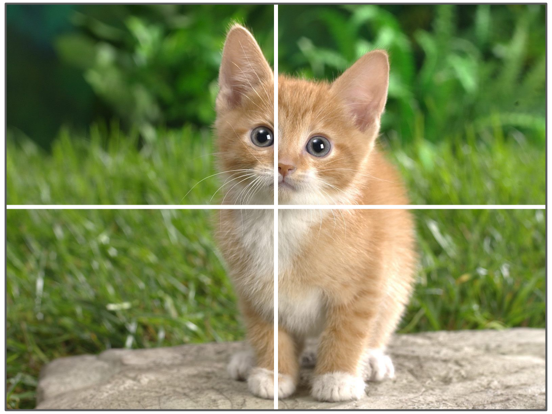
SOTA in Computer Vision

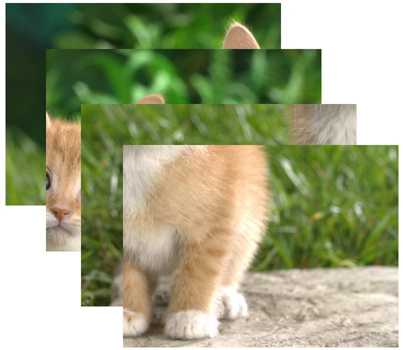
Part 3

Core idea: Tokenized image

Split image to patches and consider them as tokens (just like words in a sentence) to make prediction

Image

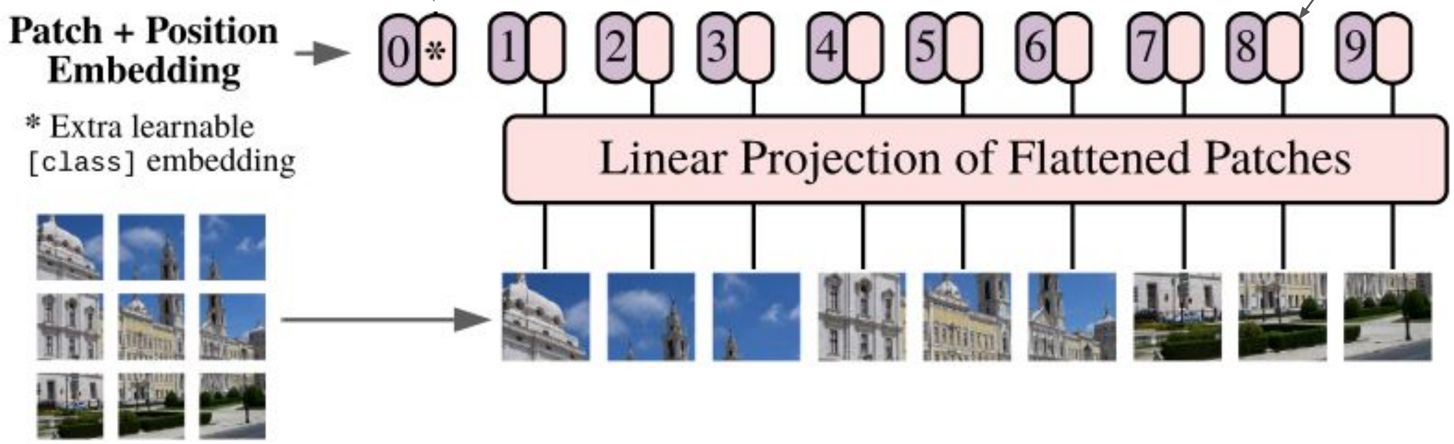

Patches


Tokens

Vision Transformer (ViT)

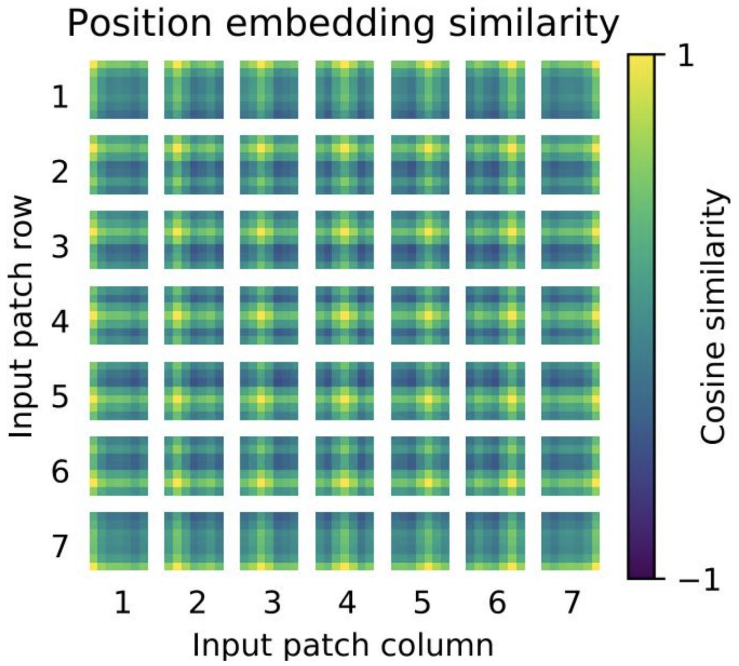
A special learnable [class] token is appended to represent the token of class prediction

Image tokens get flatten and through an embedding


Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

Vision Transformer (ViT)

- Positional embedding can be either learnable parameters or follow original sincos encoding design

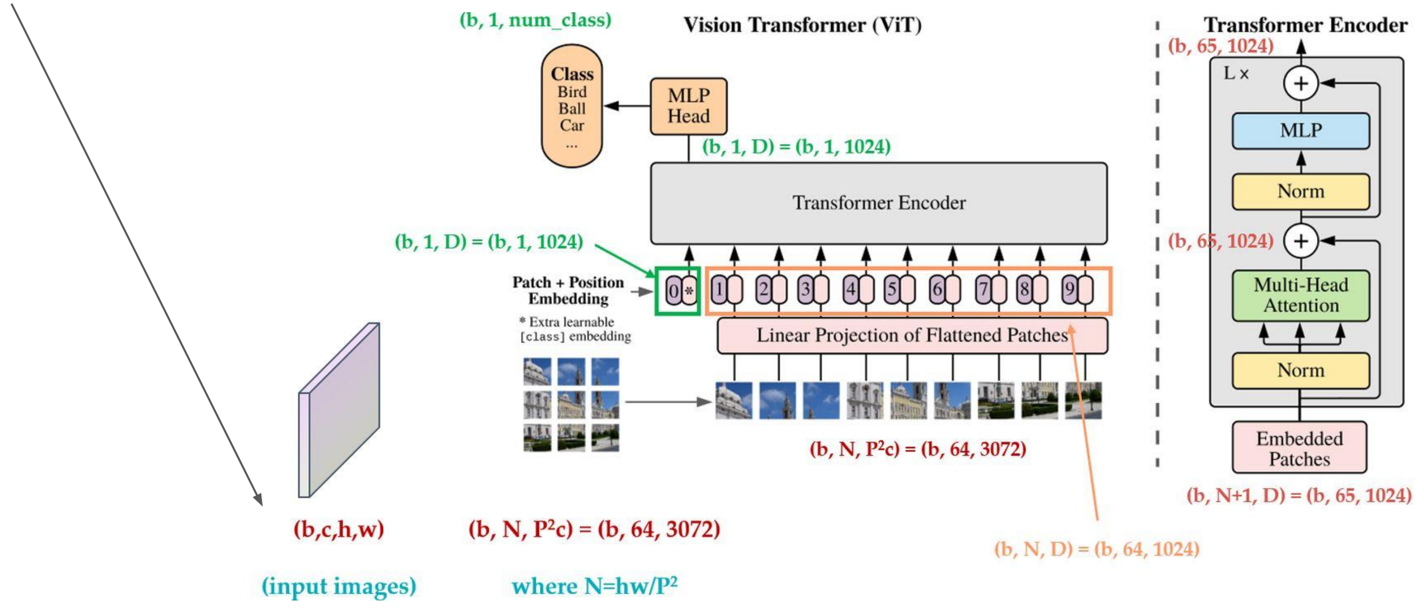


Vision Transformer (ViT)

Then all tokens get through a Transformer encoder.

After that, an extra MLP is applied to the [class] token to get final prediction

$$H \times W \times C \rightarrow N \times (P^2 \cdot C), \text{where } N = HW/P^2$$



Vision Transformer (ViT)

Variants

Model	Layers	Hidden size	MLP size	Heads	Params
ViT-Base (ViT-B)	12	768	3072	12	86M
ViT- Large (ViT-L)	24	1024	4096	16	307M
ViT-Huge (ViT-H)	32	1280	5120	16	632M

Vision Transformer (ViT)

Comparison - BiT v.s ViT

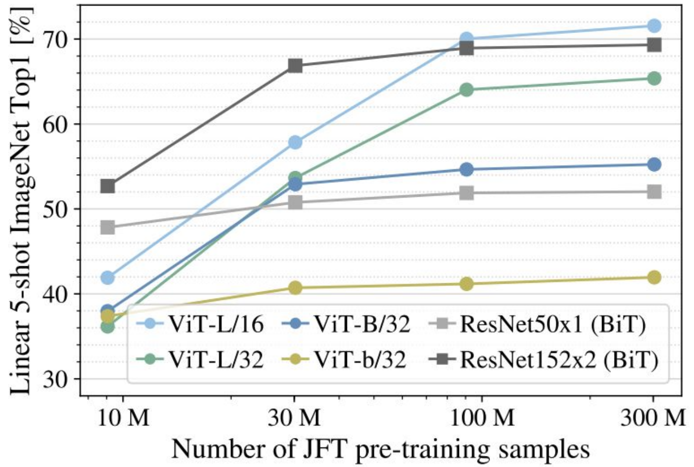
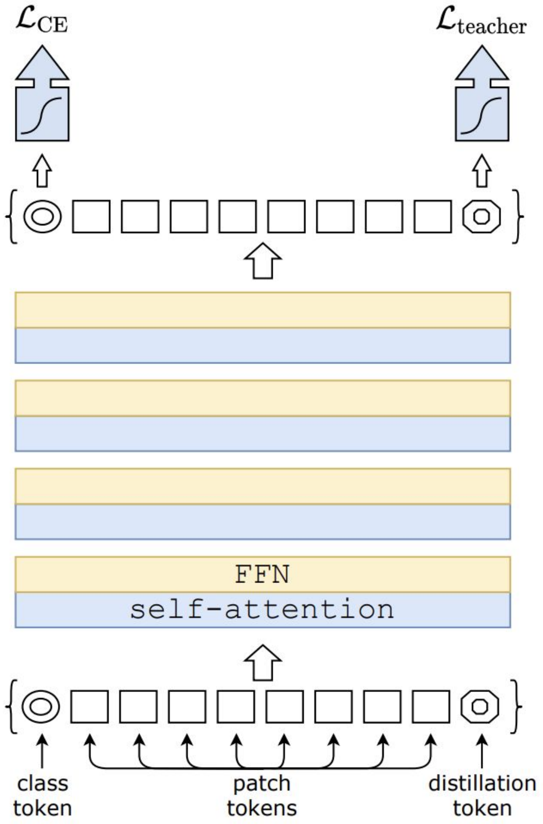


Figure 4: Linear few-shot evaluation on ImageNet versus pre-training size. ResNets perform better with smaller pre-training datasets but plateau sooner than ViT, which performs better with larger pre-training. ViT-b is ViT-B with all hidden dimensions halved.

