VOLO: Vision Outlooker for Visual Recognition

Li Yuan[1,2*] Qibin Hou[2*] Zihang Jiang[2] Jiashi Feng[1,2] Shuicheng Yan[1]
[1]Sea AI Lab [2]National University of Singapore
{ylustcnus,andrewhoux,jzh0103}@gmail.com, {fengjs, yansc}@sea.com

Abstract

Visual recognition has been dominated by convolutional neural networks (CNNs) for years. Though recently the prevailing vision transformers (ViTs) have shown great potential of self-attention based models in ImageNet classification, their performance is still inferior to that of the latest SOTA CNNs if no extra data are provided. In this work, we try to close the performance gap and demonstrate that attention-based models are indeed able to outperform CNNs. We find a major factor limiting the performance of ViTs for ImageNet classification is their low efficacy in encoding fine-level features into the token representations. To resolve this, we introduce a novel outlook attention and present a simple and general architecture, termed Vision Outlooker (VOLO). Unlike self-attention that focuses on global dependency modeling at a coarse level, the outlook attention efficiently encodes finer-level features and contexts into tokens, which is shown to be critically beneficial to recognition performance but largely ignored by the self-attention. Experiments show that our VOLO achieves 87.1% top-1 accuracy on ImageNet-1K classification, which is the first model exceeding 87% accuracy on this competitive benchmark, without using any extra training data. In addition, the pre-trained VOLO transfers well to downstream tasks, such as semantic segmentation. We achieve 84.3% mIoU score on the cityscapes validation set and 54.3% on the ADE20K validation set. Code is available at https://github.com/sail-sg/volo.

1. Introduction

Modeling in visual recognition, which was long dominated by convolutional neural networks (CNNs), has recently been revolutionized by Vision Transformers (ViTs) [14, 51, 68]. Different from CNNs that aggregate and transform features via local and dense convolutional kernels, ViTs directly model long-range dependencies of local patches (*a.k.a.* tokens) through the self-attention mechanism which is with greater flexibility in modeling visual contents. Despite the remarkable effectiveness on visual recognition [37, 32, 52, 79], the performance of ViT models still lags behind that of the state-of-the-art CNN models. For instance, as shown in Table 1, the state-of-the-art transformer-based CaiT [52] attains 86.5% top-1 accuracy on ImageNet, which however is still 0.3% lower compared with the 86.8% top-1 accuracy achieved by the CNN-based NFNet-F5 [2] with SAM and augmult [15, 16].

In this work we try to close such performance gap. We find one major factor limiting ViTs from outperforming CNNs is their low efficacy in encoding fine-level features and contexts into token representations, which are critical for achieving compelling visual recognition performance. Fine-level information can be encoded into tokens by finer-grained image tokenization, which however would lead to a token sequence of greater length that increases quadratically the complexity of the self-attention mechanism of ViTs.

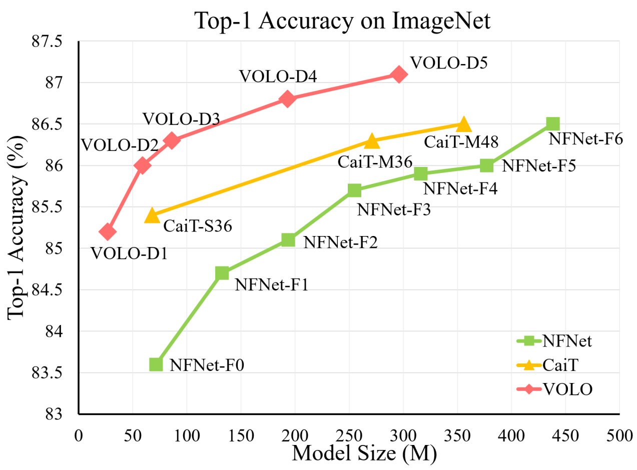

Figure 1. ImageNet top-1 accuracy of state-of-the-art CNN-based and Transformer-based models. All the results are obtained based on the best test resolutions, without using any extra training data. Our VOLO-D5 achieves the best accuracy, outperforming the latest NFNet-F6 w/ SAM [2, 15] and CaiT-M48 w/ KD [22, 69], while using much less training parameters. To our best knowledge, VOLO-D5 is the first model exceeding 87% top-1 accuracy on ImageNet.

[*]Equal contribution.

Table 1. Comparison with previous state-of-the-art classification models, most of which have once achieved leading positions on the leaderboard of PaperWithCode[2](w/o extra data).

Settings	LV-ViT [32]	CaiT [52]	NFNet-F6 [2]	NFNet-F5 [2]	VOLO-D5 (Ours)
Test Resolution	448×448	448×448	576×576	544×544	448×448 / 512×512
Model Size	140M	356M	438M	377M	296M
Computations	157B	330B	377B	290B	304B / 412B
Architecture	Vision Transformer	Vision Transformer	Convolutions	Convolutions	VOLO
Extra Augmentations	Token Labeling [32]	Knowledge Distill	SAM [15]	SAM + augmult [15, 16]	Token Labeling [32]
ImageNet Top-1 Acc.	86.4	86.5	86.5	86.8	**87.0 / 87.1**

In this work, we present a new simple and light-weight attention mechanism, termed *Outlooker*, to enrich the token representations with fine level information efficiently. The proposed Outlooker innovates the way of generating attention for token aggregation, and enables the model to efficiently encode fine-level information. In particular, it extrapolates the mechanism of aggregating surrounding tokens from the anchor token feature directly via efficient linear projections, thus getting rid of the expensive dot-product attention computation.

