Patch-level Representation Learning for Self-supervised Vision Transformers

Sukmin Yun Hankook Lee Jaehyung Kim Jinwoo Shin
Korea Advanced Institute of Science and Technology (KAIST)
{sukmin.yun, hankook.lee, jaehyungkim, jinwoos}@kaist.ac.kr

Abstract

Recent self-supervised learning (SSL) methods have shown impressive results in learning visual representations from unlabeled images. This paper aims to improve their performance further by utilizing the architectural advantages of the underlying neural network, as the current state-of-the-art visual pretext tasks for SSL do not enjoy the benefit, i.e., they are architecture-agnostic. In particular, we focus on Vision Transformers (ViTs), which have gained much attention recently as a better architectural choice, often outperforming convolutional networks for various visual tasks. The unique characteristic of ViT is that it takes a sequence of disjoint patches from an image and processes patch-level representations internally. Inspired by this, we design a simple yet effective visual pretext task, coined SelfPatch, for learning better patch-level representations. To be specific, we enforce invariance against each patch and its neighbors, i.e., each patch treats similar neighboring patches as positive samples. Consequently, training ViTs with SelfPatch learns more semantically meaningful relations among patches (without using human-annotated labels), which can be beneficial, in particular, to downstream tasks of a dense prediction type. Despite its simplicity, we demonstrate that it can significantly improve the performance of existing SSL methods for various visual tasks, including object detection and semantic segmentation. Specifically, SelfPatch significantly improves the recent self-supervised ViT, DINO, by achieving +1.3 AP on COCO object detection, +1.2 AP on COCO instance segmentation, and +2.9 mIoU on ADE20K semantic segmentation.

1. Introduction

Recently, self-supervised learning (SSL) has achieved successful results in learning visual representations from unlabeled images with a variety of elaborate pretext tasks, including contrastive learning [5, 6, 14], clustering [3], and pseudo-labeling [4, 7, 13]. The common nature of their designs is on utilizing different augmentations from the same image as the *positive pairs*, *i.e.*, they learn representations to be invariant to the augmentations. The SSL approaches without utilizing human-annotated labels have been competitive with or even outperformed the standard supervised learning [16] in various downstream tasks, including image classification [6], object detection [3], and segmentation [3].

Meanwhile, motivated by the success of Transformers [29] in natural language processing [1, 10], Vision Transformers (ViTs) [11, 27, 28] have gained much attention as an alternative to convolutional neural networks (CNNs) with superior performance over CNNs in various visual tasks [25,27]. For example, ViT-S/16 [27] has a $1.8\times$ faster throughput than ResNet-152 [16] with higher accuracy in the ImageNet [9] benchmark.

There have been several recent attempts to apply existing self-supervision techniques to ViTs [4, 8, 36]. Although the techniques have shown to be also effective with ViTs, they do not fully utilize the architectural advantages of ViTs, *i.e.*, their pretext tasks are architecture-agnostic. For example, ViTs are able to process patch-level representations, but pretext tasks used in the existing SSL schemes only use the whole image-level self-supervision without considering learning patch-level representations. As a result, existing self-supervised ViTs [4, 8] may fail to capture semantically meaningful relations among patches; *e.g.*, collapsed self-attention maps of among patches as shown in the second row of Fig. 1. This limitation inspires us to investigate the following question: *how to utilize architectural characteristics of ViTs for improving the quality of learned patch-level representations without human-annotated supervision?*

Contribution. In this paper, we propose a simple yet effective SSL scheme for learning patch-level representations, coined *SelfPatch*, which can be beneficial to various visual downstream tasks. Our patch-level SSL scheme, SelfPatch, can be incorporated into any image-level self-supervised ViT, *e.g.*, DINO [4], MoCo-v3 [8] and MoBY [36], for learning both global (*i.e.*, image-level) and local (*i.e.*, patch-level) information simultaneously. Fig. 1 shows that SelfPatch enhances the quality of self-attention maps of DINO, which is evidence that SelfPatch encourages to learn better patch-level representations.

Our key idea is to treat *semantically similar neighboring*

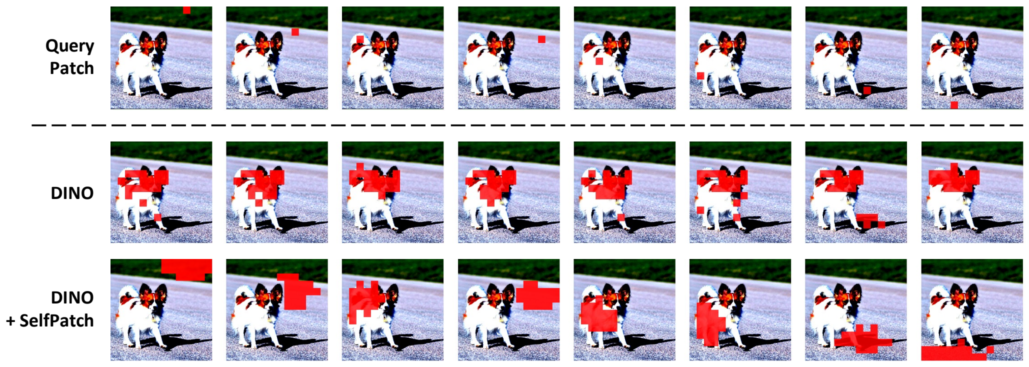


Figure 1. Visualization of top-10% patches obtained by thresholding the self-attention maps of query patches (top) in the last layer of ViT-S/16 trained with DINO (middle) and DINO + SelfPatch (bottom). While the selected patches obtained by DINO are not semantically correlated with its query patch, SelfPatch encourages the model to learn semantic correlations among patches.

patches as positive samples motivated by the following prior knowledge: adjacent patches often share a common semantic context. Since there might be multiple positive patches (but we do not know exactly which patches are positive), we first select a fixed number of adjacent patches for *positive* candidates using the cosine similarity between patch representations of the current model. Here, some of them might still be noisy (*e.g.*, not positive), and for the purpose of denoising, we summarize their patch representations by utilizing an additional attention-based aggregation module on the top of ViT. Then, we minimize the distance between each patch representation and the corresponding summarized one. We provide an overall illustration of the proposed scheme in Fig. 2a.

