [He *et al.*, 2021]	79.54
ViT-B + MAE + APL	**80.40**
ViT-B + MAE + RPL	80.21

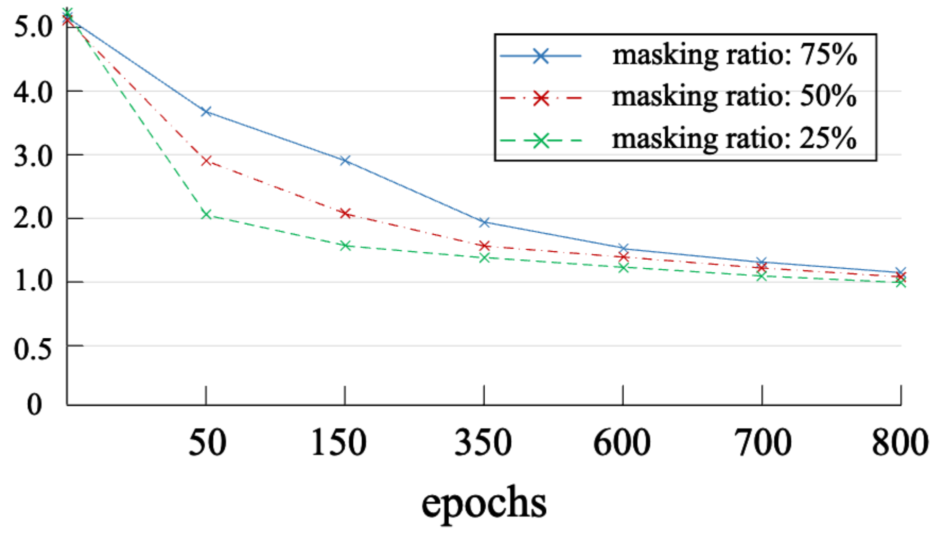


Figure 8: The position loss with different masking ratios.

Conclusions

- Authors have proposed a new method for incorporating positional information into transformer architecture. (Could be adapted for non-CV tasks?)
- The proposed method is (mostly) easily pluggable into existing architectures.
- Positional labels significantly increase the ViT accuracies, according to conducted experiments, even in case of small datasets.
- Positional labels enhance self-supervised training with the use of MAE. Lack of prove that it support different strategies.
- Positional labels seem to not interfere with the standard approach – positional encodings. They can be used together.

Thank You for your attention!

MI

Filip Kołodziejczyk

MI2.AI Seminar, Warsaw, October 21th, 2024