Based on the proposed Outlooker, we present VOLO, a simple yet powerful model architecture for visual recognition. VOLO achieves fine-level token representation encoding and global information aggregation with a two-stage architecture design. Specifically, given an input image of size 224×224, before using self-attention to build global dependencies at the coarse level (*e.g.*, 14×14), the VOLO tokenizes the image on smaller-size patches (*e.g.*, 8×8) and employs multiple Outlookers to encode token representations at the fine level (*e.g.*, 28×28). The obtained token representations are more expressive, thus significantly improving the model performance in image classification.

Experiments show that our proposed VOLO performs extremely well in ImageNet classification. Take a VOLO model with 26.6M learnable parameters as an example. It achieves 84.2% top-1 accuracy on ImageNet without using any extra data. Finetuning this model on the 384×384 input resolution can further increase the accuracy to 85.2%. Moreover, when scaling up the model size to 296M parameters, it can reach a top-1 accuracy of 87.1% on ImageNet, 90.6% on ImageNet-ReaL, and 78.0% on ImageNet-V2, setting new SOTA performance for all the three classification benchmarks.

As depicted in Figure 1, compared to the previous state-of-the-art CNN-based model (NFNet-F6 [2] with SAM [15]), and the transformer-based model (CaiT-M48 [52] with KD), our best model VOLO-D5 leverages the least amount of learnable parameters but achieves the best accuracy. Moreover, as shown in Table 1, even compared with previous state-of-the-art models using stronger data augmentation and optimization methods (such as SAM [15] and augmult [16]), our Outlooker still performs the best.

[2]https://paperswithcode.com/sota/image-classification-on-imagenet

Our VOLO also achieves strong performance on the semantic segmentation task. We run experiments on two widely-used segmentation benchmarks: Cityscapes [10] and ADE20K [77]. Experiments show that our VOLO attains 84.3% mIoU score on the Cityscapes validation set, 0.3% better than the previous state-of-the-art result (by SegFormer-B5 [64]). On the ADE20K validation set, we achieve 54.3% mIoU score, largely improving the state-of-the-art result (53.5%) by Swin Transformer [37], which is pretrained on ImageNet-22k.

2. Method

Our model can be regarded as an architecture with two separate stages. The first stage consists of a stack of Outlookers that generates fine-level token representations. The second stage deploys a sequence of transformer blocks to aggregate global information. At the beginning of each stage, a patch embedding module is used to map the input to token representations with designed shapes.

2.1. Outlooker

Outlooker consists of an outlook attention layer for spatial information encoding and a multi-layer perceptron (MLP) for inter-channel information interaction. Given a sequence of input C-dim token representations $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, Outlooker can be written as follows:

$$\tilde{\mathbf{X}} = \text{OutlookAtt}(\text{LN}(\mathbf{X})) + \mathbf{X}, \quad (1)$$

$$\mathbf{Z} = \text{MLP}(\text{LN}(\tilde{\mathbf{X}})) + \tilde{\mathbf{X}}. \quad (2)$$

Here, LN refers to LayerNorm [35].

2.1.1 Outlook Attention

Outlook attention is simple, efficient, and easy to implement. The main insights behind it are: 1) the feature at each spatial location is representative enough to generate attention weights for locally aggregating its neighboring features; 2) the dense and local spatial aggregation can encode fine-level information efficiently.

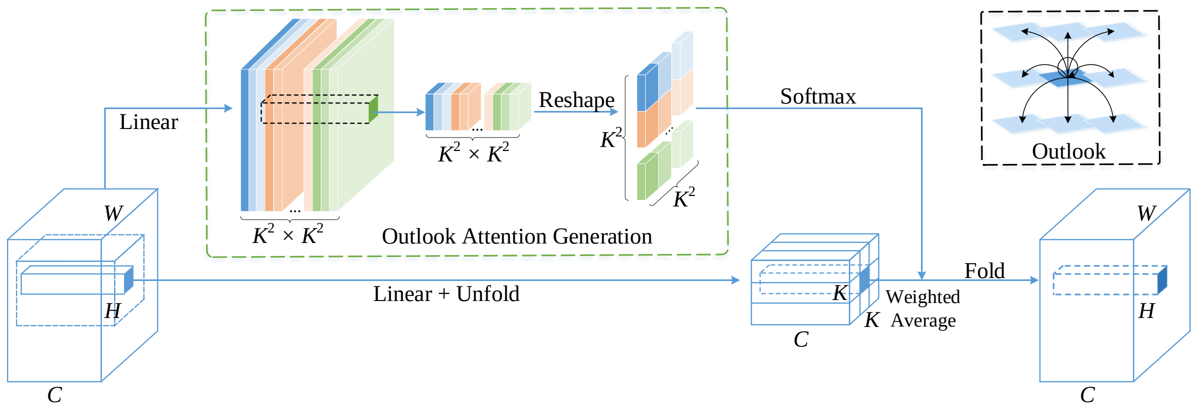

Figure 2. Illustration of outlook attention. The outlook attention matrix for a local window of size $K \times K$ can be simply generated from the center token with a linear layer followed by a reshape operation (highlighted by the green dash box). As the attention weights are generated from the center token within the window and act on the neighbor tokens and itself (as demonstrated in the black dash block), we name these operations as outlook attention.