To demonstrate the effectiveness of our method, we pretrain ViT-S/16 [27] on the ImageNet [9] dataset and evaluate transferring performance on a wide range of dense prediction downstream tasks: (a) COCO object detection and instance segmentation [20], (b) ADE20K semantic segmentation [38], and (c) DAVIS 2017 video object segmentation [24]. Specifically, our method improves the state-of-the-art image-level self-supervision, DINO [4], with a large margin, *e.g.*, +1.3 AP^{bb} (*i.e.*, 40.8 $\rightarrow$ 42.1) on COCO detection (see Tab. 1), and +2.9 mIoU (*i.e.*, 38.3 $\rightarrow$ 41.2) on ADE20K segmentation (see Tab. 2). As a result, our method outperforms all the SSL baselines [3, 6, 31, 33, 34, 36] such as DenseCL [31]. Furthermore, we demonstrate high compatibility of SelfPatch by incorporating with various objectives (*e.g.*, MoBY [36]), architectures (*e.g.*, Swin Transformer [21]), and patch sizes (*e.g.*, 8×8). For example, SelfPatch improves MoBY [36] for both ViT and Swin Transformer by +4.8 and +5.6 $(\mathcal{J}\&\mathcal{F})_m$, respectively, on the DAVIS video segmentation benchmark [24] (see Tab. 5).

Overall, our work highlights the importance of learning patch-level representations during pre-training ViTs in a self-supervised manner. We hope that ours could inspire researchers to rethink the under-explored problem and provide a new direction of patch-level self-supervised learning.

2. Method

In this section, we introduce a simple yet effective visual pretext task, coined *SelfPatch*, for learning better patch-level representations, which is tailored to Vision Transformers [11] for utilizing their unique architectural advantages. We first review Vision Transformers with recent self-supervised learning schemes [4, 8, 36] in Sec. 2.1 and then present details of SelfPatch in Sec. 2.2. Fig. 2a illustrates the overall scheme of our method, SelfPatch.

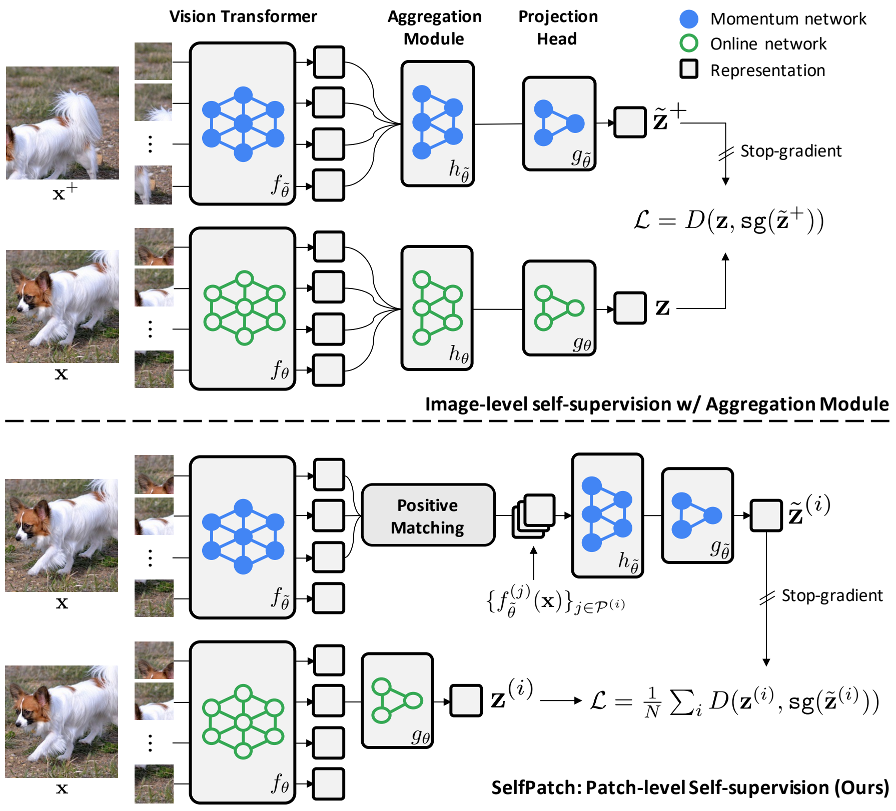
2.1. Preliminaries

Vision Transformers. Let $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ be an image where (H, W) is the resolution of $\mathbf{x}$ and C is the number of channels. Vision Transformers (ViTs) [11] treat the image $\mathbf{x}$ as a sequence of non-overlapping patches $\{\mathbf{x}^{(i)} \in \mathbb{R}^{P^2C}\}_{i=1}^{N}$ (*i.e.*, tokens) where each patch has a fixed resolution (P, P). Then, the patches are linearly transformed to D-dimensional patch embeddings $\mathbf{e}^{(i)} = E\mathbf{x}^{(i)} + E_{\text{pos}}^{(i)} \in \mathbb{R}^D$ where $E \in \mathbb{R}^{D\times P^2C}$ is a linear projection and $E_{\text{pos}}^{(i)} \in \mathbb{R}^D$ is a positional embedding for the patch index i. ViTs also prepend the [CLS] token, which represents the entire patches (*i.e.,* the given image $\mathbf{x}$), to the patch sequence with a learnable embedding $\mathbf{e}^{[\texttt{CLS}]} \in \mathbb{R}^D$. The resulting input sequence $\mathbf{e}$ is $\mathbf{e} = [\mathbf{e}^{[\texttt{CLS}]}; \mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \ldots; \mathbf{e}^{(N)}]$. Then, ViTs take the input $\mathbf{e}$ and output all the patch-level and image-level (*i.e.*, [CLS] token) representations with a transformer encoder.[1] For conciseness, we use f_θ to denote the whole process of a ViT parameterized by θ[2] as follows:

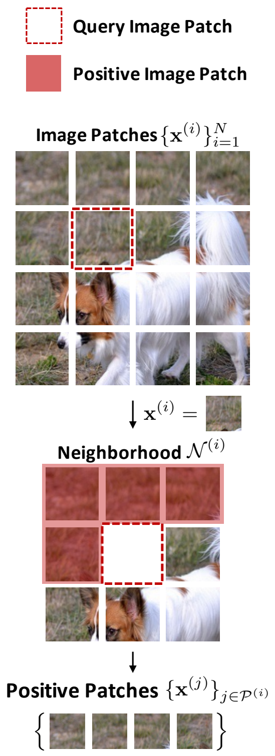
$$\begin{aligned} f_\theta(\mathbf{x}) &= f_\theta([\mathbf{e}^{[\texttt{CLS}]}; \mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \ldots; \mathbf{e}^{(N)}]) \\ &= [f_\theta^{[\texttt{CLS}]}(\mathbf{x}); f_\theta^{(1)}(\mathbf{x}); f_\theta^{(2)}(\mathbf{x}); \ldots; f_\theta^{(N)}(\mathbf{x})], \quad (1) \end{aligned}$$

[1]We omit the details of the transformer encoder [29] of which each layer consists of the self-attention module, skip connection, and multi-layer perceptron (MLP).

[2]Note that θ contains all the transformer encoder parameters and embedding parameters E, E_{pos}, and $\mathbf{e}^{[\texttt{CLS}]}$.



(a) Illustration of the proposed patch-level self-supervision (SelfPatch)


Query Image Patch
Positive Image Patch
Image Patches $\{\mathbf{x}^{(i)}\}_{i=1}^N$
Neighborhood $\mathcal{N}^{(i)}$
Positive Patches $\{\mathbf{x}^{(j)}\}_{j\in\mathcal{P}^{(i)}}$

(b) Positive matching

Figure 2. (a) **Top**: image-level self-supervision, which minimizes the distance between the final representations of two differently augmented images. **Bottom**: patch-level self-supervision, SelfPatch, which minimizes the distance between the final representations of each patch and its positives. We use both types of self-supervision for learning image-level and patch-level representations simultaneously. (b) For a given query patch, we find semantically similar patches from its neighborhood using the cosine similarity on the representation space.

where $f_\theta^{[\texttt{CLS}]}(\mathbf{x})$ and $f_\theta^{(i)}(\mathbf{x})$ are the final representations of the [CLS] token and the i-th patch, respectively. Remark that $f_\theta^{[\texttt{CLS}]}(\mathbf{x})$ is utilized for solving image-level downstream tasks such as image classification [11, 27, 28] while the patch-level representations $\{f_\theta^{(i)}(\mathbf{x})\}_{i=1}^N$ are done for dense prediction tasks, *e.g.*, object detection [2] and semantic segmentation [35].

Self-supervised learning with ViTs. The recent literature [4, 8, 36] has attempted to apply self-supervised learning (SSL) schemes to ViTs. They commonly construct a positive pair $(\mathbf{x}, \mathbf{x}^+)$ by applying different augmentations to the same image, and then learn their representations to be similar, *i.e.*, invariant to augmentations. We here provide a generic formulation of this idea.[3] To this end, we denote two ViT backbone networks as f_θ and $f_{\tilde{\theta}}$, and their projection heads as g_θ and $g_{\tilde{\theta}}$ are parametrized by θ and $\tilde{\theta}$, respectively. Then, the generic form of *image-level* self-supervision $\mathcal{L}^{\text{SSL}}(\{\mathbf{x}, \mathbf{x}^+\}; \theta, \tilde{\theta})$ can be written as follows:

$$\mathcal{L}^{\text{SSL}} := D(g_\theta(f_\theta^{[\texttt{CLS}]}(\mathbf{x})), \mathsf{sg}(g_{\tilde{\theta}}(f_{\tilde{\theta}}^{[\texttt{CLS}]}(\mathbf{x}^+)))), \quad (2)$$

where D is a distance function and sg is the stop-gradient operation. Note that the choices of distance D and architecture g depend on a type of self-supervision; for example, Caron *et al.* [4] update $\tilde{\theta}$ by the exponential moving average of θ, and use Kullback-Leibler (KL) divergence as D, where the projection heads $g_{\tilde{\theta}}$ and g_θ are designed to produce a probability distribution over the final feature dimension.

We remark that the idea of constructing a positive pair $(\mathbf{x}, \mathbf{x}^+)$ of augmented *images* is architecture-agnostic, which means it does not fully utilize the architectural benefits of ViTs. For example, ViTs can handle patch-level representations $\{f_\theta^{(i)}(\mathbf{x})\}$, but the recent SSL schemes use only $f_\theta^{[\texttt{CLS}]}(\mathbf{x})$ as described in Eq. (2). This motivates us to explore the following question: *how to construct patch-level self-supervision, i.e., positive pairs of patches?*

[3] Built upon this formulation, one can additionally consider negative pairs for contrastive learning or asymmetric architectures such as a prediction head [8, 36].

2.2. Patch-level self-supervision

Recall that our goal is to learn better patch-level representations, which can be beneficial to various downstream tasks. Our key idea is to consider neighboring patches as positive samples based on the continuous nature of image patches. Specifically, SelfPatch aims to learn patch-level representations via predicting self-supervision constructed by the following procedure: for each patch $\mathbf{x}^{(i)}$, *Positive Matching* first finds a set of candidates for positive patch indices $\mathcal{P}^{(i)}$ from its *neighborhood* $\mathcal{N}^{(i)}$ (see Fig. 2b), and then *Aggregation Module* aggregates their representations $\{f_\theta^{(j)}(\mathbf{x})\}_{j\in\mathcal{P}^{(i)}}$ as self-supervision for $\mathbf{x}^{(i)}$ (see Fig. 2a).

Neighboring patches. Given a query patch $\mathbf{x}^{(i)}$, we assume that there exists a semantically similar patch $\mathbf{x}^{(j)}$ in its neighborhood $\mathcal{N}^{(i)}$, because neighboring patches $\{\mathbf{x}^{(j)}\}_{j\in\mathcal{N}^{(i)}}$ often share a semantic context with the query $\mathbf{x}^{(i)}$. Let $\{\mathbf{x}^{(j)}\}_{j\in\mathcal{N}^{(i)}}$ be a set of neighboring patches, where $\mathcal{N}^{(i)}$ is a set of patch indices in the neighborhood. We simply use $\mathcal{N}^{(i)}$ to be adjacent patches (*i.e.,* $|\mathcal{N}^{(i)}| = 8$), and empirically found that this choice is important for selecting candidates for positive patches (see Sec. 5.1 for analysis on the importance of neighboring patches).

