All Tokens Matter: Token Labeling for Training Better Vision Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we present token labeling—a new training objective for training high-performance vision transformers (ViTs). Different from the standard training objective of ViTs that computes the classification loss on an additional trainable class token, our proposed one takes advantage of all the image patch tokens to compute the training loss in a dense manner. Specifically, token labeling reformulates the image classification problem into multiple token-level recognition problems and assigns each patch token with an individual location-specific supervision generated by a machine annotator. Experiments show that token labeling can clearly and consistently improve the performance of various ViT models across a wide spectrum. For a vision transformer with 26M learnable parameters serving as an example, with token labeling, the model can achieve 84.4% Top-1 accuracy on ImageNet. The result can be further increased to 86.2% by slightly scaling the model size up to 150M, delivering the minimal-sized model among previous models (250M+) reaching 86%. We also show that token labeling can clearly improve the generalization capability of the pretrained models on downstream tasks with dense prediction, such as semantic segmentation. Code will be made publicly available.

1 Introduction

Transformers [37] have achieved great performance for almost all the natural language processing (NLP) tasks over the past years [4, 14, 24]. Motivated by such success, recently, many researchers attempt to build transformer models for vision tasks, and their encouraging results have shown the great potential of transformer based models for image classification [6, 15, 25, 34, 38, 44], especially the strong benefits of the self-attention mechanism in building long-range dependencies between pairs of input tokens.

Despite the importance of gathering long-range dependencies, recent work on local data augmentation [55] has demonstrated that well modeling and leveraging local information for image classification would avoid biasing the model towards skewed and non-generalizable patterns and substantially improve the model performance. However, recent vision transformers normally utilize class tokens that aggregate global information to predict the output class while neglecting the role of other patch tokens that encode rich information on their respective local image patches.

In this paper, we present a new training objective for vision transformers, termed *token labeling*, that takes advantage of both the patch tokens and the class tokens. Our method takes a K-dimensional score map generated by a machine annotator as supervision to supervise all the tokens in a dense manner, where K is the number of categories for the target dataset. In this way, each patch token is explicitly associated with an individual location-specific supervision indicating the existence of the target objects inside the corresponding image patch, so as to improve the object grounding and

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

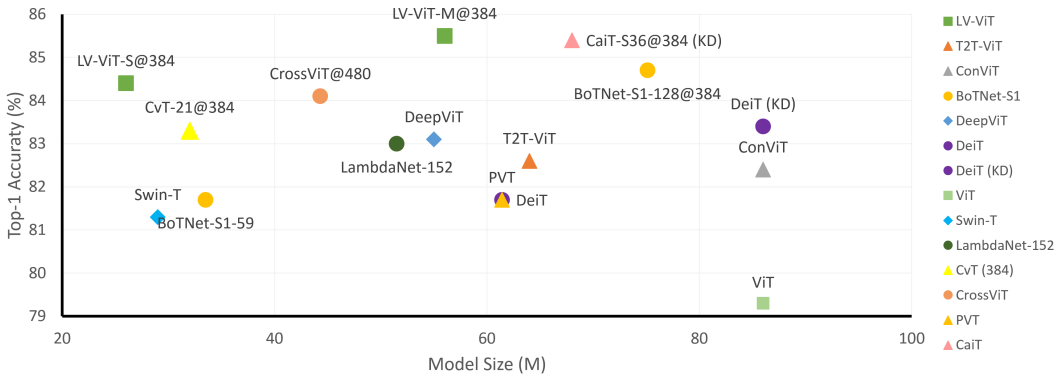

Figure 1: Comparison between the proposed LV-ViT and other recent works based on vision transformers, including T2T-ViT [44], ConViT [12], BoTNet [30], DeepViT [57], DeiT [34], ViT [15], Swin Transformer [25], LambdaNet [1], CvT [41], CrossViT [6], PVT [38], CaiT [35]. Note that we only show models whose model sizes are under 100M. As can be seen, our LV-ViT achieves the best results using the least amount of learnable parameters. The default test resolution is 224×224 unless specified after @.

recognition capabilities of vision transformers with negligible computation overhead. To the best of our knowledge, this is the first work demonstrating that dense supervision is beneficial to vision transformers in image classification.