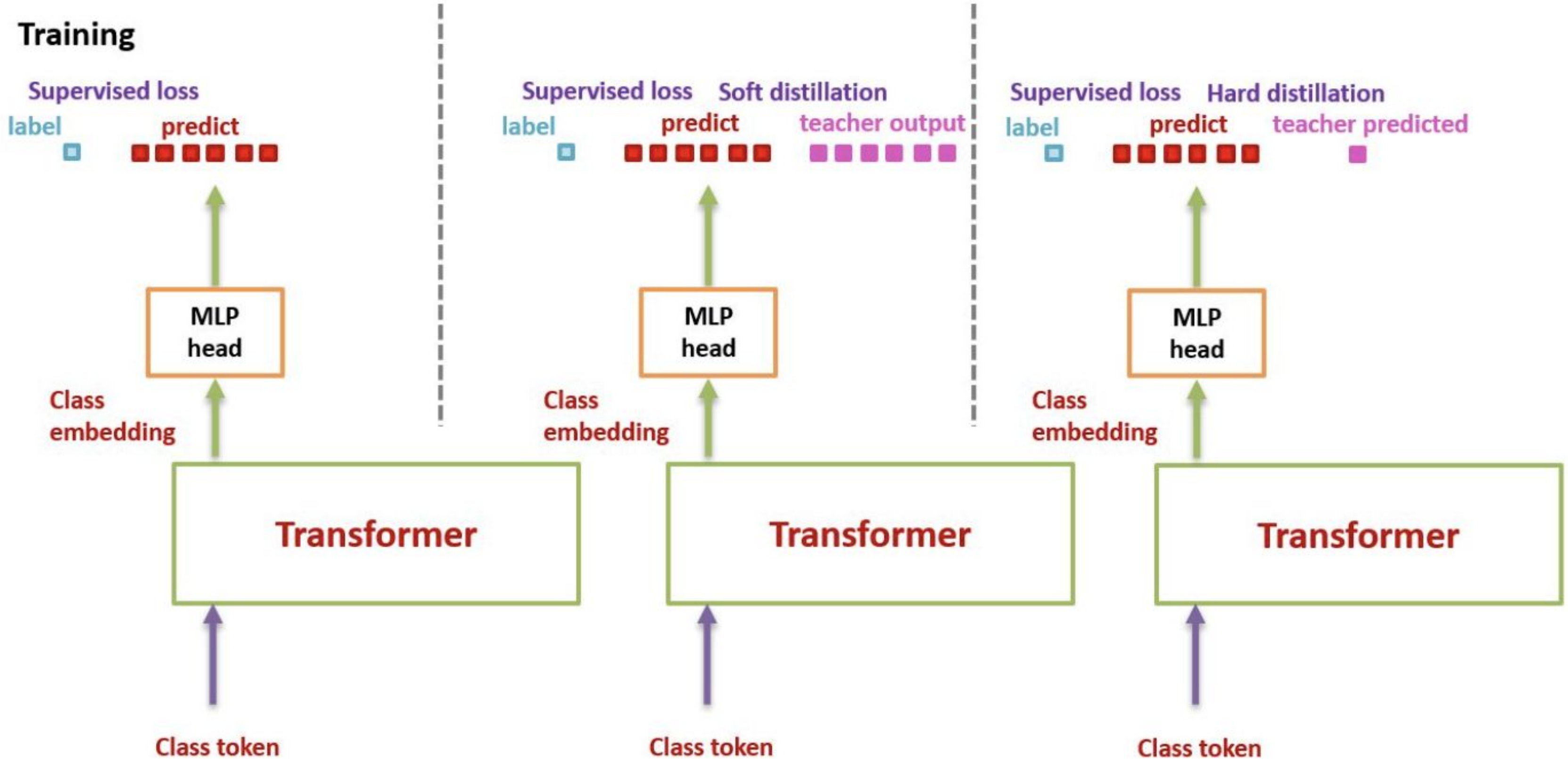
Data-efficient image Transformers(DeiT)

ViT presented excellent results with transformers trained with a large private labelled image dataset (JFT-300M [46], 300 millions images). The paper concluded that transformers *“**do not generalize well when trained on insufficient amounts of data**”*, and the training of these models involved extensive computing resources.

Data-efficient image Transformers(DeiT)



Data-efficient image Transformers(DeiT)



Data-efficient image Transformers(DeiT)

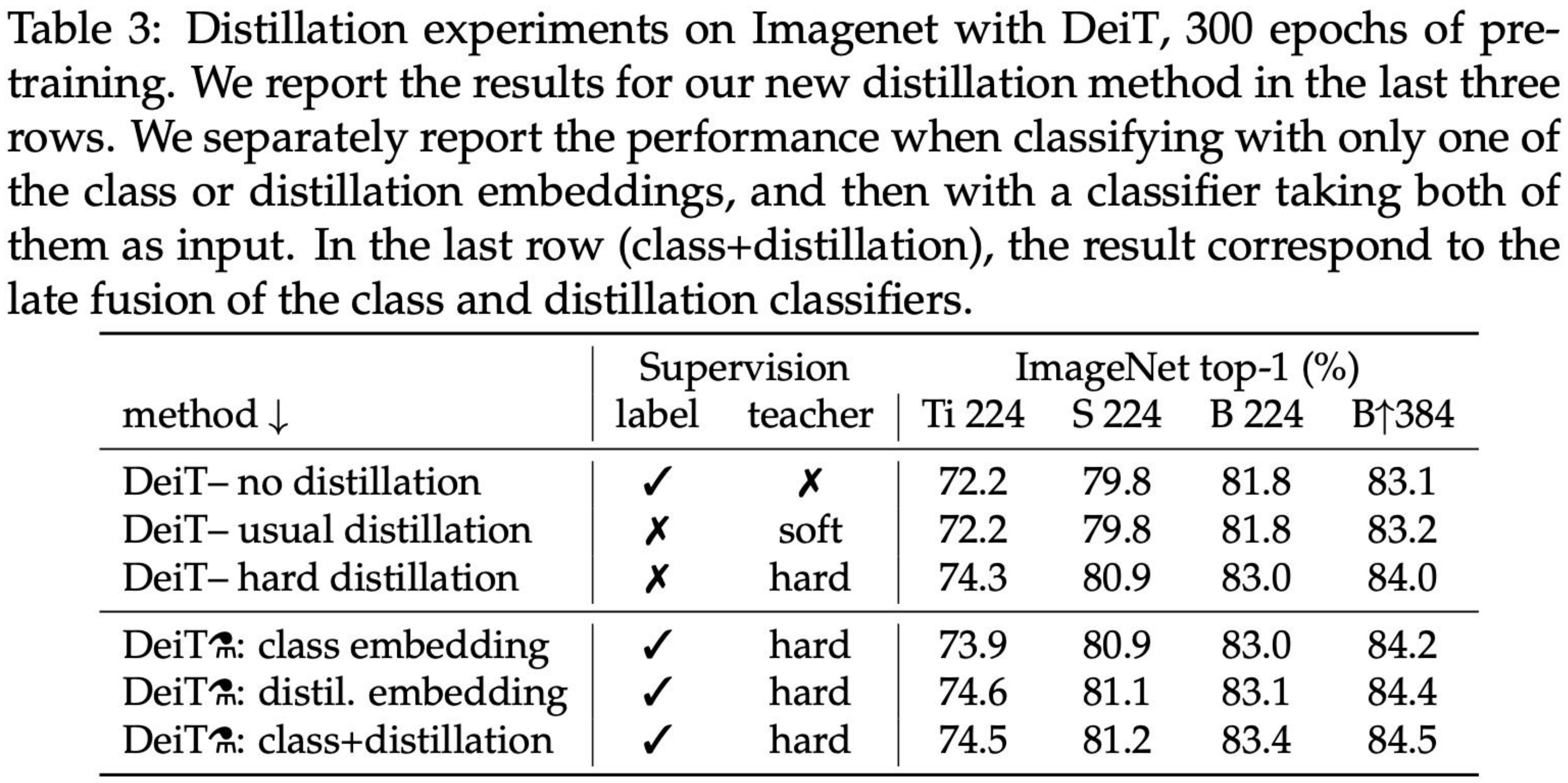
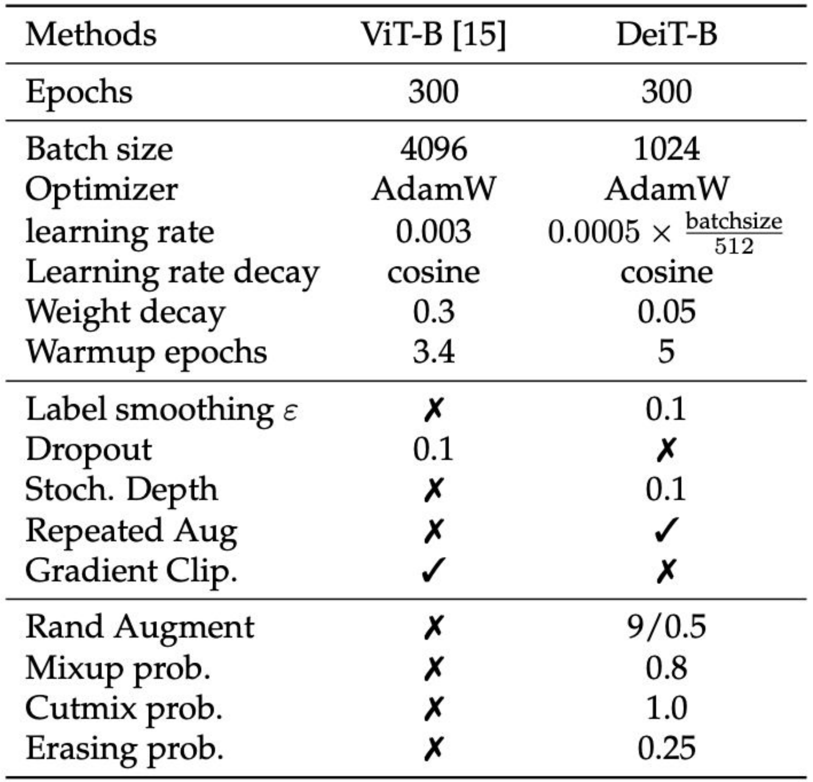


Table 3: Distillation experiments on Imagenet with DeiT, 300 epochs of pre-training. We report the results for our new distillation method in the last three rows. We separately report the performance when classifying with only one of the class or distillation embeddings, and then with a classifier taking both of them as input. In the last row (class+distillation), the result correspond to the late fusion of the class and distillation classifiers.