Matching positives from the neighborhood. To sample positive (*i.e.*, semantically similar) patches from the neighborhood $\mathcal{N}^{(i)}$, we measure the semantic closeness between the query patch $\mathbf{x}^{(i)}$ and its neighboring patch $\mathbf{x}_\theta^{(j)}$ for all $j \in \mathcal{N}^{(i)}$. To this end, we use the cosine similarity on the representation space, *i.e.*,

$$s(i,j) = f_\theta^{(i)}(\mathbf{x})^\top f_\theta^{(j)}(\mathbf{x}) / ||f_\theta^{(i)}(\mathbf{x})||_2 ||f_\theta^{(j)}(\mathbf{x})||_2. \quad (3)$$

We take *top-k* positive patches $\{\mathbf{x}^{(j)}\}_{j\in\mathcal{P}^{(i)}}$ based on the similarity scores $s(i,j)$, where $\mathcal{P}^{(i)}$ is a set of patch indices of *top-k* patches in $\mathcal{N}^{(i)}$. We use $k = |\mathcal{P}^{(i)}| = 4$ in our experiments (see Sec. 5.1 for analysis on the effect of k).

Aggregation module. We aggregate positive patches $\{\mathbf{x}^{(j)}\}_{j\in\mathcal{P}^{(i)}}$ using an aggregation module h_θ to construct patch-level self-supervision $\mathbf{y}_\theta^{(i)}$ for each patch $\mathbf{x}^{(i)}$. We utilize h_θ for not only patch representations (*i.e.*, $\mathbf{y}_\theta^{(i)}$), but also the image-level representation (*i.e.*, $\mathbf{y}_\theta$). Formally,

$$\mathbf{y}_\theta := h_\theta(\{f_\theta^{(j)}(\mathbf{x})\}_{j=1}^N), \quad (4)$$

$$\mathbf{y}_\theta^{(i)} := h_\theta(\{f_\theta^{(j)}(\mathbf{x})\}_{j\in\mathcal{P}^{(i)}}). \quad (5)$$

We follow Touvron *et al.* [28] to implement the aggregation module h_θ. To be specific, h_θ is a small Transformer that takes a sequence of representations as an input and outputs a representation of the [CLS] token. We found that the aggregation module h_θ is crucial for generating better self-supervision (see Sec. 5.1 for ablation experiments).

Training objective. Our training objective $\mathcal{L}^{\text{total}}(\mathbf{x},\mathbf{x}^+)$ consists of the *image-level* and *patch-level* self-supervision objectives, $\mathcal{L}^{\text{SSL}}(\mathbf{x},\mathbf{x}^+)$ and $\mathcal{L}^{\text{SelfPatch}}(\mathbf{x})$, as follows:

$$\mathcal{L}^{\text{SSL}}(\mathbf{x},\mathbf{x}^+) := D(g_\theta(\mathbf{y}_\theta), \texttt{sg}(g_{\tilde\theta}(\mathbf{y}_{\tilde\theta}))), \quad (6)$$

$$\mathcal{L}^{\text{SelfPatch}}(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^N D(g'_\theta(f_\theta^{(i)}(\mathbf{x})), \texttt{sg}(g'_{\tilde\theta}(\mathbf{y}_{\tilde\theta}^{(i)}))), \quad (7)$$

$$\mathcal{L}^{\text{total}}(\mathbf{x},\mathbf{x}^+) := \mathcal{L}^{\text{SSL}}(\mathbf{x},\mathbf{x}^+) + \lambda\mathcal{L}^{\text{SelfPatch}}(\mathbf{x}), \quad (8)$$

where D is a distance function, sg is the stop-gradient operation, λ is a hyperparameter, g and g' are projection heads for $\mathcal{L}^{\text{SSL}}$ and $\mathcal{L}^{\text{SelfPatch}}$, respectively. Here, our patch-level objective $\mathcal{L}^{\text{SelfPatch}}(\mathbf{x})$ enforces a query patch $\mathbf{x}^{(i)}$ and its positives $\{\mathbf{x}^{(j)}\}_{j\in\mathcal{P}^{(i)}}$ to be similar by minimizing the distance between the patch and aggregated representations, $f_\theta^{(i)}(\mathbf{x})$ and $\mathbf{y}_{\tilde\theta}^{(i)}$, respectively. We remark that $\mathcal{L}^{\text{SelfPatch}}(\mathbf{x})$ has an asymmetric form: a query representation $f_\theta^{(i)}(\mathbf{x})$ does not use the aggregation module h_θ while its (aggregated) target $\mathbf{y}_{\tilde\theta}^{(i)}$ does. We empirically found that this architectural asymmetry with the stop-gradient operation effectively avoids the mode collapse issue which comes from minimizing the distance between patch representations.

3. Related works

Transformer-based architectures for images. Vision Transformer (ViT) [11,27] is a pioneering architecture built on top of Transformer [29] for large-scale vision tasks such as ImageNet [9] classification by splitting an image into a sequence of patch images. Inspired by the success of the Vision Transformers, a number of variants [12,17,21,22,30,32,37] have been developed. They commonly incorporate convolutional designs into ViTs, *e.g.*, a spatial down-sampling operation [22,32] or a hierarchical structure that considers various spatial resolutions [21,30].

Self-supervised learning. For learning visual representations from a large number of unlabeled images, self-supervised learning (SSL) [3–6,8,13,14,36] has become a remarkable research direction as it can be effectively transferred to various downstream tasks like image classification. The common design of their pretext tasks is to learn visual representations by maximizing the similarity between augmented images originated from the same image. Recently, there have been several attempts to apply such SSL pretext tasks to ViTs [4,8,36]. Meanwhile, some of studies have focused on to design SSL schemes tailored for dense prediction tasks [23,31,33,34]. For example, ReSim [33] matches overlapping regions (*i.e.*, sub-images have the same location) between two augmented images, while our method matches neighboring regions (*i.e.,* image patches) within the same augmented image. We remark that utilizing adjacent patches as the positives can be viewed as a reasonable way to find meaningful positives without constraints of overlapping regions between augmented images.