According to our experiments, utilizing the proposed token labeling objective can clearly boost the performance of vision transformers. As shown in Figure 1, our model, named LV-ViT, with 56M parameters, yields 85.4% top-1 accuracy on ImageNet [13], behaving better than all the other transformer-based models having no more than 100M parameters. When the model size is scaled up to 150M, the result can be further improved to 86.2%. In addition, we have empirically found that the pretrained models with token labeling are also beneficial to downstream tasks with dense prediction, such as semantic segmentation.

2 Related Work

Transformers [37] refer to the models that entirely rely on the self-attention mechanism to build global dependencies, which are originally designed for natural language processing tasks. Due to their strong capability of capturing spatial information, transformers have also been successfully applied to a variety of vision problems, including low-level vision tasks like image enhancement [7, 43], as well as more challenging tasks such as image classification [9, 15], object detection [5, 11, 53, 59], segmentation [7, 32, 39] and image generation [28]. Some works also extend transformers for video and 3D point cloud processing [48, 51, 58].

Vision Transformer (ViT) is one of the earlier attempts that achieved state-of-the-art performance on ImageNet classification, using pure transformers as basic building blocks. However, ViTs need pretraining on very large datasets, such as ImageNet-22k and JFT-300M, and huge computation resources to achieve comparable performance to ResNet [18] with a similar model size trained on ImageNet. Later, DeiT [34] manages to tackle the data-inefficiency problem by simply adjusting the network architecture and adding an additional token along with the class token for Knowledge Distillation [21, 45] to improve model performance.

Some recent works [6, 16, 41, 44] also attempt to introduce the local dependency into vision transformers by modifying the patch embedding block or the transformer block or both, leading to significant performance gains. Moreover, there are also some works [20, 25, 38] adopting a pyramid structure to reduce the overall computation while maintaining the model's ability to capture low-level features.

Unlike most aforementioned works that design new transformer blocks or transformer architectures, we attempt to improve vision transformers by studying the role of patch tokens that embed rich local information inside image patches. We show that by slightly tuning the structure of vision transformers

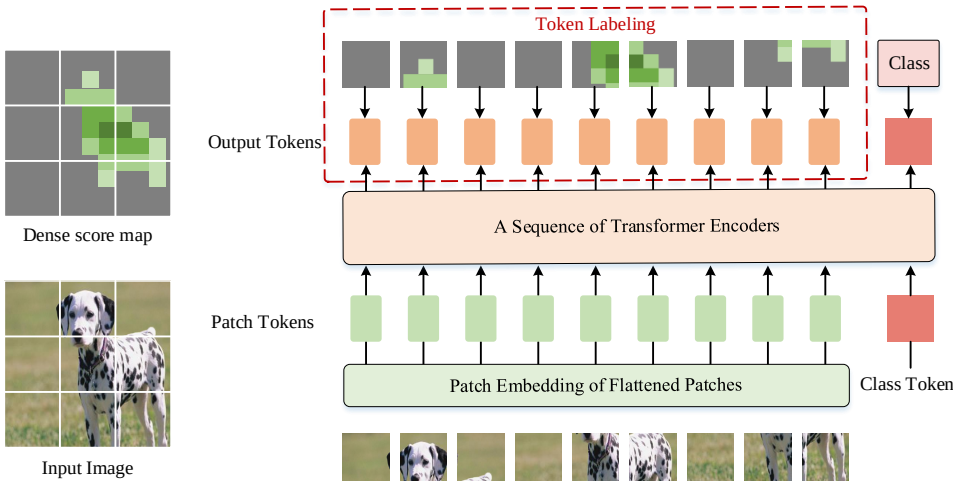

Figure 2: Pipeline of training vision transformers with token labeling. Other than utilizing the class token (pink rectangle), we also take advantage of all the output patch tokens (orange rounded rectangle) by assigning each patch token an individual location-specific prediction generated by a machine annotator [3] as supervision (see the part in the red dash rectangle). Our proposed token labeling method can be treated as an auxiliary objective to provide each patch token the local details that aid vision transformers to more accurately locate and recognize the target objects. Note that the traditional vision transformer training does not include the red dash rectangle part.

and employing the proposed token labeling objective, we can achieve strong baselines for transformer models at different model size levels.

3 Token Labeling Method

In this section, we first briefly review the structure of the vision transformer [15] and then describe the proposed training objective—*token labeling*.

3.1 Revisiting Vision Transformer

A typical vision transformer [15] first decomposes a fixed-size input image into a sequence of small patches. Each small patch is mapped to a feature vector, or called a token, by projection with a linear layer. Then, all the tokens combined with an additional learnable class token for classification score prediction are sent into a stack of transformer blocks for feature encoding.