	Supervision		ImageNet top-1 (%)			
method ↓	label	teacher	Ti 224	S 224	B 224	B↑384
DeiT– no distillation	✓	✗	72.2	79.8	81.8	83.1
DeiT– usual distillation	✗	soft	72.2	79.8	81.8	83.2
DeiT– hard distillation	✗	hard	74.3	80.9	83.0	84.0
DeiT⚗: class embedding	✓	hard	73.9	80.9	83.0	84.2
DeiT⚗: distil. embedding	✓	hard	74.6	81.1	83.1	84.4
DeiT⚗: class+distillation	✓	hard	74.5	81.2	83.4	84.5

Data-efficient image Transformers(DeiT)



Methods	ViT-B [15]	DeiT-B
Epochs	300	300
Batch size	4096	1024
Optimizer	AdamW	AdamW
learning rate	0.003	$0.0005 \times \frac{\text{batchsize}}{512}$
Learning rate decay	cosine	cosine
Weight decay	0.3	0.05
Warmup epochs	3.4	5
Label smoothing ε	✗	0.1
Dropout	0.1	✗
Stoch. Depth	✗	0.1
Repeated Aug	✗	✓
Gradient Clip.	✓	✗
Rand Augment	✗	9/0.5
Mixup prob.	✗	0.8
Cutmix prob.	✗	1.0
Erasing prob.	✗	0.25

Table 9: Ingredients and hyper-parameters for our method and Vit-B.

Data-efficient image Transformers(DeiT)

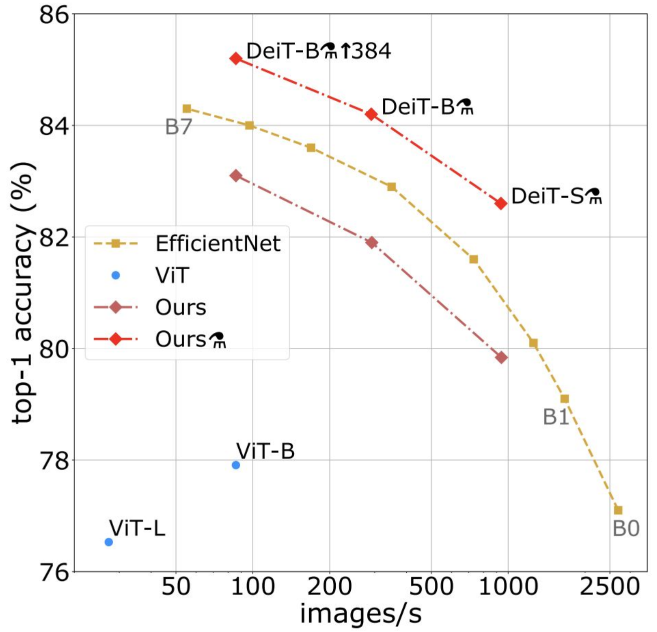
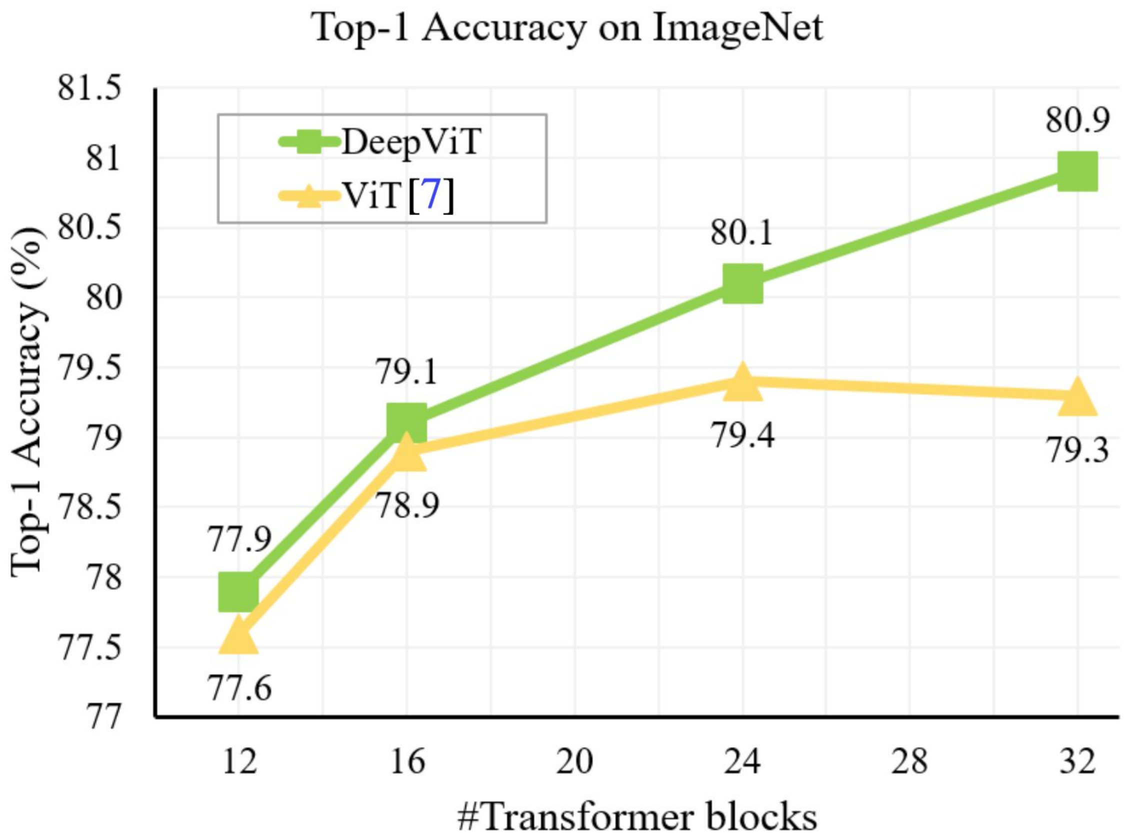


Figure 1: Throughput and accuracy on Imagenet of our methods compared to EfficientNets, trained on Imagenet1k only. The throughput is measured as the number of images processed per second on a V100 GPU. DeiT-B is identical to VIT-B, but the training is more adapted to a data-starving regime. It is learned in a few days on one machine. The symbol ⚗ refers to models trained with our transformer-specific distillation. See Table 5 for details and more models.

Towards Deeper Vision Transformer (DeepViT)

In this paper, we show that, unlike convolution neural networks (CNNs) that can be improved by stacking more convolutional layers, the performance of ViTs saturate fast when scaled to be deeper. More specifically, we empirically observe that such scaling difficulty is caused by the **attention collapse** ***issue***

Towards Deeper Vision Transformer (DeepViT)



Towards Deeper Vision Transformer (DeepViT)

Solution - 1

Increase the **embedding dimension** of each token.

Solution - 2

Re-attention*, to re-generate the attention maps to increase their diversity at different layers with negligible computation and memory cost.*

Towards Deeper Vision Transformer (DeepViT)

Increase the **embedding dimension** of each token.

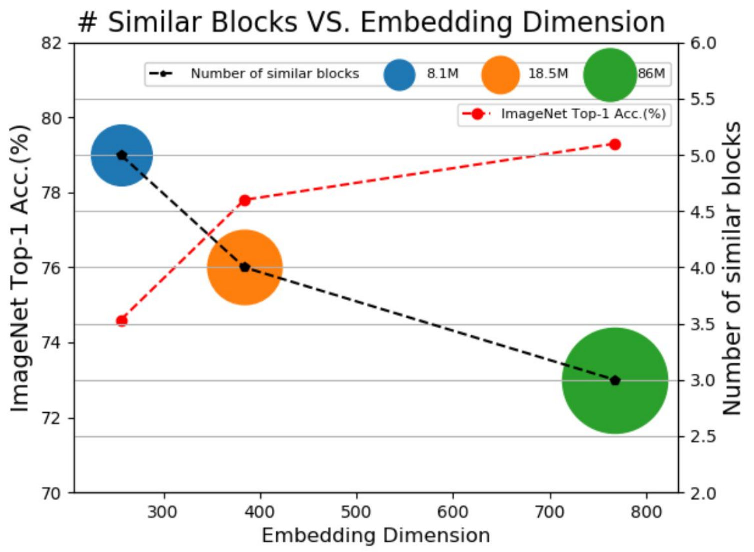


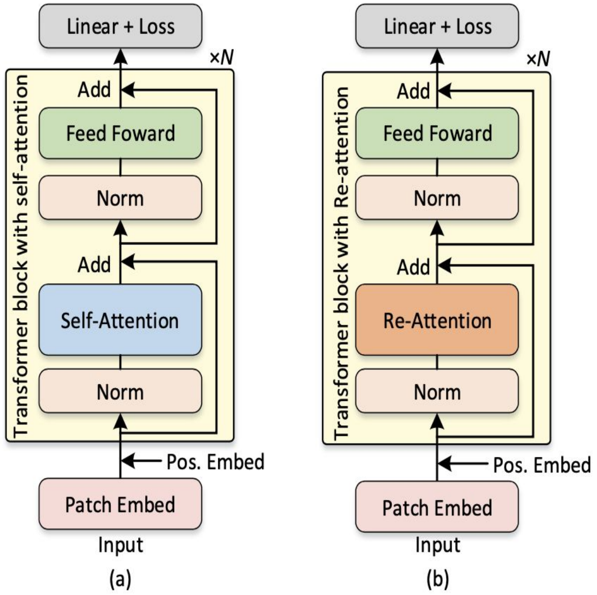
Figure 5: Impacts of embedding dimension on the similarity of generated self-attention map across layers. As can be seen, the number of similar attention maps decreases with increasing embedding dimension. However, the model size also increases rapidly.

Towards Deeper Vision Transformer (DeepViT)

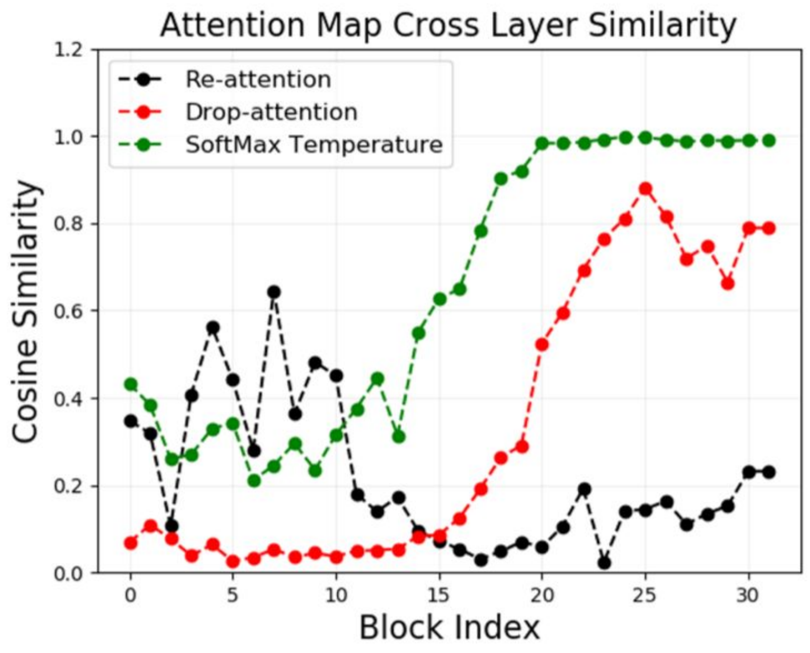
Re-attention

$$\text{Re-Attention}(Q, K, V) = \text{Norm}(\Theta^\top(\text{Softmax}(\frac{QK^\top}{\sqrt{d}})))V, \tag{3}$$