				Detection			Segmentation		
Method	Backbone	Epoch	Param.(M)	AP^{bb}	$\text{AP}^{\text{bb}}_{50}$	$\text{AP}^{\text{bb}}_{75}$	AP^{mk}	$\text{AP}^{\text{mk}}_{50}$	$\text{AP}^{\text{mk}}_{75}$
MoCo-v2 [6]	ResNet50	200	26	38.9	59.2	42.4	35.5	56.2	37.8
SwAV [3]	ResNet50	200	26	38.5	60.4	41.4	35.4	57.0	37.7
DenseCL [31]	ResNet50	200	26	40.3	59.9	44.3	36.4	57.0	39.2
ReSim [33]	ResNet50	200	26	40.3	60.6	44.2	36.4	57.5	38.9
DetCo [34]	ResNet50	200	26	40.1	61.0	43.9	36.4	58.0	38.9
MoCo-v3 [8]	ViT-S/16	300	22	39.8	62.6	43.1	37.1	59.6	39.2
MoBY [36]	ViT-S/16	300	22	41.1	63.7	44.8	37.6	60.3	39.8
DINO [4]	ViT-S/16	300	22	40.8	63.4	44.2	37.3	59.9	39.5
+ SelfPatch (ours)	ViT-S/16	200	22	**42.1**	**64.9**	**46.1**	**38.5**	**61.3**	**40.8**

Table 1. **COCO object detection and instance segmentation** performances of the recent self-supervised approaches pre-trained on ImageNet. The metrics AP^{bb} and AP^{mk} denote bounding box and mask average precision (AP), respectively.

4. Experiments

In this section, we evaluate the effectiveness of the proposed self-supervised learning scheme, *SelfPatch*, through extensive large-scale experiments. Specifically, we compare SelfPatch with existing self-supervised learning (SSL) approaches in various dense prediction tasks:[4] (a) COCO object detection and segmentation [20] (Sec. 4.1), (b) ADE20K segmentation [38] (Sec. 4.2), and (c) DAVIS video object segmentation [24] (Sec. 4.3). More details of experimental setups are described in each section and the supplementary material.

Baselines. We consider a variety of existing SSL methods developed for ResNet [16] and ViT [27] architectures: (a) self-supervised ResNets: MoCo-v2 [6], SwAV [3], DenseCL [31], ReSim [33], and DetCo [34]; and (b) self-supervised ViTs: DINO [4], MoCo-v3 [8], and MoBY [36]. We use ViT-S/16 [27] (22M parameters) and ResNet50 [16] (26M parameters) since they are conventional choices and have the similar number of parameters. We denote our method built upon an existing method by "+ SelfPatch", *e.g.,* DINO + SelfPatch. We use publicly available ImageNet pre-trained models for the baselines; the public ResNet models [3, 6, 31, 33, 34] and ViT models [4, 8, 36] are pre-trained for 200 and 300 epochs, respectively.

Implementation details. We incorporate the SelfPatch objective with the state-of-the-art SSL scheme, DINO [4], as described in Eq. (8), where we pre-train ViT-S/16 on ImageNet for 200 epochs with a batch size of 1024. We follow DINO's training details (*e.g.*, optimizer, learning rate schedule). For the distance function D in Eq. (7), we use the KL divergence in $\mathcal{L}^{\text{SelfPatch}}$ following DINO. We use $\lambda = 0.1$ unless stated otherwise (see Sec. 5.3 and Tab. 6 for hyper-parameter analysis). The other training details are provided in the supplementary material.

It is worth noting that our choice of epoch is due to fair comparisons with ResNet and ViT baselines; when pre-training ViT-S/16, our additional self-supervision increases (around 21%) training time per epoch and we consider a smaller number of epoch for DINO + SelfPatch, compared to that of DINO for a fair comparison. Nevertheless, we expect that our method can be improved further if trained by 300 epochs as like other public ViT baselines [4, 8, 36].

[4]Although our major focus is on dense prediction tasks, we also evaluate transferring performance to ImageNet linear classification task [9], and observe that our method achieves competitive performance compared to the state-of-the-art methods as presented in the supplementary material.

4.1. COCO object detection and segmentation

Setup. We evaluate pre-trained models on the COCO object detection and instance segmentation tasks [20]. Here, all models are fine-tuned with Mask R-CNN [15] and FPN [19] under the standard `1x` schedule.

Results. Tab. 1 shows that our SelfPatch consistently improves DINO in both detection and segmentation tasks, and consequently, DINO + SelfPatch outperforms all the baselines. For example, the bounding box average precision (*i.e.*, AP^{bb}) of DINO + SelfPatch is 1.3 points higher than that of DINO. One can find similar results in the segmentation task; for example, DINO + SelfPatch achieves 38.5 mask average precision (*i.e.*, AP^{mk}), which is 1.2 points higher than DINO, and also 2.1 points higher than the best ResNet-based baseline, DenseCL. We also emphasize that the improvements from DINO + SelfPatch are even achieved with a smaller number of pre-training epochs (*i.e.,* 200 epochs) than DINO. These results demonstrate that the advantages of our method are not only high performance, but also training efficiency (*e.g.*, DINO + SelfPatch is 1.24× faster than DINO in total running time).