In loss computing, the class token from the output tokens of the last transformer block is usually selected and sent into a linear layer for the classification score prediction. Mathematically, given an image I, denote the output of the last transformer block as $[X^{cls}, X^1, ..., X^N]$, where N is the total number of patch tokens, and X^{cls} and $X^1, ..., X^N$ correspond to the class token and the patch tokens, respectively. The classification loss for image I can be written as

$$L_{cls} = H(X^{cls}, y^{cls}), \tag{1}$$

where $H(\cdot, \cdot)$ is the softmax cross-entropy loss and y^{cls} is the class label.

3.2 Token Labeling

The above classification problem only adopts an image-level label as supervision whereas it neglects the rich information embedded in each image patch. In this subsection, we present a new training objective—*token labeling*—that takes advantage of the complementary information between the patch tokens and the class tokens.

Token Labeling: Different from the classification loss as formulated in Eqn. (1) that measures the distance between the single class token (representing the whole input image) and the corresponding

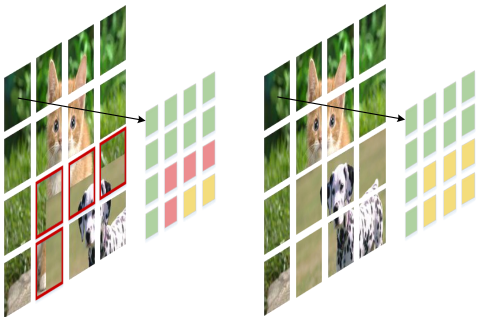

Figure 3: Comparison between CutMix [46] (**Left**) and our proposed MixToken (**Right**). CutMix is operated on the input images. This results in patches containing mixed regions from the two images (see the patches enclosed by red bounding boxes). Differently, MixToken targets at mixing tokens after patch embedding. This enables each token after patch embedding to have clean content as shown in the right part of this figure. The detailed advantage of MixToken can be found in Sec. 4.2.

image-level label, token labeling emphasizes the importance of all output tokens and advocates that each output token should be associated with an individual location-specific label. Therefore, in our method, the ground truth for an input image involves not only a single K-dimensional vector y^{cls} but also a $K \times N$ matrix or called a K-dimensional score map as represented by $[y^1, ..., y^N]$, where N is the number of the output patch tokens.

Specifically, we leverage a dense score map for each training image and use the cross-entropy loss between each output patch token and the corresponding aligned label in the dense score map as an auxiliary loss at the training phase. Figure 2 provides an intuitive interpretation. Given the output patch tokens $X^1, ..., X^N$ and the corresponding labels $[y^1, ..., y^N]$, the token labeling objective can be defined as

$$L_{tl} = \frac{1}{N} \sum_{i=1}^{N} H(X^i, y^i). \tag{2}$$

Recall that H is the cross-entropy loss. Therefore, the total loss function can be written as

$$L_{total} = H(X^{cls}, y^{cls}) + \beta \cdot L_{tl}, \tag{3}$$

$$= H(X^{cls}, y^{cls}) + \beta \cdot \frac{1}{N} \sum_{i=1}^{N} H(X^i, y^i), \tag{4}$$

where β is a hyper-parameter to balance the two terms. In our experiment, we empirically set it to 0.5.

Advantages: Our token labeling offers the following advantages. *First of all*, unlike knowledge distillation methods that require a teacher model to generate supervision labels online, token labeling is a cheap operation. The dense score map can be generated by a pretrained model in advance (e.g., EfficientNet [33] or NFNet [3]). During training, we only need to crop the score map and perform interpolation to make it aligned with the cropped image in the spatial coordinate. Thus, the additional computations are negligible. *Second*, rather than utilizing a single label vector as supervision as done in most classification models and the ReLabel strategy [47], we also harness score maps to supervise the models in a dense manner and thereby the label for each patch token provides location-specific information, which can aid the training models to easily discover the target objects and improve the recognition accuracy. *Last but not the least*, as dense supervision is adopted in training, we found that the pretrained models with token labeling benefit downstream tasks with dense prediction, like semantic segmentation.