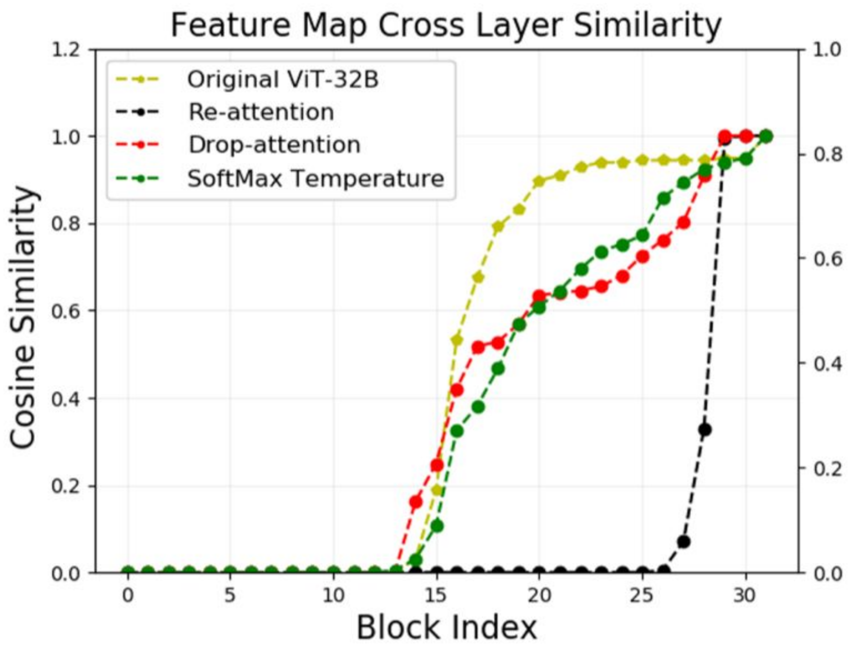
where transformation matrix Θ is multiplied to the self-attention map $\mathbf{A}$ along the head dimension. Here Norm is a normalization function used to reduced the layer-wise variance. Θ is end-to-end learnable.



Towards Deeper Vision Transformer (DeepViT)



(a)



(b)

Class-Attention in Image Transformers (CaiT)

The optimization of image transformers, has been little studied so far. In this work, we build and optimize deeper transformer networks for image classification. In particular, we investigate the ***interplay of architecture and optimization*** *of such dedicated transformers*

Class-Attention in Image Transformers (CaiT)

Two improvements:

- **LayerScale:** Stable optimization for deeper vision transformers
- **Class Attention:** Specialize layers during optimization

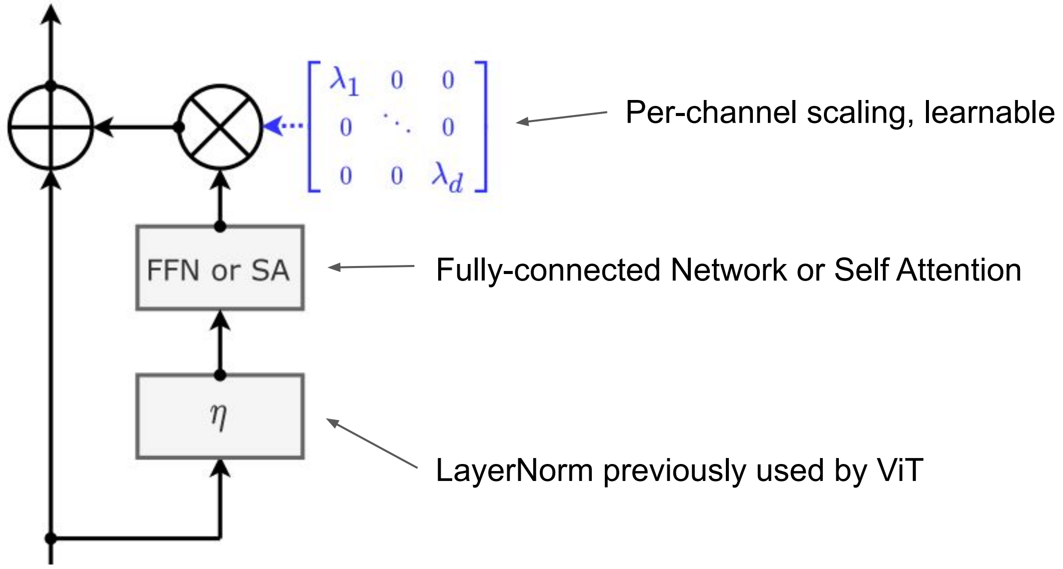
Class-Attention in Image Transformers (CaiT)

Why LayerScale?

By experiments, ViT does not perform well as depth increases.

Class-Attention in Image Transformers (CaiT)

LayerScale: Perform per-channel scaling on residual output. Scale parameters are learnable. Can think of it as a weight normalization method, hence stable the gradient and the training


$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_d \end{bmatrix}$
Per-channel scaling, learnable
FFN or SA
Fully-connected Network or Self Attention
η
LayerNorm previously used by ViT

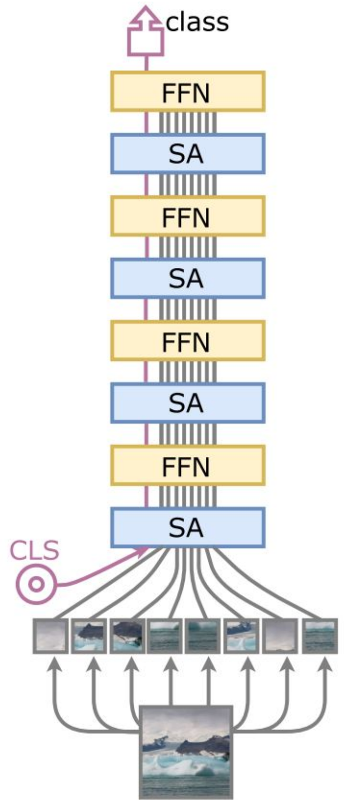
Class-Attention in Image Transformers (CaiT)

Why Class Attention?

As seen in ViT, the [class] token is appended at the start of the Transformer Encoders.

Hence, the learned weights are asked to optimize two contradictory objectives:

1. Guiding the self-attention between patches
2. Summarizing the information useful to the linear classifier

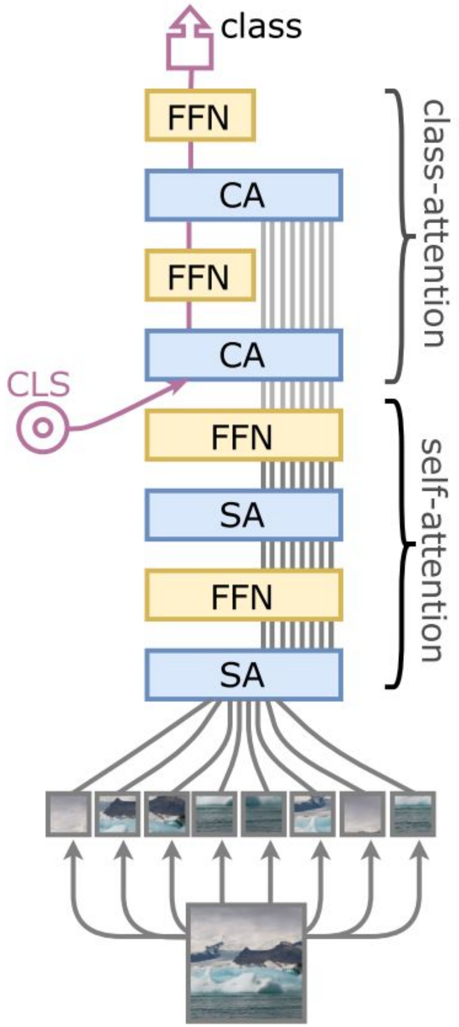


Class-Attention in Image Transformers (

Class Attention

Two stages architecture:

- **1st stage:** Self-attention blocks without [class] token. Prepare the vectors for classifier
- **2nd stage:** [class] token appended. Apply self-attentior but only update the [class] token. Focus on predicting



Class-Attention in Image Transformers (CaiT)

Variants

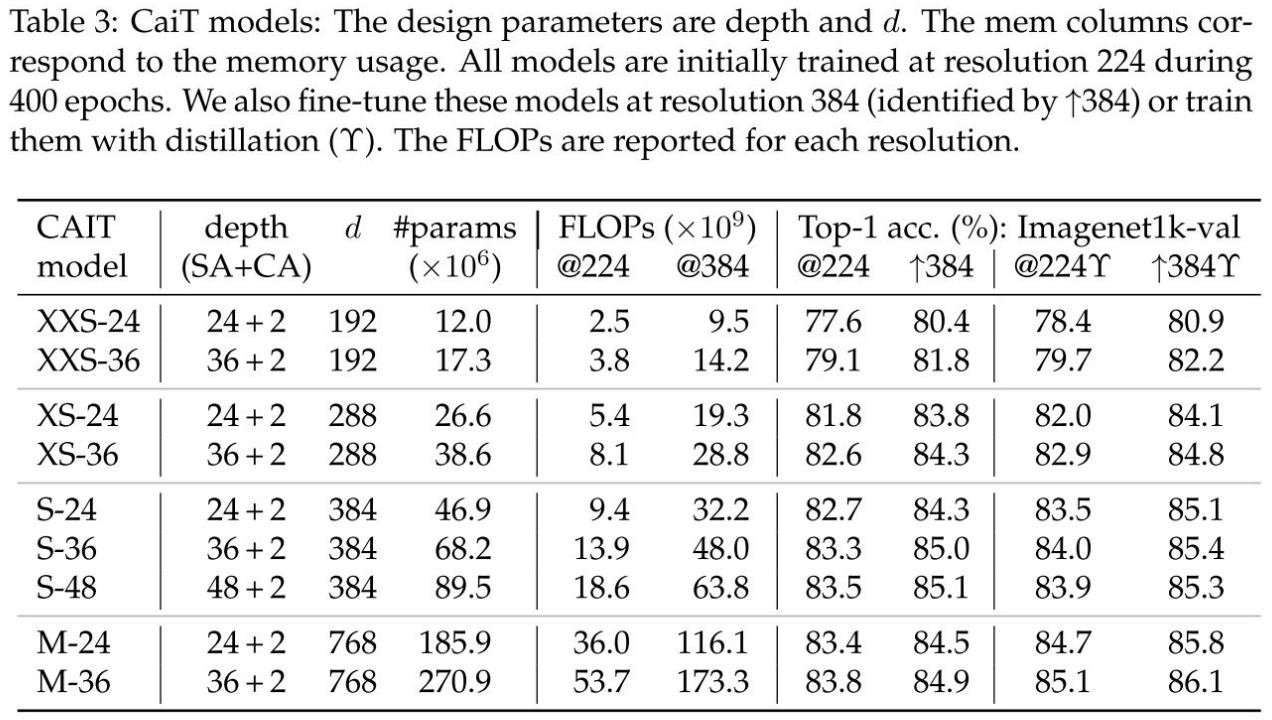


Table 3: CaiT models: The design parameters are depth and d. The mem columns correspond to the memory usage. All models are initially trained at resolution 224 during 400 epochs. We also fine-tune these models at resolution 384 (identified by $\uparrow$384) or train them with distillation (Υ). The FLOPs are reported for each resolution.