Algorithm 1 Outlook attention code (PyTorch-like)

```
# H: height, W: width, K: kernel size
# x: input tensor (H, W, C)

def init()
   v_pj = nn.Linear(C, C)
   attn = nn.Linear(C, k ** 4)
   unfold = nn.Unfold(K, padding)
   fold = nn.Fold(output_size=(H, W), K, padding)

def outlook_attention(x): # code in forward
   v = v_pj(x).permute(2, 1, 0)

   # Eqn. (3), embedding set of neighbors
   v = unfold(v).reshape(C, K*K, H*W).permute(2, 1, 0)
   a = attn(x).reshape(H*W, K*K, K*K)

   # Eqn. (4), weighted average
   a = a.softmax(dim=-1)
   x = mul(a, v).permute(2, 1, 0).reshape(C*K*K, H*W)

   # Eqn. (5)
   x = fold(x).permute(2, 1, 0)

   return x
```

For each spatial location (i, j), outlook attention computes its similarity to all the neighbors within a local window of size $K \times K$ centered at (i, j). Unlike self-attention that requires a Query-Key matrix multiplication for the computation of the attention (i.e., $\mathrm{Softmax}(\mathbf{Q}^\top \mathbf{K}/\sqrt{d})$), outlook attention simplifies this process via just a reshaping operation.

Formally, given the input $\mathbf{X}$, each C-dim token is first projected, using two linear layers of weights $\mathbf{W}_A \in \mathbb{R}^{C \times K^4}$ and $\mathbf{W}_V \in \mathbb{R}^{C \times C}$, into outlook weights $\mathbf{A} \in \mathbb{R}^{H \times W \times K^4}$ and value representation $\mathbf{V} \in \mathbb{R}^{H \times W \times C}$, respectively. Let $\mathbf{V}_{\Delta_{i,j}} \in \mathbb{R}^{C \times K^2}$ denote all the values within the local window centered at (i, j), *i.e.*,

$$\mathbf{V}_{\Delta_{i,j}} = \{\mathbf{V}_{i+p-\lfloor \frac{K}{2} \rfloor, j+q-\lfloor \frac{K}{2} \rfloor}\}, \quad 0 \le p, q < K. \tag{3}$$

Outlook attention The outlook weight at location (i, j) is directly used as the attention weight for value aggregation, by reshaping it to $\hat{\mathbf{A}}_{i,j} \in \mathbb{R}^{K^2 \times K^2}$, followed by a Softmax function. Thus, the value projection procedure can be written as

$$\mathbf{Y}_{\Delta_{i,j}} = \mathrm{MatMul}(\mathrm{Softmax}(\hat{\mathbf{A}}_{i,j}), \mathbf{V}_{\Delta_{i,j}}). \tag{4}$$

Dense aggregation Outlook attention aggregates the projected value representations densely. Summing up the different weighted values at the same location from different local windows yields the output

$$\tilde{\mathbf{Y}}_{i,j} = \sum_{0 \le m,n < K} \mathbf{Y}^{i,j}_{\Delta_{i+m-\lfloor \frac{K}{2} \rfloor, j+n-\lfloor \frac{K}{2} \rfloor}}. \tag{5}$$

PyTorch-like outlook attention codes are summarized in Algorithm 1. Eqn. (3) and Eqn. (5) correspond to the `Unfold` and `Fold` operations, respectively. After outlook attention, a linear layer is often adopted as in self-attention.

2.1.2 Multi-Head Outlook Attention

The implementation of multi-head outlook attention is simple. Suppose the head number is set to N. We just need to adjust the weight shape of $\mathbf{W}_A$ such that $\mathbf{W}_A \in \mathbb{R}^{C \times N \cdot K^4}$. Then, the outlook weight and value embeddings are uniformly split into N segments, yielding $\mathbf{A}_n \in \mathbb{R}^{H \times W \times K^4}$ and $\mathbf{V}_n \in \mathbb{R}^{H \times W \times C_N}, \{n = 1, 2, ..., N\}$, where C_N is the dimension of each head which satisfies $C_N \times N = C$. For each $(\mathbf{A}_n, \mathbf{V}_n)$ pair, the outlook attention is separately computed, which are then concatenated as the output of the multi-head outlook attention. In our experiment section, we will ablate the impact of the head number on model performance.

Table 2. Architecture information of different variants of VOLO. The resolution information is based on an input image of size 224×224. The number of parameters includes that of weights for both the network backbone and the classifier head. 'Layer' refers to either a Outlooker block or a Transformer block.

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

2.1.3 Discussion

Our outlook attention inherits the merits of both convolutions and self-attention. It offers the following advantages. First of all, outlook attention encodes spatial information by measuring the similarity between pairs of token representations, which is more parameter-efficient for feature learning than convolutions, as studied in previous work [37, 45]. Second, outlook attention adopts a sliding window mechanism to locally encode token representations at fine level, and to some extent preserves the crucial positional information for vision tasks [25, 56]. Third, the way of generating attention weights is simple and efficient. Unlike self-attention that relies on a query-key matrix multiplication, our outlook weight can be directly produced by a simple reshaping operation, saving computation. To see this, we compare the computation for a self-attention (SA) and that for a local version of self-attention (LSA) when operating on $H \times W$ tokens with a sliding window size $K \times K$:

$$\text{M-Adds}(\textbf{SA}) \approx 4HWC^2 + 2(HW)^2C \tag{6}$$

$$\text{M-Adds}(\textbf{LSA}) \approx 4HWC^2 + 2HWK^2C \tag{7}$$

$$\text{M-Adds}(\textbf{OA}) \approx HWC(2C + NK^4) + HWK^2C. \tag{8}$$

Considering a normal case in which $C = 384$, $K = 3$, and $N = 6$, our outlook attention is more computationally efficient as $NK^4 < 2C$.

2.2. Network Architecture Variants

We build the proposed VOLO based on the LV-ViT model [32] which we find is a surprisingly strong baseline that achieves 86.2% ImageNet top-1 accuracy with 150M learnable parameters. The original LV-ViT model consists of a patch embedding module that maps an input image of size 224×224 to 14×14 tokens and a sequence of transformers that operate on the 14×14 tokens. To leverage the fine-level token representations, in the first stage, we adjust the patch embedding module to make the image tokenize on small image patches of size 8×8 instead of 16×16. A stack of Outlookers is used to generate more expressive token representations at the fine level. In the second stage, another patch embedding module is utilized to downsample the tokens. A sequence of transformers is then adopted to encode global information.