3.3 Token Labeling with MixToken

While training vision transformer, previous studies [34, 44] have shown that augmentation methods, like MixUp [50] and CutMix [46], can effectively boost the performance and robustness of the models. However, vision transformers rely on patch-based tokenization to map each input image to a sequence of tokens and our token labeling strategy also operates on patch-based token labels. If we apply CutMix directly on the raw image, some of the resulting patches may contain content from two images, leading to mixed regions within a small patch as shown in Figure 3. When performing token labeling, it is difficult to assign each output token a clean and correct label. Taking this situation into account, we rethink the CutMix augmentation method and present MixToken, which can be viewed

as a modified version of CutMix operating on the tokens after patch embedding as illustrated in the right part of Figure 3.

To be specific, for two images denoted as I_1, I_2 and their corresponding token labels $Y_1 = [y_1^1, ..., y_1^N]$ as well as $Y_2 = [y_2^1, ..., y_2^N]$, we first feed the two images into the patch embedding module to tokenize each as a sequence of tokens, resulting in $T_1 = [t_1^1, ..., t_1^N]$ and $T_2 = [t_2^1, ..., t_2^N]$. Then, we produce a new sequence of tokens by applying MixToken using a binary mask M as follows:

$$\hat{T} = T_1 \odot M + T_2 \odot (1 - M), \tag{5}$$

where $\odot$ is element-wise multiplication. We use the same way to generate the mask M as in [46]. For the corresponding token labels, we also mix them using the same mask M:

$$\hat{Y} = Y_1 \odot M + Y_2 \odot (1 - M). \tag{6}$$

The label for the class token can be written as

$$\hat{y^{cls}} = \bar{M} y_1^{cls} + (1 - \bar{M}) y_2^{cls}, \tag{7}$$

where $\bar{M}$ is the average of all element values of M.

4 Experiments

4.1 Experiment Setup

We evaluate our method on the ImageNet [13] dataset. All experiments are built and conducted upon `PyTorch` [29] and the `timm` [40] library. We follow the standard training schedule and train our models on the ImageNet dataset for 300 epochs. Besides normal augmentations like CutOut [55] and RandAug [10], we also explore the effect of applying MixUp [50] and CutMix [46] together with our proposed token labeling. Empirically, we have found that using MixUp together with token labeling brings no benefit to the performance, and thus we do not apply it in our experiments.

For optimization, by default, we use the AdamW optimizer [27] with a linear learning rate scaling strategy $lr = 10^{-3} \times \frac{batch_size}{640}$ and 5×10^{-2} weight decay rate. For Dropout regularization, we observe that for small models, using Dropout hurts the performance. This has also been observed in a few other works related to training vision transformers [34, 35, 44]. As a result, we do not apply Dropout [31] and use Stochastic Depth [23] instead. More details on hyper-parameters and finetuning can be found in our supplementary materials.

We use the NFNet-F6 [3] trained on ImageNet with an 86.3% Top-1 accuracy as the machine annotator to generate dense score maps for the ImageNet dataset, yielding a 1000-dimensional score map for each image for training. The score map generation procedure is similar to [47], but we limit our experiment setting by training all models from scratch on ImageNet without extra data support, such as JFT-300M and ImageNet-22K. This is different from the original ReLabel paper [47], in which the EfficientNet-L2 model pretrained on JFT-300M is used. The input resolution for NFNet-F6 is 576×576, and the dimension of the corresponding output score map for each image is $L \in \mathbb{R}^{18 \times 18 \times 1000}$. During training, the target labels for the tokens are generated by applying RoIAlign [17] on the corresponding score map. In practice, we only store the top-5 score maps for each position in half-precision to save space as storing the entire score maps for all the images results in 2TB storage. In our experiment, we only need 10GB of storage to store all the score maps.

4.2 Ablation Analysis

Model Settings: The default settings of the proposed LV-ViT are given in Table 1, where both token labeling and MixToken are used. A slight architecture modification to ViT [15] is that we replace the patch embedding module with a 4-layer convolution to better tokenize the input image and integrate local information. Detailed ablation about patch embedding can be found in our supplementary materials. As can be seen, our LV-ViT-T with only 8.5M parameters can already achieve a top-1 accuracy of 79.1% on ImageNet. Increasing the embedding dimension and network depth can further boost the performance. More experiments compared to other methods can be found in Sec. 4.3. In the following ablation experiments, we will set our LV-ViT-S as baseline and show the advantages of the proposed token labeling and MixToken methods.

Table 1: Performance of the proposed LV-ViT with different model sizes. Here, 'depth' denotes the number of transformer blocks used in different models. By default, the test resolution is set to 224×224 except the last one which is 288×288.