4.2. ADE20K semantic segmentation

Setup. We evaluate semantic segmentation performances of pre-trained models on ADE20K [38], which contains 150 fine-grained semantic categories and 25k training data. All models are fine-tuned with Semantic FPN [18] under the

Method	Backbone	mIoU	aAcc	mAcc
MoCo-v2 [6]	ResNet50	35.8	77.6	45.1
SwAV [3]	ResNet50	35.4	77.5	44.9
DenseCL [31]	ResNet50	37.2	78.5	47.1
ReSim [33]	ResNet50	36.6	78.4	46.4
DetCo [34]	ResNet50	37.3	78.4	46.7
MoCo-v3 [8]	ViT-S/16	35.3	78.9	45.9
MoBY [36]	ViT-S/16	39.5	79.9	50.5
DINO [4]	ViT-S/16	38.3	79.0	49.4
+ SelfPatch (ours)	ViT-S/16	**41.2**	**80.7**	**52.1**

Table 2. **ADE20K semantic segmentation** performances of the recent self-supervised approaches pre-trained on ImageNet. The metrics mIoU, aAcc, and mAcc denote mean intersection of union, all pixel accuracy, and mean class accuracy, respectively.

standard 40k iteration schedule. We report three metrics: (a) mean intersection of union (mIoU) averaged over all semantic categories, (b) all pixel accuracy (aAcc), and (c) mean class accuracy (mAcc).

Results. As shown in Tab. 2, DINO + SelfPatch achieves significant improvements over DINO in all the metrics; *e.g.,* DINO + SelfPatch achieves 2.9 and 2.7 points higher than DINO in terms of the mIoU and mAcc metrics, respectively. Also, DINO + SelfPatch consistently outperforms all the baselines; *e.g.,* in terms of the mIoU metric, our method achieves 41.2 points, while DetCO and MoBY do 37.3 and 39.5 points, respectively. These comparisons across the architectures verify the effectiveness of SelfPatch.

4.3. DAVIS video object segmentation

Setup. We perform video object segmentation using pre-trained models on DAVIS 2017 [24]. We follow the evaluation protocol in Caron *et al.* [4], which does not require extra training costs. To be specific, it evaluates the quality of frozen representations of image patches by segmenting scenes with the nearest neighbor between consecutive frames. We report three evaluation metrics: (a) mean region similarity $\mathcal{J}_m$, (b) mean contour-based accuracy $\mathcal{F}_m$, and (c) their average score $(\mathcal{J}\&\mathcal{F})_m$.

Results. In Tab. 3, DINO + SelfPatch does not only consistently improve DINO, but also largely surpasses the other baselines. For example, SelfPatch improves the $(\mathcal{J}\&\mathcal{F})_m$ score of DINO from 60.7 to 62.7, while SwAV and MoBY achieve only 57.4 and 54.7, respectively. We present visualizations of video object segmentation results obtained by DINO and DINO + SelfPatch in Fig. 3, and it shows that SelfPatch clearly enhances the video segmentation quality. From these observations, we confirm that our method encourages the model to produce more meaningful segmentation maps, which also demonstrate the effectiveness of our scheme for learning better patch-level representations.

Method	Backbone	$(\mathcal{J}\&\mathcal{F})_m$	$\mathcal{J}_m$	$\mathcal{F}_m$
MoCo-v2 [6]	ResNet50	55.5	56.0	55.0
SwAV [3]	ResNet50	57.4	57.6	57.3
DenseCL [31]	ResNet50	50.7	52.6	48.9
ReSim [33]	ResNet50	49.3	51.2	47.3
DetCo [34]	ResNet50	56.7	57.0	56.4
MoCo-v3 [8]	ViT-S/16	53.5	51.2	55.9
MoBY [36]	ViT-S/16	54.7	52.0	57.3
DINO [4]	ViT-S/16	60.7	59.1	62.4
+ SelfPatch (ours)	ViT-S/16	**62.7**	**60.7**	**64.7**

Table 3. **DAVIS 2017 video object segmentation** performances of the recent self-supervised approaches pre-trained on ImageNet. The metrics $\mathcal{J}_m$, $\mathcal{F}_m$, and $(\mathcal{J}\&\mathcal{F})_m$ denote mean region similarity, contour-based accuracy, and their average, respectively.

	Neighbors $\mathcal{N}^{(i)}$	Matching	Agg	$(\mathcal{J}\&\mathcal{F})_m$
(a)	-	-	-	55.1
(b)	3×3	$k=4$	✓	**57.0**
	5×5	$k=4$	✓	56.5
	All patches	$k=4$	✓	47.3
(c)	3×3	$k=1$	✓	56.3
	3×3	$k=2$	✓	56.4
	3×3	$k=4$	✓	**57.0**
	3×3	$k=8$	✓	56.5
(d)	3×3	$k=4$	-	51.4
	3×3	$k=4$	✓	**57.0**

Table 4. **Ablation studies on component contributions** of SelfPatch: (a) the baseline (*i.e.*, DINO [4] without SelfPatch), (b) the neighboring patches ("Neighbors $\mathcal{N}^{(i)}$"), (c) positive matching ("Matching"), and (d) aggregation module ("Agg"). All models are pre-trained on COCO [20] and evaluated on DAVIS 2017 [24].

5. Ablation study

In this section, we perform ablation studies to understand further how does SelfPatch works. To this end, we pre-train all cases using ViT-Ti/16 on the MS COCO `train2017` dataset for 200 epochs with a batch size of 256 and evaluate the pre-trained models on the DAVIS 2017 benchmark [24].

5.1. Component analysis

Tab. 4 shows the individual effects of SelfPatch's components: (b) neighboring patches ("Neighbors" column), (c) positive matching ("Matching" column), and (d) the aggregation module ("Agg" column). Note that (a) shows the performance of the baseline, *i.e.*, DINO [4] without SelfPatch. In what follows, we analyze their effects in detail.

Effect of neighboring patches. We investigate the importance of neighboring patches $\mathcal{N}^{(i)}$ for selecting positive patches $\mathcal{P}^{(i)}$ for each patch $\mathbf{x}^{(i)}$. To this end, we test three