Based on the above network structure, we introduce five versions of the proposed VOLO: VOLO-D1, VOLO-D2, VOLO-D3, VOLO-D4, and VOLO-D5. Detailed hyper-parameter settings of all the five versions can be found in Table 2. In all versions, we keep the ratio of Outlooker and Transformer to around 1:3, which we have empirically found works the best in our experiments. We also add two class attention layers [52] in the final stage to update the class embedding. The hidden dimension in Outlookers is set to half of that in Transformers.

3. Experiments

We evaluate our proposed VOLO on the ImageNet [12] dataset. During training, we do not use any extra training data. Our code is based on `PyTorch` [39], the `Token Labeling toolbox` [32], and `timm` [59]. We use the LV-ViT-S [32] model with Token Labeling as our baseline.

Setup: We use the AdamW optimizer [38] with a linear learning rate scaling strategy $lr = \text{LR}_{\text{base}} \times \frac{\text{batch\_size}}{1024}$ and 5×10^{-2} weight decay rate as suggested by previous work [51, 32], and LR_{base} are given in Table 3 for all VOLO models. Stochastic Depth [29] is used. We train our models on the ImageNet dataset for 300 epochs. For data augmentation methods, we use CutOut [76], RandAug [11], and the Token Labeling objective with MixToken [32]. We do not use MixUp [72] or CutMix [70] as they conflict with MixToken. We train all VOLO models on a machine node

Table 3. Model settings. We use a linear learning rate scaling strategy $lr = \mathrm{LR}_{\mathrm{base}} \cdot \frac{\mathrm{batch\_size}}{1024}$. For all models, we set the batch size to 1024.

Specification	D1	D2	D3	D4	D5
MLP Ratio	3	3	3	3	4
Parameters	27M	59M	86M	193M	296M
Stoch. Dep. Rate	0.1	0.2	0.5	0.5	0.75
Crop Ratio	0.96	0.96	0.96	1.15	1.15
$\mathrm{LR}_{\mathrm{base}}$	1.6e-3	1e-3	1e-3	1e-3	8e-4
weight decay	5e-2	5e-2	5e-2	5e-2	5e-2

with 8 NVIDIA V100 or A100 GPUs except for VOLO-D5 which needs two nodes. For VOLO-D1 and VOLO-D2, 4 GPUs also suffice with batch size 512 (16G) or 1024 (32G). For finetuning on larger image resolutions, we set the batch size to 512, learning rate to 5e-6, weight decay to 1e-8 and run the models for 30 epochs. Other hyper-parameters are set the same as default. Finetuning requires 2-8 nodes depending on the model size.

Model Settings: The model settings for VOLO-D1 to VOLO-D5 are listed in Table 3. We find that larger models (with 100M+ parameters) suffer overfitting. To mitigate this issue, we set large stochastic depth rate for them. Moreover, the learning rate selection also has a slight impact on the performance. We find it is more beneficial to use larger initial learning rates for small-sized models. In addition, the crop ratio can also slightly influence the performance. Larger models prefer larger crop ratios.

3.1. Main Results

We compare the proposed VOLO with the state-of-the-art models from the literature in Table 4. All results listed are based on using only ImageNet-1k images for training and no extra training data are used. "Top-1," "Real Top-1," and "V2 Top-1" refer to the top-1 accuracy using the original ImageNet validation labels, cleaned-up real labels [1], and ImageNetV2 labels [43], respectively. "Train size" and "Test size" represent resolutions used in training and finetuning (test for CNNs). We separate the results into five segments according to model size (number of parameters).

As can be seen, for different model sizes, our proposed VOLO consistently performs better than previous state-of-the-art models. Specially, taking the proposed VOLO-D1 with 26.6M parameters as an example, testing on a resolution of 224 already yields 84.2% top-1 accuracy on ImageNet. Finetuning on 384 resolution further improves the performance to 85.2%, which is clearly better than all the models with a comparable amount of training parameters. When the model size is scaled up to 296M, we can achieve 87.1% top-1 accuracy on ImageNet, setting a new record in case of no extra training data. *To the best of our knowledge, our VOLO-D5 is the first reaching 87.1% top-1 accuracy on ImageNet without extra training data.*

Our models also achieve the best results on the "Real Top-1" and "V2 Top-1" benchmarks. As shown in Table 4, our VOLO-D4 with merely 193M parameters performs much better than previous state-of-the-art models, such as CaiT-M48 and NFNet. Our models perform even better on the ImageNet-V2 benchmark. As can be seen, our VOLO-D3 can improve upon the previous best result by 0.8% (76.9% *v.s.* 77.7%) using only a quarter of the parameters of CaiT-M48 (86M *v.s.* 356M). Our largest VOLO-D5 can further boost the performance to 78%.

3.2. Performance of Outlooker

In this subsection, we demonstrate the importance of the proposed Outlooker in VOLO. We take the recent state-of-the-art vision transformer model, named LV-ViT-S, as our baseline. LV-ViT-S contains 16 transformers in total and receives 83.3% top-1 accuracy on ImageNet. Each token in LV-ViT-S corresponds to an image patch of size 16×16, and hence there are totally 14×14 tokens for a 224×224 input image. The experiment path from the LV-ViT-S [32] baseline to our VOLO-D1 and the corresponding results can be found in Table 5.