Name	Depth	Embed dim.	MLP Ratio	#Heads	#Parameters	Resolution	Top-1 Acc. (%)
LV-ViT-T	12	240	3.0	4	8.5M	224x224	79.1
LV-ViT-S	16	384	3.0	6	26M	224x224	83.3
LV-ViT-M	20	512	3.0	8	56M	224x224	84.1
LV-ViT-L	24	768	3.0	12	150M	288x288	**85.3**

MixToken: We use MixToken as a substitution for CutMix while applying token labeling. Our experiments show that MixToken performs better than CutMix for token-based transformer models. As shown in Table 2, when training with the original ImageNet labels, using MixToken is 0.1% higher than using CutMix. When using the ReLabel supervision, we can also see an advantage of 0.2% over the CutMix baseline. Combining with our token labeling, the performance can be further raised to 83.3%.

Table 2: Ablation on the proposed MixToken and token labeling augmentations. We also show results with either the ImageNet hard label and the ReLabel [47] as supervision.

Aug. Method	Supervision	Top-1 Acc.
MixToken	Token labeling	**83.3**
MixToken	ReLabel	83.0
CutMix	ReLabel	82.8
Mixtoken	ImageNet Label	82.5
CutMix	ImageNet Label	82.4

Table 3: Ablation on different widely-used data augmentations. We have empirically found our proposed MixToken performs even better than the combination of MixUp and CutMix in vision transformers.

MixToken	MixUp	CutOut	RandAug	Top-1 Acc.
✓	✗	✓	✓	**83.3**
✗	✗	✓	✓	81.3
✓	✓	✓	✓	83.1
✓	✗	✗	✓	83.0
✓	✗	✓	✗	82.8

Data Augmentation: Here, we study the compatibility of MixToken with other augmentation techniques, such as MixUp [50], CutOut [55] and RandAug [10]. The ablation results are shown in Table 3. We can see when all the four augmentation methods are used, a top-1 accuracy of 83.1% is achieved. Interestingly, when the MixUp augmentation is removed, the performance can be improved to 83.3%. This may be explained as, using MixToken and MixUp at the same time would bring too much noise in the label, and consequently cause confusion of the model. Moreover, the CutOut augmentation, which randomly erases some parts of the image, is also effective and removing it brings a performance drop of 0.3%. Similarly, the RandAug augmentation also contributes to the performance and using it brings an improvement of 0.5%.

All Tokens Matter: To show the importance of involving all tokens in our token labeling method, we attempt to randomly drop some tokens and use the remaining ones for computing the token labeling loss. The percentage of the remaining tokens is denoted as Token Participation Rate. As shown in Figure 4 (Left), we conduct experiments on two models: LV-ViT-S and LV-ViT-M. As can be seen, using only 20% of the tokens to compute the token labeling loss decreases the performance (-0.5% for LV-ViT-S and -0.4% for LV-ViT-M). Involving more tokens for loss computation consistently leads to better performance. Since involving all tokens brings negligible computation cost and gives the best performance, we always set the token participation rate as 100% in the following experiments.

Robustness to Different Annotators: To evaluate the robustness of our token labeling method, we use different pretrained CNNs, including EfficientNet-B3,B4,B5,B6,B7,B8 [33], NFNet-F6 [3] and ResNest269E [49], as annotator models to provide dense supervision. Results are shown in the right part of Figure 4. We can see that, even if we use an annotator with relatively lower performance, such as EfficientNet-B3 whose Top-1 accuracy is 81.6%, it can still provide multi-label location-specific supervision and help improve the performance of our LV-ViT-S model. Meanwhile, annotator models with better performance can provide more accurate supervision, bringing even better performance, as stronger annotator models can generate better token-level labels. The largest annotator NFNet-F6 [3], which has the best performance of 86.3%, allows us to achieve the best result for LV-ViT-S, which is 83.3%. In addition, we also attempt to use a better model, EfficientNet-L2 pretrained on JFT-300M as described in [47] which has 88.2% Top-1 ImageNet accuracy, as our annotator. The performance