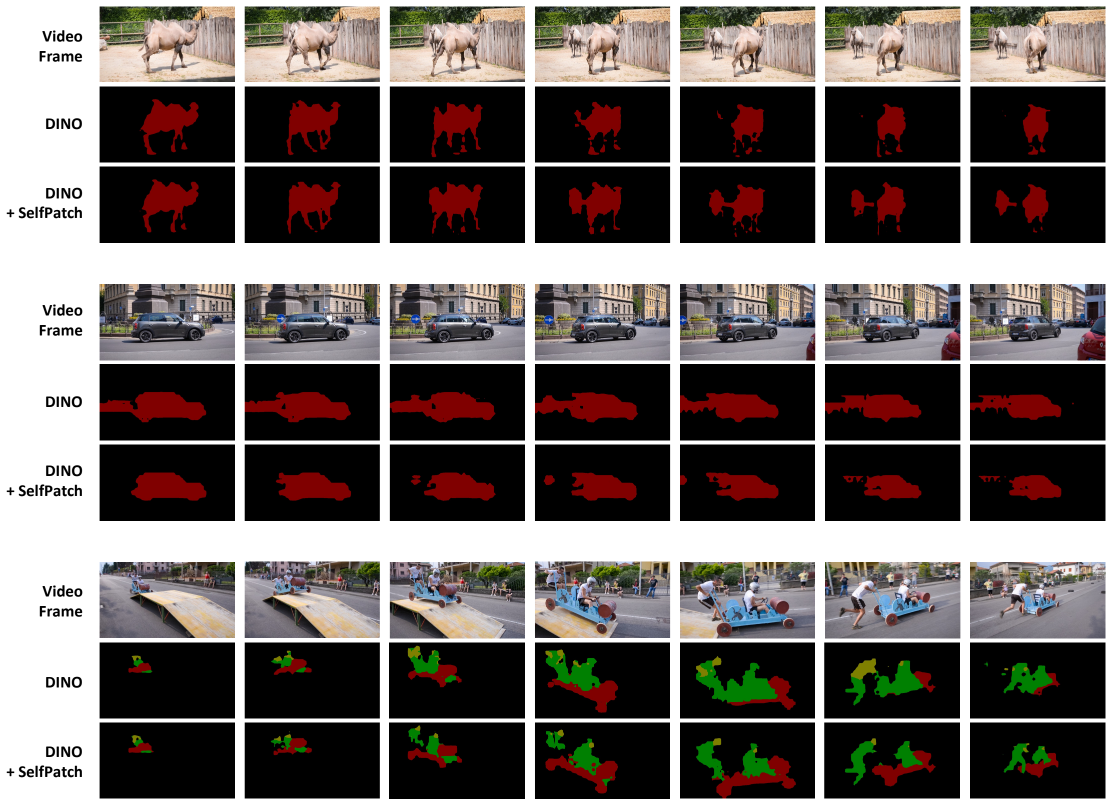


Figure 3. Visualization of video segmentation results on the DAVIS 2017 [24] benchmark. **Top**: input video frames. **Middle** and **Bottom**: segmentation results obtained by DINO and DINO + SelfPatch (ours), respectively. SelfPatch clearly improves the segmentation results, which is evidence that SelfPatch encourages the patch representations to learn semantic information of each object. Best viewed in color.

different types of neighbors: 3×3 or 5×5 patches around $\mathbf{x}^{(i)}$, or all patches (*e.g.*, 14×14 patches in the input resolution of 224×224) in the given image $\mathbf{x}$. As shown in the (b) group in Tab. 4, considering patches far from $\mathbf{x}^{(i)}$ is not helpful for selecting positives $\mathcal{P}^{(i)}$; in particular, considering all patches shows even worse performance than the baseline (*i.e.*, 55.1 $\rightarrow$ 47.3) on the video segmentation task. This validates that restricting positive candidates as neighboring patches is a more reasonable way to find more reliable positive patches than considering far patches.

Effect of positive matching. Our positive matching process selects *top-k* patches as positives $\mathcal{P}^{(i)}$ among the neighbors $\mathcal{N}^{(i)}$ using the cosine similarity on the representation space. To demonstrate the effectiveness of this process, we vary the number of positive patches, $k = |\mathcal{P}^{(i)}|$. As shown in the (c) group in Tab. 4, any $k \in \{1, 2, 4, 8\}$ shows better performance than (a) the baseline. Also, selecting $k = 4$ patches among the neighborhood in the positive matching module shows the best performance (*i.e.*, 57.0 $(\mathcal{J}\&\mathcal{F})_m$) compared to selecting all neighboring patches (*i.e.*, $k = 8$) and few neighboring patches (*i.e.*, $k \in \{1, 2\}$). This result implies that some of the neighboring patches might not be positive (*i.e.*, noisy), but a moderate number of them could be considered as positives.

Effect of aggregation module. We here validate the contribution of our aggregation module h_θ which aims at aggregating multiple patch representations. To this end, we simply replace the aggregation module with the average pooling operation. Note that this pooling operation utilizes patch representations equally while our aggregation module can summarize them in an adaptive manner. Hence, even if a noisy (*e.g.*, not positive) patch exists among the selected positives $\mathcal{P}^{(i)}$, our module h_θ could reduce the negative effect from the noises. The (d) group in Tab. 4 shows that the aggregation module is a crucial component of our scheme.

5.2. Compatibility analysis

We here validate the compatibility of our method with (a) another image-level SSL scheme, MoBY [36], and (b) another Transformer-based architecture, Swin Transformer [21]. Note that MoBY [36] is a contrastive method that utilizes negative pairs, and Swin Transformer [21] is a variant of ViT, which has a hierarchical structure. Tab. 5 summarizes the compatibility experiments.

As shown in Tab. 5, our method is well-incorporated with MoBY [36]. For example, SelfPatch improves MoBY by 4.8 (*i.e.*, 54.1 $\rightarrow$ 58.9) and 5.6 (*i.e.*, 50.8 $\rightarrow$ 56.4) points when using ViT-Ti/16 and Swin-T architectures, re-

Method	Backbone	Negative	$(\mathcal{J}\&\mathcal{F})_m$
MoBY	ViT-Ti/16	-	54.1
+ SelfPatch (ours)	ViT-Ti/16	-	58.4
+ SelfPatch (ours)	ViT-Ti/16	✓	**58.9**
MoBY	Swin-T	-	50.8
+ SelfPatch (ours)	Swin-T	✓	**56.4**
DINO	ViT-Ti/16	-	55.1
+ SelfPatch (ours)	ViT-Ti/16	-	**57.0**
DINO	ViT-Ti/8	-	61.6
+ SelfPatch (ours)	ViT-Ti/8	-	**65.8**

Table 5. **Ablation studies on compatibility** with various SSL schemes (DINO [4], MoBY [36]), architectures (ViT [27], Swin Transformer [21]), and patch sizes (16×16, 8×8). All models are pre-trained on COCO [20] and evaluated on DAVIS 2017 [24]. We denote the use of negative pairs for our method as "Negative".

spectively. Since MoBY is a contrastive method, we also construct negative pairs for our loss $\mathcal{L}^{\text{SelfPatch}}$ by sampling global representations (*i.e.*, Eq. (4)) of different images. Interestingly, even under such a contrastive method, SelfPatch is not required to use negative pairs: SelfPatch without negatives achieves the competitive performance compared to SelfPatch with the negatives (58.4 *vs.* 58.9). Since the current negative construction for patches is straightforward, we believe that an elaborate design for the negatives might further improve the performance. We leave it for future work.