As the goal of our proposed Outlooker is to encode expressive finer-level features, we first adjust the starting patch embedding module and change the patch size from 16×16 to 8×8. We replace two transformers with our Outlooker at the fine level. As can be seen from the second row of Table 5, such a slight adjustment brings us 0.4% gain based on the baseline that already reaches 83.3% top-1 accuracy. Adding another two Outlookers further increases the performance to 83.9%. Finally, changing the head number in all the transformers from 6 to 12 and finetuning the resulting model at 384×384 resolution allows us to yield a result of 85.2%, which, to the best of our knowledge, is the first time to attain 85+% accuracy within less than 30M parameters.

We also attempt to replace the proposed outlook attention with other methods for fine-level feature encoding, including local self-attention [37] and spatial convolutions. For a fair comparison, we set the window size to 3×3 for both local self-attention and convolutions. The results can be found in Table 6. As can be seen, under the same training recipe and architecture, our Outlooker performs better than both local self-attention and convolutions. In addition, we can also observe that local self-attention and convolutions can also lift the performance compared to the LV-ViT-S baseline, demonstrating that encoding fine-level token representations indeed helps.

3.3. Ablation Analysis

Model Scaling: We scale up the VOLO-D1 model to 4 different models (VOLO-D2 to VOLO-D5) in two different

Table 4. Top-1 accuracy comparison of our method with previous state-of-the-art methods on ImageNet [12], ImageNet Real [1], and ImageNet-V2 [43]. We split the results into 5 segments according the model size. All models are trained without external data. With the same computation and parameter constraint, our model consistently outperforms other MLP-like, CNN-based, and transformer-based counterparts. 'Train size' and 'Test size' refer to resolutions used in training and finetuning (test for CNNs). Our VOLO-D5 sets a new record on all three benchmarks, which is the first model attaining 87.1% top-1 accuracy on ImageNet.

Network	Architecture	Params	FLOPs	Train size	Test size	Top-1	Real Top-1	V2 Top-1
DeiT-S [51]	Transformer	22M	4.6B	224	224	79.9	85.7	68.5
T2T-ViT-14 [68]	Transformer	22M	5.2B	224	224	81.5	86.8	69.9
T2T-ViT-14↑384 [68]	Transformer	22M	17.1B	224	384	83.3	87.8	72.4
DeepViT-S [78]	Transformer	27M	6.2B	224	224	82.3	-	-
ViP-Small/7 [23]	MLP-like	25M	-	224	224	81.5	-	-
BoTNet-S1-59 [45]	Hybrid	34M	7.3B	224	224	81.7	-	-
EfficientNet-B5 [50]	CNN	30M	9.9B	456	456	83.6	88.3	73.6
LV-ViT-S↑384 [32]	Transformer	26M	22.2B	224	384	84.4	88.9	74.5
VOLO-D1	VOLO	27M	6.8B	224	224	84.2	89.0	74.0
VOLO-D1↑384	**VOLO**	**27M**	**22.8B**	**224**	**384**	**85.2**	**89.6**	**75.6**
CrossViT [4]	Transformer	45M	56.6B	224	480	84.1	-	-
TNT-B [19]	Transformer	66M	14.1B	224	224	82.8	-	-
ViP-Medium/7 [23]	MLP-like	55M	-	224	224	82.7	-	-
DeepViT-L [78]	Transformer	55M	12.5B	224	224	83.1	-	-
EfficientNet-B7 [50]	CNN	66M	37.0B	600	600	84.3	-	-
NFNet-F0 [2]	CNN	72M	12.4B	192	256	83.6	88.1	72.6
CaiT-S36↑384 [52]	Transformer	68M	48.0B	224	384	85.4	89.8	76.2
LV-ViT-M↑384 [32]	Transformer	56M	42.2B	224	384	85.4	89.5	76.0
VOLO-D2	VOLO	59M	14.1B	224	224	85.2	89.3	75.2
VOLO-D2↑384	**VOLO**	**59M**	**46.1B**	**224**	**384**	**86.0**	**89.7**	**76.4**
ViT-B/16 [14]	Transformer	86M	55.4B	224	384	77.9	83.6	-
DeiT-B [51]	Transformer	86M	17.5B	224	224	81.8	86.7	-
ViP-Large/7 [23]	MLP-like	88M	-	224	224	83.2	-	-
Swin-B [37]	Transformer	88M	47.0B	224	384	84.2	-	-
BoTNet-S1-128↑384 [45]	Hybrid	79M	45.8B	256	384	84.7	-	-
Fix-EfficientNet-B8 [50, 53]	CNN	87M	89.5B	672	800	85.7	90.0	-
Refined-ViT-L↑448 [79]	Transformer	81M	98.0B	224	448	85.9	-	-
VOLO-D3	VOLO	86M	20.6B	224	224	85.4	89.6	75.6
VOLO-D3↑448	**VOLO**	**86M**	**67.9B**	**224**	**448**	**86.3**	**90.0**	**77.7**
NFNet-F1 [2]	CNN	133M	35.5B	224	320	84.7	88.9	74.4
NFNet-F2 [2]	CNN	194M	62.6B	256	352	85.1	88.9	74.3
NFNet-F3 [2]	CNN	255M	115.0B	320	416	85.7	89.4	75.2
VOLO-D4	VOLO	193M	43.8B	224	224	85.7	89.7	75.6
VOLO-D4↑448	**VOLO**	**193M**	**197B**	**224**	**448**	**86.8**	**90.5**	**77.8**
NFNet-F4 [2]	CNN	316M	215B	384	512	85.9	89.4	75.2
NFNet-F5 [2]	CNN	377M	290B	416	544	86.0	89.2	74.6
NFNet-F6 [2]+SAM	CNN	438M	377B	448	576	86.5	89.2	75.8
ViT-L/16 [14]	Transformer	307M	191B	224	384	76.5	82.2	-
CaiT-M36↑448 [52]	Transformer	271M	248B	224	448	86.3	90.2	76.7
CaiT-M48↑448 [52]	Transformer	356M	330B	224	448	86.5	90.2	76.9
VOLO-D5	VOLO	296M	69.0B	224	224	86.1	89.9	76.3
VOLO-D5↑448	**VOLO**	**296M**	**304B**	**224**	**448**	**87.0**	**90.6**	**77.8**
VOLO-D5↑512	**VOLO**	**296M**	**412B**	**224**	**512**	**87.1**	**90.6**	**78.0**