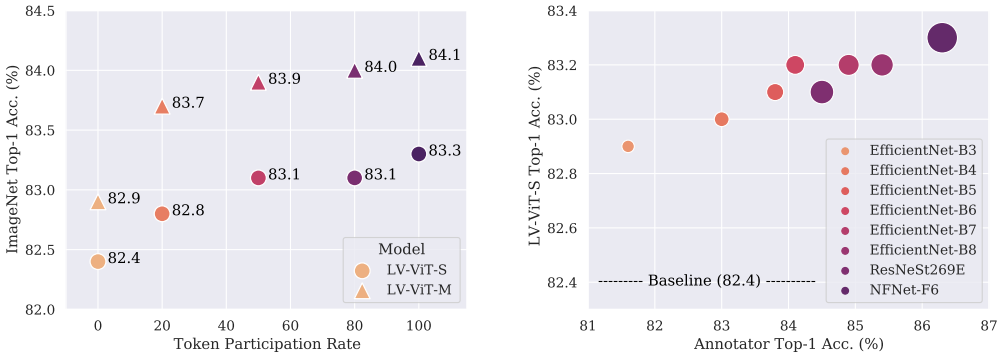

Figure 4: **Left**: LV-ViT ImageNet Top-1 Accuracy w.r.t. the token participation rate while applying token labeling. Token participation rate indicates the percentage of patch tokens involved in computing the token labeling loss. This experiment reflects that all tokens matter for vision transformers. **Right**: LV-ViT-S ImageNet Top-1 Accuracy w.r.t. different annotator models. The point size indicates the parameter number of the annotator model. Clearly, our token labeling objective is robust to different annotator models.

of LV-ViT-S can be further improved to 83.5%. However, to fairly compare with the models without extra training data, we only report results based on dense supervision produced by NFNet-F6 [3] that uses only ImageNet training data.

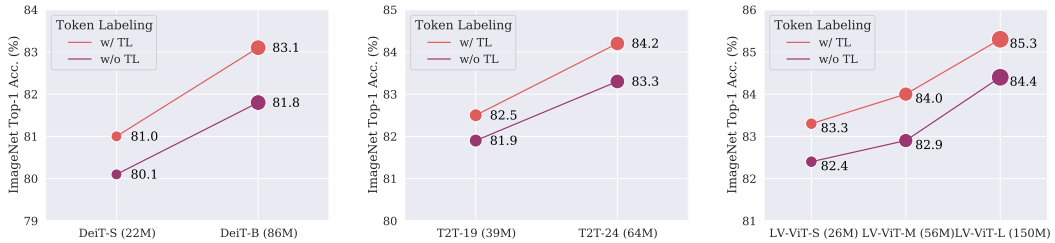

Figure 5: Performance of the proposed token labeling objective on three different vision transformers: DeiT [34] (**Left**), T2T-ViT [44] (**Middle**), and LV-ViT (**Right**). Our method has a consistent improvement on all 7 different ViT models.

Robustness to Different ViT Variants: To further evaluate the robustness of our token labeling, we train different transformer-based networks, including DeiT [34], T2T-ViT [3] and our model LV-ViT, with the proposed training objective. Results are shown in Figure 5. It can be found that, all the models trained with token labeling consistently outperform their vanilla counterparts, demonstrating the robustness of token labeling with respect to different variants of patch-based vision transformers. Meanwhile, for different scales of the models, the improvement is also consistent. Interestingly, we observe larger improvements for larger models. These indicate that our proposed token labeling method is widely applicable to a large range of patch-based vision transformer variants.

4.3 Comparison to Other Methods

We compare our proposed model LV-ViT with other state-of-the-art methods in Table 4. For small-sized models, when the test resolution is set to 224×224, we achieve an 83.3% accuracy on ImageNet with only 26M parameters, which is 3.4% higher than the strong baseline DeiT-S [34]. For medium-sized models, when the test resolution is set to 384×384 we achieve the performance of 85.4%, the same as CaiT-S36 [35], but with much less computational cost and parameters. Note that both DeiT and CaiT use knowledge distillation to improve their models, which introduce much more computations in training. However, we do not require any extra computations in training and only have to compute and store the dense score maps in advance. For large-sized models, our LV-ViT-L

Table 4: Top-1 accuracy comparison with other methods on ImageNet [13] and ImageNet Real [2]. All models are trained without external data. With the same computation and parameter constraint, our model consistently outperforms other CNN-based and transformer-based counterparts. The results of CNNs and ViT are referenced from [35].