CAIT	depth	d	#params	FLOPs ($\times 10^9$)		Top-1 acc. (%): Imagenet1k-val			
model	(SA+CA)		($\times 10^6$)	@224	@384	@224	$\uparrow$384	@224Υ	$\uparrow$384Υ
XXS-24	24 + 2	192	12.0	2.5	9.5	77.6	80.4	78.4	80.9
XXS-36	36 + 2	192	17.3	3.8	14.2	79.1	81.8	79.7	82.2
XS-24	24 + 2	288	26.6	5.4	19.3	81.8	83.8	82.0	84.1
XS-36	36 + 2	288	38.6	8.1	28.8	82.6	84.3	82.9	84.8
S-24	24 + 2	384	46.9	9.4	32.2	82.7	84.3	83.5	85.1
S-36	36 + 2	384	68.2	13.9	48.0	83.3	85.0	84.0	85.4
S-48	48 + 2	384	89.5	18.6	63.8	83.5	85.1	83.9	85.3
M-24	24 + 2	768	185.9	36.0	116.1	83.4	84.5	84.7	85.8
M-36	36 + 2	768	270.9	53.7	173.3	83.8	84.9	85.1	86.1

Incorporating Convolution Designs into Visual Transformers (CeiT)

*In DeiT, **a CNN teacher gives better performance than using a Transformer one**, which probably due to “the inductive bias inherited by the Transformer through distillation”. These observations make us rethink **whether it is appropriate to remove all convolutions** from the Transformer. And should the inductive biases inherited in the convolution be forgotten?*

Incorporating Convolution Designs into Visual Transformers (CeiT)

Pure Transformer architectures often require a large amount of training data or extra supervision to obtain comparable performance with convolutional neural networks (CNNs). We propose a new Convolution-enhanced image Transformer (CeiT) which ***combines the advantages of CNNs in extracting low-level features****, strengthening locality, and the advantages of* ***Transformers in establishing long-range dependencies****.*

Incorporating Convolution Designs into Visual Transformers (CeiT)

Solution

1) Instead of the straightforward tokenization from raw in- put images, we design an Image-to-Tokens (I2T) module that extracts patches from generated low-level features

2) The feed-froward network in each encoder block is replaced with a Locally-enhanced Feed-Forward (LeFF) layer that promotes the correlation among neighboring tokens in the spatial dimension

3) A Layer-wise Class token Attention (LCA) is attached at the top of the Transformer that utilizes the multi-level representations.

Incorporating Convolution Designs into Visual Transformers (CeiT)

Image-to-Tokens(I2T)

$$\mathbf{x}' = \mathrm{I2T}(\mathbf{x}) = \mathrm{MaxPool}(\mathrm{BN}(\mathrm{Conv}(\mathbf{x})))$$

where $\mathbf{x}' \in \mathbb{R}^{\frac{H}{S} \times \frac{W}{S} \times D}$, S is the stride w.r.t the raw input images, and D is the number of enriched channels.

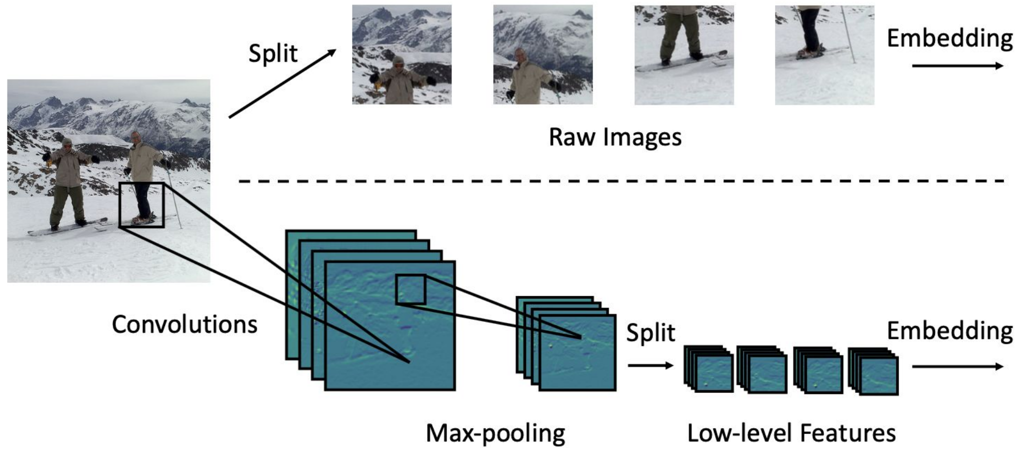


Figure 2. Comparisons of different tokenization methods. The upper one extracts patches from raw input images. The below one (I2T) uses the low-level features generated by a convolutional stem.

Incorporating Convolution Designs into Visual Transformers (CeiT)

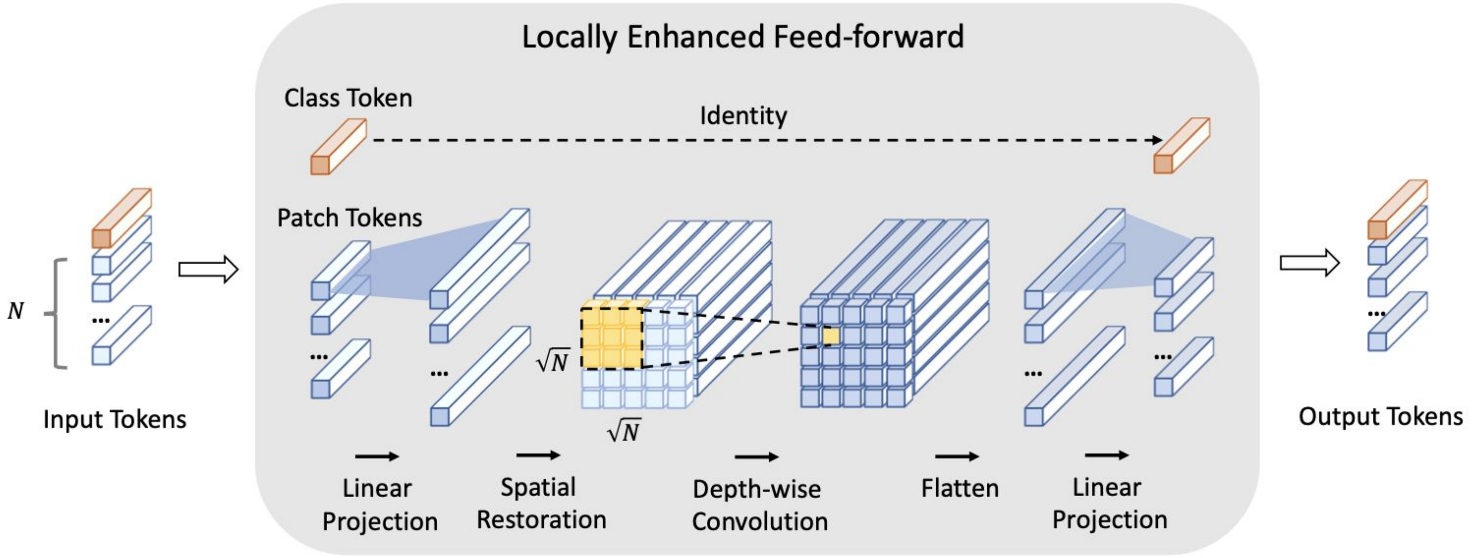


Figure 3. Illustration of the Locally-enhanced Feed-Forward module. First, patch tokens are projected into a higher dimension. Second, they are restored to "images" in the spatial dimension based on the original positions. Third, a depth-wise convolution is performed on the restored tokens as shown in the yellow region. Then the patch tokens are flattened and projected to the initial dimension. Besides, the class token conducts an identical mapping.

Incorporating Convolution Designs into Visual Transformers (CeiT)

Locally-enhanced Feed-Forward (LeFF) layer

$$\mathbf{x}_c^h, \mathbf{x}_p^h = \mathrm{Split}(\mathbf{x}_t^h) \tag{5}$$

$$\mathbf{x}_p^{l_1} = \mathrm{GELU}(\mathrm{BN}(\mathrm{Linear1}(\mathbf{x}_p^h))) \tag{6}$$

$$\mathbf{x}_p^s = \mathrm{SpatialRestore}(\mathbf{x}_p^{l_1}) \tag{7}$$

$$\mathbf{x}_p^d = \mathrm{GELU}(\mathrm{BN}(\mathrm{DWConv}(\mathbf{x}_p^s))) \tag{8}$$

$$\mathbf{x}_p^f = \mathrm{Flatten}(\mathbf{x}_p^d) \tag{9}$$

$$\mathbf{x}_p^{l_2} = \mathrm{GELU}(\mathrm{BN}(\mathrm{Linear2}(\mathbf{x}_p^f))) \tag{10}$$

$$\mathbf{x}_t^{h+1} = \mathrm{Concat}(\mathbf{x}_c^h, \mathbf{x}_p^{l_2}) \tag{11}$$

Incorporating Convolution Designs into Visual Transformers (CeiT)

Layer-wise Class token Attention (LCA)

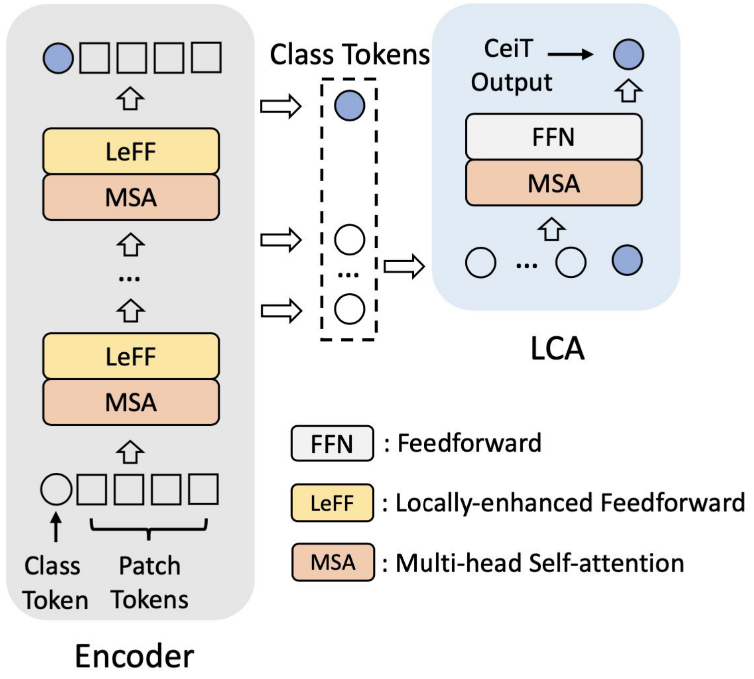


Figure 4. The proposed Layer-wise Class-token Attention block. It integrates information across different layers through receiving a sequence of class tokens as inputs.