ways: 1) increasing the model size during training, including network depth, hidden dimension, expansion ratio in MLP, and head number in both Outlookers and Transformers, and 2) increasing the image resolution during finetuning and test. The specifications for all models have been shown in Table 2 and their corresponding results can be found in Table 7. We can observe that both aforementioned ways can largely improve the model performance. From VOLO-D1 to VOLO-D2, there is 1% improvement with doubled parameters. Further increasing the model size form VOLO-

Table 5. Ablation path from the LV-ViT-S [32] baseline to our VOLO-D1. All experiments, except for larger input resolution, can be finished within 3 days using a single server node with 8 V100 GPUs or 2 days with 8 A100 GPUs. Clearly, with only 27M learnable parameters, the performance can be boosted from 83.3 to 85.2 (**+1.9**) using the proposed VOLO architecture. 'T' and 'O' refer to Transformer and Outlooker, respectively.

Training techniques	Layers	#Param.	Top-1 Acc. (%)
Baseline (LV-ViT-S [32])	16	26M	83.3
+ Replace 2 Ts with Os	16	25M	83.7 (**+0.4**)
+ Add 2 more Os	18	27M	84.0 (**+0.7**)
+ #Head in Ts (6 $\rightarrow$ 12)	18	27M	84.2 (**+0.9**)
+ Resolution (224 $\rightarrow$ 384)	18	27M	85.2 (**+1.9**)

Table 6. Performance of Outlooker against local self-attention and convolutions. For both self-attention and convolutions, we set the kernel size to 3×3.

Model	Layer type	#Params	Top-1 Acc.
VOLO-D1	Outlooker	27M	**84.2**
VOLO-D1	Local self-attention	27M	83.8
VOLO-D1	Convolution	27M	83.8

Table 7. Model performance when scaling up in two different ways: training model size and testing resolution. The computations (M-Adds) reported here are based on 224×224 image resolution.

Model	#Params	M-Adds	Top-1 Acc.	Top-1 Acc.$\uparrow$
VOLO-D1	26.6M	6.8B	84.2@224	85.4@384
VOLO-D2	58.7M	14.1B	85.2@224	86.0@384
VOLO-D3	86.3M	20.6B	85.4@224	86.3@448
VOLO-D4	193M	43.8B	85.7@224	86.7@448
VOLO-D5	296M	69.0B	**86.1@224**	**87.1@512**

D2 to VOLO-D5 yields nearly another 1% accuracy gain. In addition, for all the five models, increasing the resolution during finetuning brings around 1% performance gain.

Number of Outlookers: We observe that the number of Outlookers used in our VOLO has an impact on the classification performance. Here, we investigate the influence of using different numbers of Outlookers in our VOLO. Note that all Outlookers act on finer-level token representations (28×28). The results have been shown in the top part of Table 8. Without any Outlookers, the baseline with 16 transformers receives 83.3% accuracy. Increasing the number of Outlookers can improve the result but the performance saturates when using 4 Outlookers. Further adding Outlookers does not bring any performance gain. Thus, when scaling up the model, we approximately use a ratio of 1:3 for Outlooker and Transformers.

Head Number in Outlookers: In Transformers, the channel dimension in each head is inversely proportional with the head number given a fixed hidden dimension. Differently, in Outlookers, the channel dimension in each head is fixed when the kernel size is fixed (*i.e.*, $C = K^4$). So, will Outlookers perform better if more heads are used? In the bottom part of Table 8, we show the results with different head numbers Outlookers. Experiments show that using more heads in Outlookers can slightly improve the performance with nearly no extra parameter increase but such increase stops when the head number is more than 6. Therefore, by default, we set the head number in Outlookers to 6 for 384 hidden dimension. When the hidden dimension is set to 768, we use 12 heads in Outlookers.

Table 8. More ablation experiments on Outlooker. 'O' and 'T' refer to Outlooker and Transformer, respectively. All results are based on VOLO-D1 with test resolution 224×224.

(#O, #T)	#Heads in (O, T)	Kernel Size	#Params	Top-1 Acc.
(0, 16)	(-, 6)	3×3	29.1M	83.3
(2, 14)	(6, 6)	3×3	25.9M	83.7
(4, 14)	(6, 6)	3×3	26.6M	**84.0**
(6, 12)	(6, 6)	3×3	24.5M	83.9
(4, 14)	(2, 6)	3×3	26.4M	83.9
(4, 14)	(4, 6)	3×3	26.5M	83.9
(4, 14)	(6, 6)	3×3	26.6M	84.0
(4, 14)	(8, 6)	3×3	26.8M	84.0
(4, 14)	(6, 12)	3×3	26.6M	**84.2**

3.4. Semantic Segmentation

In this subsection, we use our VOLO as pretrained models to evaluate the performance in semantic segmentation. Our code is based on `mmsegmentation` [9]. We report results on two widely-used segmentation benchmarks: Cityscapes [10] and ADE20K [77]. The UperNet [62] segmentation framework is adopted. In training, we utilize the AdamW optimizer with an initial learning rate of 6e-5 and a weight decay of 0.01. We also use a linear learning schedule with a minimum learning rate of 5e-6. All models can be trained on a machine node with 8 A100 GPUs. For cityscapes, we set the batch size to 8 and the input resolution to 1024×1024. For ADE20K, the batch size is set to 16 and input resolution 512×512 is used. As suggested by [77], we report results in terms of mean intersection-over-union (mIoU) for both datasets and mean pixel accuracy for ADE20K. In inference, we perform multi-scale test with interpolation rates of [0.75, 1.0, 1.25, 1.5, 1.75].