	Network	Params	FLOPs	Train size	Test size	Top-1(%)	Real Top-1 (%)
CNNs	EfficientNet-B5 [33]	30M	9.9B	456	456	83.6	88.3
	EfficientNet-B7 [33]	66M	37.0B	600	600	84.3	_
	Fix-EfficientNet-B8 [33, 36]	87M	89.5B	672	800	85.7	90.0
	NFNet-F3 [3]	255M	114.8B	320	416	85.7	89.4
	NFNet-F4 [3]	316M	215.3B	384	512	85.9	89.4
	NFNet-F5 [3]	377M	289.8B	416	544	86.0	89.2
Transformers	ViT-B/16 [15]	86M	55.4B	224	384	77.9	83.6
	ViT-L/16 [15]	307M	190.7B	224	384	76.5	82.2
	T2T-ViT-14 [44]	22M	5.2B	224	224	81.5	_
	T2T-ViT-14↑384 [44]	22M	17.1B	224	384	83.3	_
	CrossViT [6]	45M	56.6B	224	480	84.1	_
	Swin-B [25]	88M	47.0B	224	384	84.2	_
	TNT-B [16]	66M	14.1B	224	224	82.8	_
	DeepViT-S [57]	27M	6.2B	224	224	82.3	_
	DeepViT-L [57]	55M	12.5B	224	224	83.1	_
	DeiT-S [34]	22M	4.6B	224	224	79.9	85.7
	DeiT-B [34]	86M	17.5B	224	224	81.8	86.7
	DeiT-B↑384 [34]	86M	55.4B	224	384	83.1	87.7
	BoTNet-S1-128 [30]	79.1M	19.3B	256	256	84.2	-
	BoTNet-S1-128↑384 [30]	79.1M	45.8B	256	384	84.7	-
	CaiT-S36↑384 [35]	68M	48.0B	224	384	85.4	89.8
	CaiT-M36 [35]	271M	53.7B	224	224	85.1	89.3
	CaiT-M36↑448 [35]	271M	247.8B	224	448	86.3	90.2
Ours LV-ViT	LV-ViT-S	26M	6.6B	224	224	83.3	88.1
	LV-ViT-S↑384	26M	22.2B	224	384	84.4	88.9
	LV-ViT-M	56M	16.0B	224	224	84.1	88.4
	LV-ViT-M↑384	56M	42.2B	224	384	85.4	89.5
	LV-ViT-L	150M	59.0B	288	288	85.3	89.3
	LV-ViT-L↑448	150M	157.2B	288	448	85.9	89.7
	LV-ViT-L↑448	150M	157.2B	448	448	86.2	89.9

with a test resolution of 448×448 achieves an 86.2% top-1 accuracy, which is comparable to CaiT-M36 [35] but with far fewer FLOPs and parameters.

4.4 Semantic Segmentation on ADE20K

It has been shown in [19] that different training techniques for pretrained models have different impacts on downstream tasks with dense prediction, like semantic segmentation. To demonstrate the advantage of the proposed token labeling objective on tasks with dense prediction, we apply our pretrained LV-ViT with token labeling to the semantic segmentation task.

Similar to previous work [25], we run experiments on the widely-used ADE20K [56] dataset. ADE20K contains 25K images in total, including 20K images for training, 2K images for validation and 3K images for test, and covering 150 different foreground categories. We take both FCN [26] and UperNet [42] as our segmentation frameworks and use the mmseg toolbox to implement. During training, following [25], we use 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 are trained on 8 GPUs and with a batch size of 16 (i.e., 2 images on each GPU). The input resolution is set to 512×512. In inference, a multi-scale test with interpolation rates of [0.75, 1.0, 1.25, 1.5, 1.75] is used. As suggested by [56], we report results in terms of both mean intersection-over-union (mIoU) and the average pixel accuracy (Pixel Acc.).

In Table 5, we test the performance of token labeling on both FCN and UperNet frameworks. The FCN framework has a light convolutional head and can directly reflect the performance of the pretrained models in terms of transferable capability. As can be seen, pretrained models with token

Table 5: Transfer performance of the proposed LV-ViT in semantic segmentation. We take two classic methods, FCN and UperNet, as segmentation architectures and show both single-scale (SS) and multi-scale (MS) results on the validation set.

Method	Token Labeling	Model Size	mIoU (SS)	P. Acc. (SS)	mIoU (MS)	P. Acc. (MS)
LV-ViT-S + FCN	✗	30M	46.1	81.9	47.3	82.6
LV-ViT-S + FCN	✓	30M	47.2	82.4	48.4	83.0
LV-ViT-S + UperNet	✗	44M	46.5	82.1	47.6	82.7
LV-ViT-S + UperNet	✓	44M	47.9	82.6	48.6	83.1

labeling perform better than those without token labeling. This indicates token labeling is indeed beneficial to semantic segmentation.