Incorporating Convolution Designs into Visual Transformers (CeiT)

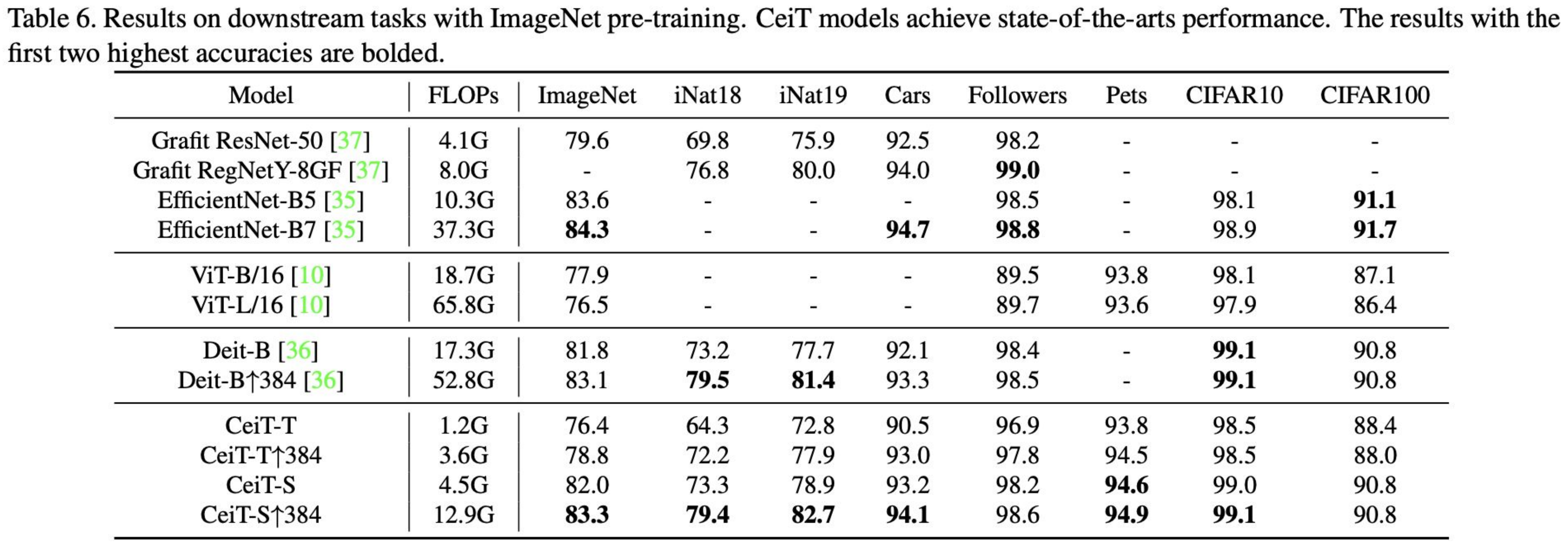


Table 6. Results on downstream tasks with ImageNet pre-training. CeiT models achieve state-of-the-arts performance. The results with the first two highest accuracies are bolded.

Model	FLOPs	ImageNet	iNat18	iNat19	Cars	Followers	Pets	CIFAR10	CIFAR100
Grafit ResNet-50 [37]	4.1G	79.6	69.8	75.9	92.5	98.2	-	-	-
Grafit RegNetY-8GF [37]	8.0G	-	76.8	80.0	94.0	**99.0**	-	-	-
EfficientNet-B5 [35]	10.3G	83.6	-	-	-	98.5	-	98.1	**91.1**
EfficientNet-B7 [35]	37.3G	**84.3**	-	-	**94.7**	**98.8**	-	98.9	**91.7**
ViT-B/16 [10]	18.7G	77.9	-	-	-	89.5	93.8	98.1	87.1
ViT-L/16 [10]	65.8G	76.5	-	-	-	89.7	93.6	97.9	86.4
Deit-B [36]	17.3G	81.8	73.2	77.7	92.1	98.4	-	**99.1**	90.8
Deit-B↑384 [36]	52.8G	83.1	**79.5**	**81.4**	93.3	98.5	-	**99.1**	90.8
CeiT-T	1.2G	76.4	64.3	72.8	90.5	96.9	93.8	98.5	88.4
CeiT-T↑384	3.6G	78.8	72.2	77.9	93.0	97.8	94.5	98.5	88.0
CeiT-S	4.5G	82.0	73.3	78.9	93.2	98.2	**94.6**	99.0	90.8
CeiT-S↑384	12.9G	**83.3**	**79.4**	**82.7**	**94.1**	98.6	**94.9**	**99.1**	90.8

Reference

Yuan, L., Hou, Q., Jiang, Z., Feng, J., & Yan, S. (2021). Volo: Vision outlooker for visual recognition. *arXiv preprint arXiv:2106.13112*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Zhou, D., Kang, B., Jin, X., Yang, L., Lian, X., Jiang, Z., ... & Feng, J. (2021). Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*.

Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., & Jégou, H. (2021). Going deeper with image transformers. arXiv preprint *arXiv:2103.17239.*

Yuan, K., Guo, S., Liu, Z., Zhou, A., Yu, F., & Wu, W. (2021). Incorporating convolution designs into visual transformers. *arXiv preprint arXiv:2103.11816*.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021, July). Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning* (pp. 10347-10357). PMLR.

https://zhuanlan.zhihu.com/p/348593638

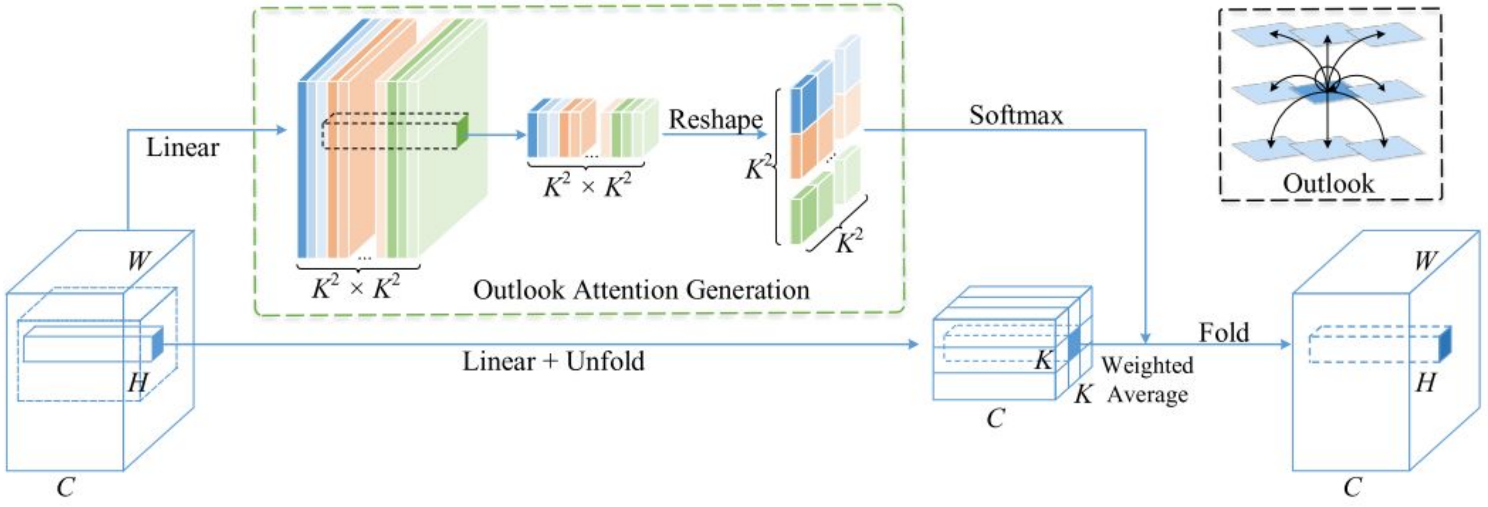
Appendix : Vision Outlooker (VOLO)

Why Vision Outlooker?

Authors argue that ViT cannot out-performed CNN due to low efficacy in encoding fine-level features and context in token representation

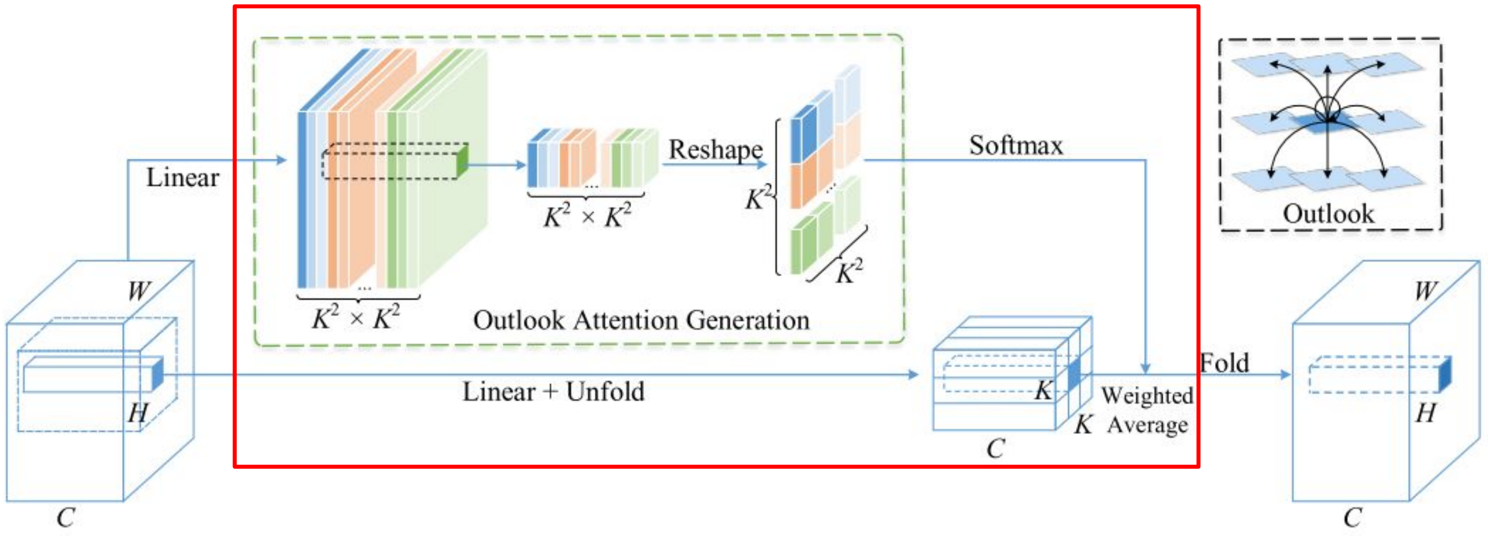
Vision Outlooker (VOLO)