We also found that our method, SelfPatch, is working with a smaller size (*i.e.*, 8×8) of input patches as shown in the last two rows in Tab. 5. Note that using such smaller patches is beneficial to solving dense prediction tasks since their final representations are more fine-grained. As a result, DINO + SelfPatch with ViT-Ti/8 [27] achieves the best performance in Tab. 5. These results demonstrate the high compatibility of SelfPatch with various SSL schemes, architectures, and patch sizes.

5.3. Hyperparameter analysis

In Tab. 6, we examine the effect of the loss weight λ in Eq. (8) across an array of $\lambda \in \{0, 0.1, 1\}$. We observed that the performance of $\lambda = 0$ (*i.e.,* 55.3 $(\mathcal{J}\&\mathcal{F})_m$) shows the almost similar to the baseline, DINO (*i.e.,* 55.1 $(\mathcal{J}\&\mathcal{F})_m$). It shows that the improvement of our method comes from the patch-level self-supervision in Eq. (7), not the architectural modification (*i.e.,* adding the aggregation module). As shown in the last two rows in Tab. 6, any $\lambda \in \{0.1, 1\}$ shows better performance than the baselines. Based on this result, we conducted all experiments in Sec. 4 and 5 with $\lambda = 0.1$. Remark that this choice of $\lambda = 0.1$ shows consistent improvements across architectures (ViT [11, 27] and Swin Transformer [21]), and SSL objectives (DINO [4] and MoBY [36]) as shown in Sec. 5.2.

Method	λ	$(\mathcal{J}\&\mathcal{F})_m$	$\mathcal{J}_m$	$\mathcal{F}_m$
DINO	-	55.1	52.8	57.4
+ SelfPatch (ours)	0.0	55.3	53.1	57.6
	0.1	**57.0**	**56.1**	**57.8**
	1.0	56.4	55.5	57.4

Table 6. **Ablation studies on varying the loss weight λ** for SelfPatch objective. All models are pre-trained on COCO [20] and evaluated on DAVIS 2017 [24].

6. Discussion

Conclusion. We propose a simple yet effective patch-level self-supervised learning scheme (SelfPatch) for improving visual representations of individual image patches. Our key idea is to treat semantically similar neighboring patches as positives. More specifically, we select semantically similar (*i.e.*, positive) patches from the neighborhood, and then enforce invariance against each patch and its positives by minimizing the distance of their final representations. Through the extensive experiments, we demonstrate the effectiveness of our scheme in various downstream tasks, including object detection and semantic segmentation. We believe that this work would guide many research directions for self-supervised learning.

Limitation. In this work, we focus on constructing *patch-level* self-supervision. However, there might be other types of self-supervision, *e.g.*, more coarse-grained (*i.e.*, region-level) one. Hence, an interesting future research direction would be to develop a universal self-supervision for learning all types (*e.g.*, patch, region, and image-level) of representations. Even for this direction, we believe that our idea of treating neighbors as positives could be utilized.

Potential negative societal impact. Due to the absence of supervision, self-supervised learning often requires a large number of training samples (*e.g.*, DALL-E [26] requires 250M image-text pair of data) or a large network capacity (*e.g.*, GPT-3 [1] requires 175B parameters). Such an enormous computational cost may cause environmental problems such as carbon generation or global warming. Hence, developing various types (*e.g.*, pixel- or patch-level) of self-supervision for efficient training would be required to alleviate the problems.

Acknowledgements. We thank Sangwoo Mo for providing helpful feedbacks and suggestions. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub; No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)) and the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921).

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901, 2020. 1, 8

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 3

[3] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020. 1, 2, 4, 5, 6

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021. 1, 2, 3, 4, 5, 6, 8

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1, 4

[6] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 2, 4, 5, 6

[7] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 1

[8] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021. 1, 2, 3, 4, 5, 6

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 2, 4, 5

[10] 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. 1

[11] 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. 1, 2, 3, 4, 8

[12] Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. *arXiv preprint arXiv:2104.01136*, 2021. 4

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020. 1, 4

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 4

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 5

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 5

[17] 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. 4

[18] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 5

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 5

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 5, 6, 8

[21] 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. 2, 4, 7, 8

[22] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable visual transformers with hierarchical pooling. *arXiv preprint arXiv:2103.10619*, 2021. 4

[23] Pedro O Pinheiro, Amjad Almahairi, Ryan Y Benmalek, Florian Golemo, and Aaron Courville. Unsupervised learning of dense visual representations. In *NeurIPS*, 2020. 4

[24] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 2, 5, 6, 7, 8

[25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 1

[26] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021. 8

[27] 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. 1, 2, 3, 4, 5, 8

[28] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 1, 3, 4

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 1, 2, 4

[30] 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. 4

[31] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3024–3033, 2021. 2, 4, 5, 6

[32] 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. 4

[33] Tete Xiao, Colorado J Reed, Xiaolong Wang, Kurt Keutzer, and Trevor Darrell. Region similarity representation learning. In *ICCV*, 2021. 2, 4, 5, 6

[34] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. In *ICCV*, 2021. 2, 4, 5, 6

[35] Enze Xie, Wenjia Wang, Wenhai Wang, Peize Sun, Hang Xu, Ding Liang, and Ping Luo. Segmenting transparent object in the wild with transformer. *arXiv preprint arXiv:2101.08461*, 2021. 3

[36] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers. *arXiv preprint arXiv:2105.04553*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[37] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021. 4

[38] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 2, 5