We also compare our segmentation results with previous state-of-the-art segmentation methods in Table 6. Without pretraining on large-scale datasets such as ImageNet-22K, our LV-ViT-M with the UperNet segmentation architecture achieves an mIoU score of 50.6 with only 77M parameters. This result is much better than the previous CNN-based and transformer-based models. Furthermore, using our LV-ViT-L as the pretrained model yields a better result of 51.8 in terms of mIoU. As far as we know, this is the best result reported on ADE20K with no pretraining on ImageNet-22K or other large-scale datasets.

Table 6: Comparison with previous work on ADE20K validation set. As far as we know, our LV-ViT-L + UperNet achieves the best result on ADE20K with only ImageNet-1K as training data in pretraining. †Pretrained on ImageNet-22K.

	Backbone	Segmentation Architecture	Model Size	mIoU (MS)	Pixel Acc. (MS)
CNNs	ResNet-269	PSPNet [52]	-	44.9	81.7
	ResNet-101	UperNet [42]	86M	44.9	-
	ResNet-101	Strip Pooling [22]	-	45.6	82.1
	ResNeSt200	DeepLabV3+ [8]	88M	48.4	-
Transformers	DeiT-S	UperNet	52M	44.0	-
	ViT-Large†	SETR [54]	308M	50.3	83.5
	Swin-T [25]	UperNet	60M	46.1	-
	Swin-S [25]	UperNet	81M	49.3	-
	Swin-B [25]	UperNet	121M	49.7	-
	Swin-B† [25]	UperNet	121M	51.6	-
LV-ViT	LV-ViT-S	FCN	30M	48.4	83.0
	LV-ViT-S	UperNet	44M	48.6	83.1
	LV-ViT-M	UperNet	77M	50.6	83.5
	LV-ViT-L	UperNet	209M	**51.8**	**84.1**

5 Conclusions and Discussion

In this paper, we introduce a new token labeling method to help improve the performance of vision transformers. We also analyze the effectiveness and robustness of our token labeling with respect to different annotators and different variants of patch-based vision transformers. By applying token labeling, our proposed LV-ViT achieves 84.4% Top-1 accuracy with only 26M parameters and 86.2% Top-1 accuracy with 150M parameters on ImageNet-1K benchmark.

Despite the effectiveness, token labeling has a limitation of requiring a pretrained model as the machine annotator. Fortunately, the machine annotating procedure can be done in advance to avoid introducing extra computational cost in training. This makes our method quite different from knowledge distillation methods that rely on online teaching. For users with limited machine resources on hand, our token labeling provides a promising training technique to improve the performance of vision transformers.

References

[1] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *arXiv preprint arXiv:2102.08602*, 2021.

[2] 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.

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

[4] 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.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020.

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

[7] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *arXiv preprint arXiv:2012.00364*, 2020.

[8] 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.

[9] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

[10] 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.

[11] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. *arXiv preprint arXiv:2011.09094*, 2020.

[12] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*, 2021.

[13] 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.

[14] 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.

[15] 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.

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

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[18] 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.

[19] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

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

[28] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

[29] 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.

[30] 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.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[32] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris Kitani. Rethinking transformer-based set prediction for object detection. *arXiv preprint arXiv:2011.10881*, 2020.

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

[34] 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.

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

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

[37] 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.

[38] 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.

[39] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. *arXiv preprint arXiv:2011.14503*, 2020.

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

[41] 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.

[42] 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.

[43] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5791–5800, 2020.

[44] 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.

[45] 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.

[46] 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.

[47] Sangdoo Yun, Seong Joon Oh, Byeongho Heo, Dongyoon Han, Junsuk Choe, and Sanghyuk Chun. Re-labeling imagenet: from single to multi-labels, from global to localized labels. *arXiv preprint arXiv:2101.05022*, 2021.

[48] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *European Conference on Computer Vision*, pages 528–543. Springer, 2020.

[49] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Muller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.

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

[51] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020.

[52] 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.

[53] Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *arXiv preprint arXiv:2011.09315*, 2020.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8739–8748, 2018.

[59] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please refer to Sec. 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We use the widely-used publicly available dataset.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We use the widely-used publicly available dataset.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]