Outlooker use information in K x K window to generate its representation



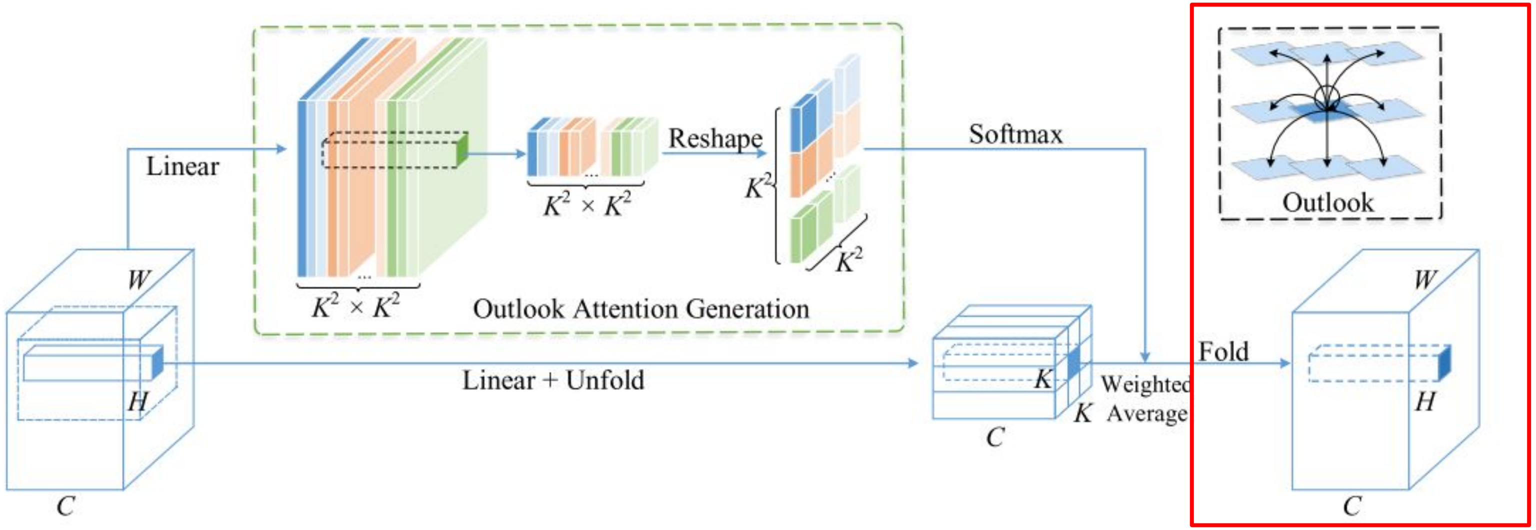
Vision Outlooker (VOLO)

For each position (token), generate a K^2 x K^2 attention map, meaning an K x K attention map for each neighbor. Then calculate weighted average value. Finally, we got K x K x C tensor



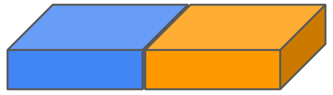
Vision Outlooker (VOLO)

But each neighbor also has its own KxKxC tensors, these tensors also have the weighted average value generated by the neighbor. So we have to sum all the values from the neighbors according to the relative position. This is called **folding**