3.4.1 Cityscapes

Cityscapes [10] is one of the most popular datasets for semantic segmentation, which targets at street scene segmentation. It has 5K high-quality pixel-annotated images with resolution 1024×2048 and contains 19 classes in total. As in most previous work, we split the whole dataset into three splits for training, validation and test, which contain

Table 9. Comparisons with the state-of-the-arts on the Cityscapes validation set [10]. 'Pretrained' refers to the dataset each backbone network is pretrained on. All models are trained on the training set and multi-scale test results are reported in the 'mIoU' column.

Method	Backbone	Pretrained	mIoU
DenseASPP [66]	DenseNet [28]	ImgNet-1k	80.6
DeepLabv3+ [6]	Xception-65 [8]	ImgNet-1k	79.1
DPC [5]	Xception-71 [8]	ImgNet-1k	80.8
DANet [17]	ResNet-101	ImgNet-1k	81.5
CCNet [31]	ResNet-101	ImgNet-1k	81.3
Strip Pooling [24]	ResNet-101	ImgNet-1k	81.9
SETR [75]	ViT-L [14]	ImgNet-22k	82.1
PatchDiverse [18]	Swin-L [37]	ImgNet-22k	83.6
SpineNet-S143+ [42]	SpineNet	ImgNet-1k	83.0
SegFormer-B5 [64]	SegFormer	ImgNet-1k	84.0
VOLO-D1	VOLO	ImgNet-1k	83.1
VOLO-D4	VOLO	ImgNet-1k	**84.3**

Table 10. Comparison with previous state-of-the-art methods on the ADE20K validation set. Our VOLO-D5 achieves the best result on ADE20K with only ImageNet-1K as training data in pretraining. 'Pixel' refers to mean pixel accuracy.

Method	Backbone	Pretrained	mIoU	Pixel
PSPNet [74]	ResNet-269	ImgNet-1k	44.9	81.7
UperNet [62]	ResNet-101	ImgNet-1k	44.9	-
Strip Pooling [24]	ResNet-101	ImgNet-1k	45.6	82.1
DeepLabV3+ [6]	ResNeSt200	ImgNet-1k	48.4	-
SETR [75]	ViT-Large	ImgNet-22k	50.3	83.5
SegFormer-B5 [64]	SegFormer	ImgNet-1k	51.8	-
Swin-B [37]	Swin-B	ImgNet-22k	51.6	-
Seg-L-Mask/16 [46]	ViT-Large	ImgNet-22k	53.2	-
Swin-L [37]	Swin-L	ImgNet-22k	53.5	-
VOLO-D1	VOLO	ImgNet-1k	50.5	83.3
VOLO-D3	VOLO	ImgNet-1k	52.9	84.6
VOLO-D5	VOLO	ImgNet-1k	**54.3**	**85.0**

2,975, 500, and 1,525 images, respectively. We report results on the validation set. The comparison results can be found in Table 9. It is obvious that the proposed approach outperforms all other methods, including the recent state-of-the-art SegFormer-B5 model. Our VOLO-D4 with UperNet decoder head achieves the best result 84.3%, 0.3% better than the previous state-of-the-art result 84.0% made by SegFormer-B5. According to PaperWithCode[3], this is a new state-of-the-art result on Cityscapes validation set.

3.4.2 ADE20K

We also run experiments on the widely-used ADE20K [77] dataset. ADE20K contains 25K images in total, including 20K images for training, 2K images for validation, and 3K images for test. It covers 150 different common foreground categories. We compare our segmentation results with previous state-of-the-art segmentation methods in Table 10. Without pretraining on large-scale datasets, such as ImageNet-22K, our VOLO-D1 with UperNet achieves an mIoU score of 50.5. When the VOLO-D5 is used as backbone, the mIoU score can be further improved to 54.3, a new state-of-the-art result on ADE20K with no extra pretraining data except for ImageNet-1k.

4. Related Work

As one of the most fundamental problems in computer vision, image classification has experienced remarkable progress since the introduction of deep neural network models. In what follows, we briefly review those successful models that are closely related to this work.

Earlier models attaining state-of-the-art performance for image classification are mostly CNN-based ones that simply stack a sequence of spatial convolutions and poolings, represented by AlexNet [33] and VGGNet [44]. ResNets [20] advances the design of CNN architectures by introducing skip connections to enable training of very deep models. Inceptions [48, 49, 47] and ResNeXt [65] examine the design principles of the model building blocks and introduce multiple parallel paths of sets of specialized filters. SENet [27] presents a squeeze-and-excitation module to explicitly model the inter-dependencies among channels. DPNs [7] leverage both residual and dense connections for designing stronger building blocks. EfficientNet [50] and NasNet [80] take advantage of neural architecture search to search for powerful network architectures. Later state-of-the-art models [30, 53, 63] mostly utilize different training or optimization methods or finetuning techniques to improve EfficientNet. Very recently, NFNet [2] breaks the dominance of EfficientNet by designing a normalization-free architecture, making the first work attaining 86.5% top-1 accuracy on ImageNet using no extra data. CNNs, as the de-facto networks in visual recognition for years, have indeed been very successful but their focus is on how to learn more discriminative local features by designing better architectures. Essentially, they are short of the capability of explicitly building global relationships among representations that have been proven crucial [58].