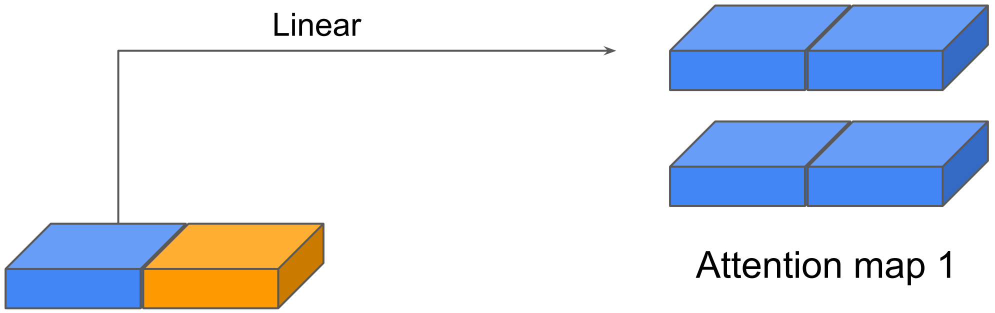


Outlooker illustration

Let’s take a simple “image” size 1x2

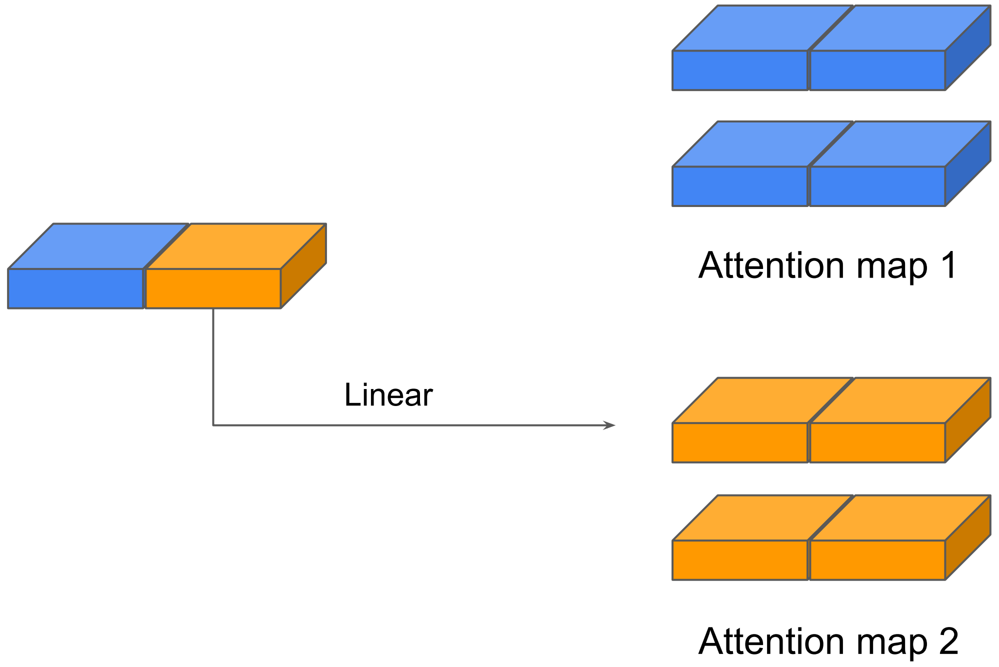


Attention map generation



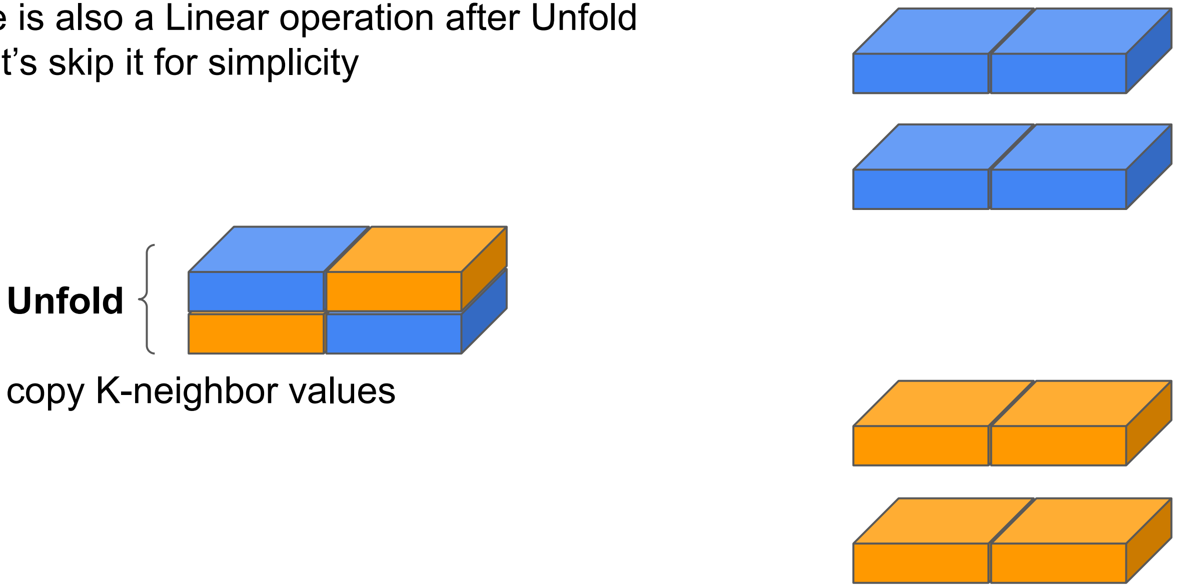
Attention map 1

Attention map generation

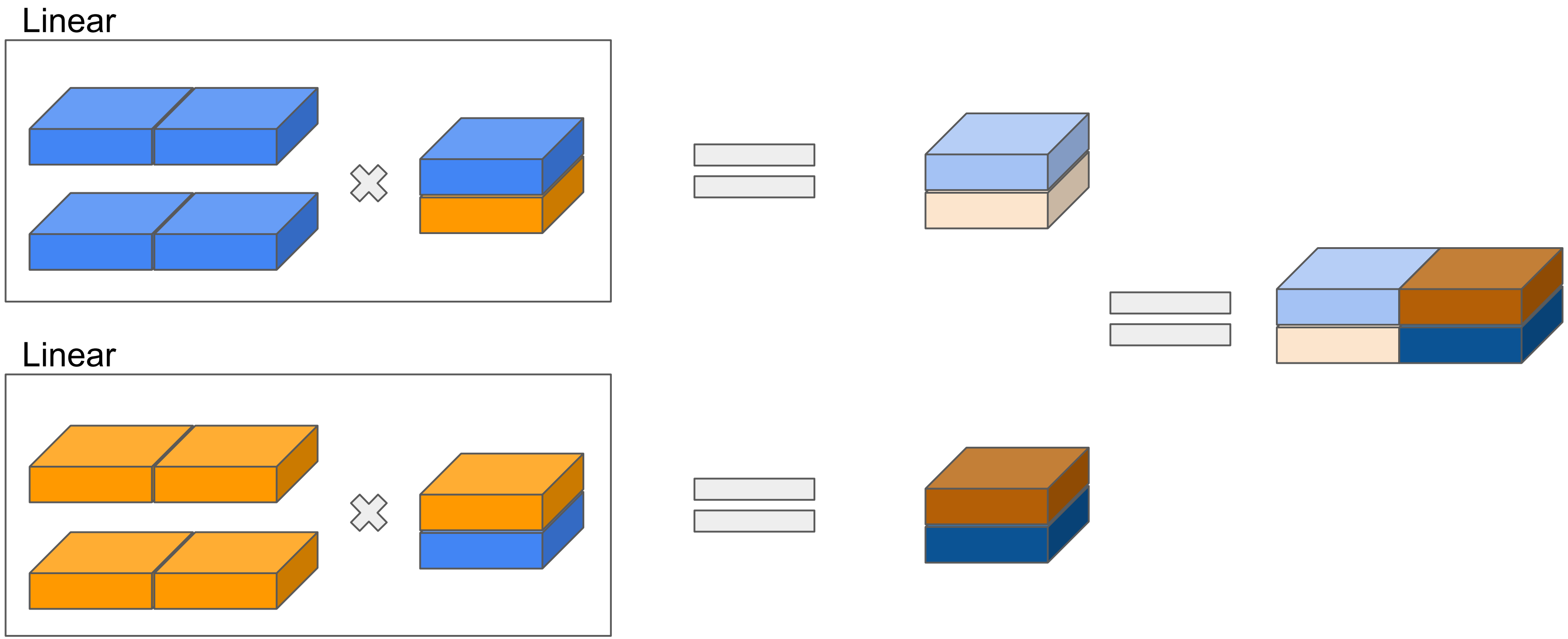


Unfold

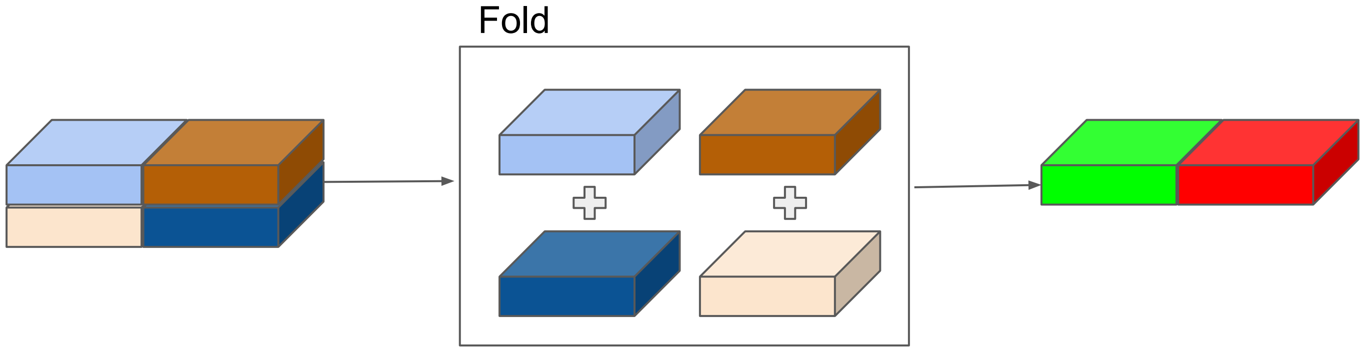
There is also a Linear operation after Unfold
but let's skip it for simplicity



Projection



Fold for final result



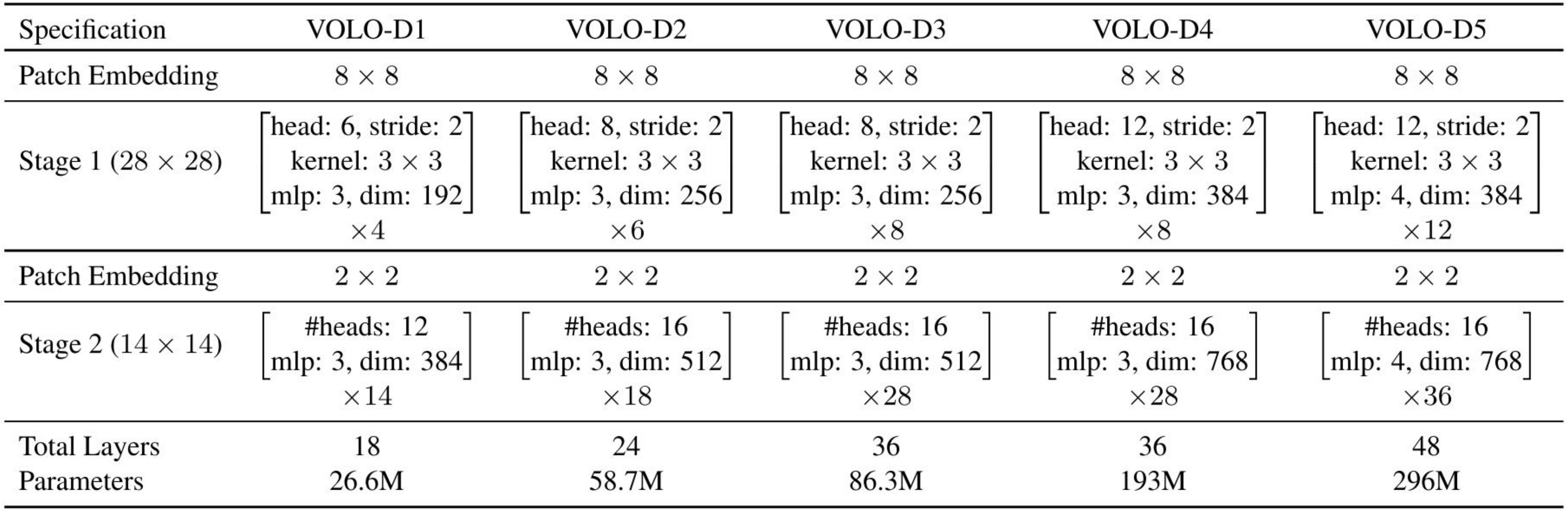
Vision Outlooker (VOLO)

Network Architecture:

- 1st stage: Stack of Outlook Attentions for patch tokens
- 2nd stage: Stack of Self-Attention

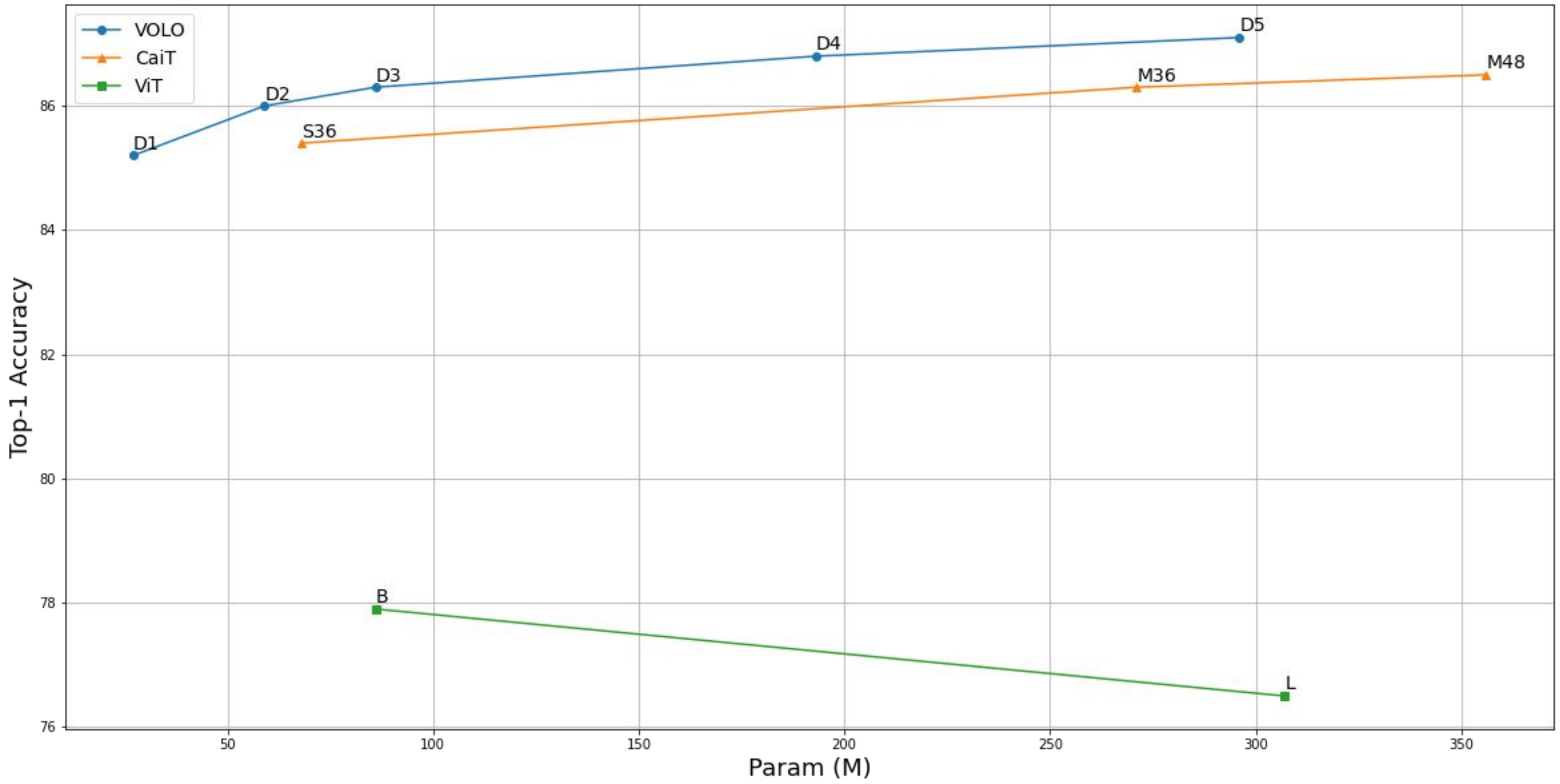
Vision Outlooker (VOLO)

Variants:



Specification	VOLO-D1	VOLO-D2	VOLO-D3	VOLO-D4	VOLO-D5
Patch Embedding	8×8	8×8	8×8	8×8	8×8
Stage 1 (28×28)	$\begin{bmatrix}\text{head: 6, stride: 2}\\ \text{kernel: } 3 \times 3\\ \text{mlp: 3, dim: 192}\end{bmatrix} \times 4$	$\begin{bmatrix}\text{head: 8, stride: 2}\\ \text{kernel: } 3 \times 3\\ \text{mlp: 3, dim: 256}\end{bmatrix} \times 6$	$\begin{bmatrix}\text{head: 8, stride: 2}\\ \text{kernel: } 3 \times 3\\ \text{mlp: 3, dim: 256}\end{bmatrix} \times 8$	$\begin{bmatrix}\text{head: 12, stride: 2}\\ \text{kernel: } 3 \times 3\\ \text{mlp: 3, dim: 384}\end{bmatrix} \times 8$	$\begin{bmatrix}\text{head: 12, stride: 2}\\ \text{kernel: } 3 \times 3\\ \text{mlp: 4, dim: 384}\end{bmatrix} \times 12$
Patch Embedding	2×2	2×2	2×2	2×2	2×2
Stage 2 (14×14)	$\begin{bmatrix}\text{\#heads: 12}\\ \text{mlp: 3, dim: 384}\end{bmatrix} \times 14$	$\begin{bmatrix}\text{\#heads: 16}\\ \text{mlp: 3, dim: 512}\end{bmatrix} \times 18$	$\begin{bmatrix}\text{\#heads: 16}\\ \text{mlp: 3, dim: 512}\end{bmatrix} \times 28$	$\begin{bmatrix}\text{\#heads: 16}\\ \text{mlp: 3, dim: 768}\end{bmatrix} \times 28$	$\begin{bmatrix}\text{\#heads: 16}\\ \text{mlp: 4, dim: 768}\end{bmatrix} \times 36$
Total Layers	18	24	36	36	48
Parameters	26.6M	58.7M	86.3M	193M	296M

Comparison - Training ImageNet from scratch


VOLO
CaiT
ViT
D1
D2
D3
D4
D5
S36
M36
M48
B
L
Top-1 Accuracy
Param (M)