Recent progress on image classification is mostly driven by attention-based models [73, 58, 26] or specifically transformer-based models. Transformers make use of the self-attention mechanism, making modeling long-range dependencies possible. Transformers [55] are originally designed for natural language tasks [13, 41, 3, 67, 40, 36] and have recently been demonstrated effective in image classi-

[3]https://paperswithcode.com/sota/semantic-segmentation-on-cityscapes-val

fication. Dosovitskiy *et al.* [14] are among the first to show that purely transformer-based architectures (*i.e.*, ViT) can also get state-of-the-art performance in image classification but require large-scale datasets, such as ImageNet-22k and JFT-300M (which is not publicly available) for pretraining. DeiT [51] and T2T-ViT [68] mitigate the problem of ViTs requiring large-scale datasets and propose data efficient ViTs. Since then, a surge of works on ViT continuously come into being with further improvements. Some of them [4, 19, 61, 54, 79] introduce local dependency into vision transformers by modifying the patch embedding block or the transformer block or both, while others [21, 37, 57] adopt a pyramid structure to reduce the overall computation while maintaining the models' ability to capture low-level features. There are also some works [78, 71, 52, 18] aiming at solving the optimization and scaling problems of ViTs.

Our VOLO not only models long-range dependencies but also encodes fine-level features into token representations by the proposed Outlooker. Unlike the recent hybrid architectures (*e.g.*, Hybrid-ViT [14] and BoTNet [45]) that rely on convolutions for feature encoding, Outlooker proposes to use local pair-wise token similarities to encode fine-level features and spatial context into tokens features and hence is more effective and parameter-efficient. This also makes our model different from the Dynamic Convolution [60] and Involution [34] that generate input-dependent convolution kernels to encode the features.

5. Conclusions

We presented a new model, Vision Outlooker (VOLO). Extensive experiments for image classification and segmentation demonstrate VOLO outperforms CNN- and Transformer-based models, and establishes new SOTA results. We hope that the strong performance of VOLO on several computer vision tasks will encourage follow-up research on better fine-level feature learning. The performance superiority of VOLO comes from the new outlook attention mechanism that dynamically aggregates fine-level features in a dense manner, and we will continue our investigation in other applications, like natural language processing.

Acknowledgement

We gratefully acknowledge the support of NVIDIA AI Tech Center (NVAITC) to this research project, especially the great helps in GPU technology supports from Terry Jianxiong Yin (NVAITC) and Qingyi Tao (NVAITC).

References

[1] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

[2] Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[4] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *arXiv preprint arXiv:2103.14899*, 2021.

[5] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NeurIPS*, pages 8699–8710, 2018.

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[7] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *Advances in Neural Information Processing Systems*, pages 4467–4475, 2017.

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[9] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[11] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[12] 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.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[15] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[16] Stanislav Fort, Andrew Brock, Razvan Pascanu, Soham De, and Samuel L Smith. Drawing multiple augmentation samples per image during training efficiently decreases test error. *arXiv preprint arXiv:2105.13343*, 2021.

[17] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[18] Chengyue Gong, Dilin Wang, Meng Li, Vikas Chandra, and Qiang Liu. Improve vision transformers training by suppressing over-smoothing. *arXiv preprint arXiv:2104.12753*, 2021.

[19] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv preprint arXiv:2103.16302*, 2021.

[22] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[23] Qibin Hou, Zihang Jiang, Li Yuan, Ming-Ming Cheng, Shuicheng Yan, and Jiashi Feng. Vision permutator: A permutable mlp-like architecture for visual recognition, 2021.

[24] Qibin Hou, Li Zhang, Ming-Ming Cheng, and Jiashi Feng. Strip pooling: Rethinking spatial pooling for scene parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4003–4012, 2020.

[25] Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2021.

[26] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3464–3473, 2019.

[27] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[28] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[29] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[30] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32:103–112, 2019.

[31] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. *arXiv preprint arXiv:1811.11721*, 2018.

[32] Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. *arXiv preprint arXiv:2104.10858*, 2021.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[34] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12321–12330, 2021.

[35] Fenglin Liu, Xuancheng Ren, Zhiyuan Zhang, Xu Sun, and Yuexian Zou. Rethinking skip connection with layer normalization. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3586–3598, 2020.

[36] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[37] 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.

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

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[40] Matthew E Peters, Mark Neumann, Robert L Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith.

Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164*, 2019.

[41] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[42] Abdullah Rashwan, Xianzhi Du, Xiaoqi Yin, and Jing Li. Dilated spinenet for semantic segmentation. *arXiv preprint arXiv:2103.12270*, 2021.

[43] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[45] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021.

[46] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021.

[47] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[49] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[50] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[51] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[52] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021.

[53] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. *arXiv preprint arXiv:1906.06423*, 2019.

[54] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. *arXiv preprint arXiv:2103.12731*, 2021.

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[56] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Standalone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020.

[57] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021.

[58] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[59] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[60] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

[61] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

[62] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018.

[63] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020.

[64] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *arXiv preprint arXiv:2105.15203*, 2021.

[65] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[66] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[67] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[68] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[69] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911, 2020.

[70] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[71] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *arXiv preprint arXiv:2106.04560*, 2021.

[72] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[73] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020.

[74] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[75] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *arXiv preprint arXiv:2012.15840*, 2020.

[76] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.

[77] 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*, 127(3):302–321, 2019.

[78] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

[79] Daquan Zhou, Yujun Shi, Bingyi Kang, Weihao Yu, Zihang Jiang, Yuan Li, Xiaojie Jin, Qibin Hou, and Jiashi Feng. Refiner: Refining self-attention for vision transformers. *arXiv preprint arXiv:2106.03714*, 2021.

[80] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.