May 2021

★ DINO – self-distillation with NO Labels.

Idea ViT – needs a lot of data.

Emerging Properties in Self-Supervised Vision Transformers

Mathilde Caron[1,2] Hugo Touvron[1,3] Ishan Misra[1] Hervé Jegou[1]
Julien Mairal[2] Piotr Bojanowski[1] Armand Joulin[1]

[1] Facebook AI Research [2] Inria* [3] Sorbonne University

FFI

Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the [CLS] token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

Abstract

In this paper, we question if self-supervised learning provides new properties to Vision Transformer (ViT) [19] that stand out compared to convolutional networks (convnets). Beyond the fact that adapting self-supervised methods to this architecture works particularly well, we make the following observations: first, self-supervised ViT features contain explicit information about the semantic segmentation of an image, which does not emerge as clearly with supervised ViTs, nor with convnets. Second, these features are also excellent k-NN classifiers, reaching 78.3% top-1 on ImageNet with a small ViT. Our study also underlines the importance of momentum encoder [33], multi-crop training [10], and the use of small patches with ViTs. We implement our findings into a simple self-supervised method, called DINO, which we interpret as a form of self-distillation with no labels. We show the synergy between DINO and ViTs by achieving 80.1% top-1 on ImageNet in linear evaluation with ViT-Base.

Combine

Compact

*Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France.
Correspondence: mathilde@fb.com
Code: https://github.com/facebookresearch/dino

1. Introduction

Transformers [70] have recently emerged as an alternative to convolutional neural networks (convnets) for visual recognition [19, 69, 83]. Their adoption has been coupled with a training strategy inspired by natural language processing (NLP), that is, pretraining on large quantities of data and finetuning on the target dataset [18, 55]. The resulting Vision Transformers (ViT) [19] are competitive with convnets but, they have not yet delivered clear benefits over them: they are computationally more demanding, require more training data, and their features do not exhibit unique properties.

In this paper, we question whether the muted success of Transformers in vision can be explained by the use of supervision in their pretraining. Our motivation is that one of the main ingredients for the success of Transformers in NLP was the use of self-supervised pretraining, in the form of close procedure in BERT [18] or language modeling in GPT [55]. These self-supervised pretraining objectives use the words in a sentence to create pretext tasks that provide a richer learning signal than the supervised objective of predicting a single label per sentence. Similarly, in images, image-level supervision often reduces the rich visual information contained in an image to a single concept selected from a predefined set of a few thousand categories of objects [60].

While the self-supervised pretext tasks used in NLP are

text specific, many existing self-supervised methods have shown their potential on images with convnets [10, 12, 30, 33]. They typically share a similar structure but with different components designed to avoid trivial solutions (collapse) or to improve performance [16]. In this work, inspired from these methods, we study the impact of self-supervised pretraining on ViT features. Of particular interest, we have identified several interesting properties that do not emerge with supervised ViTs, nor with convnets:

- Self-supervised ViT features explicitly contain the scene layout and, in particular, object boundaries, as shown in Figure 1. This information is directly accessible in the self-attention modules of the last block.
- Self-supervised ViT features perform particularly well with a basic nearest neighbors classifier (k-NN) *without any finetuning, linear classifier nor data augmentation*, achieving 78.3% top-1 accuracy on ImageNet.

The emergence of segmentation masks seems to be a property shared across self-supervised methods. However, the good performance with k-NN only emerge when combining certain components such as momentum encoder [33] and multi-crop augmentation [10]. Another finding from our study is the importance of using smaller patches with ViTs to improve the quality of the resulting features.

Overall, our findings about the importance of these components lead us to design a simple self-supervised approach that can be interpreted as a form of knowledge distillation [35] with **no** labels. The resulting framework, DINO, simplifies self-supervised training by directly predicting the output of a teacher network—built with a momentum encoder—by using a standard cross-entropy loss. Interestingly, our method can work with only a centering and sharpening of the teacher output to avoid collapse, while other popular components such as predictor [30], advanced normalization [10] or contrastive loss [33] add little benefits in terms of stability or performance. Of particular importance, our framework is flexible and works on both convnets and ViTs without the need to modify the architecture, nor adapt internal normalizations [58].

We further validate the synergy between DINO and ViT by outperforming previous self-supervised features on the ImageNet linear classification benchmark with 80.1% top-1 accuracy with a ViT-Base with small patches. We also confirm that DINO works with convnets by matching the state of the art with a ResNet-50 architecture. Finally, we discuss different scenarios to use DINO with ViTs in case of limited computation and memory capacity. In particular, training DINO with ViT takes just two 8-GPU servers over 3 days to achieve 76.1% on ImageNet linear benchmark, which outperforms self-supervised systems based on convnets of comparable sizes with significantly reduced compute requirements [10, 30].

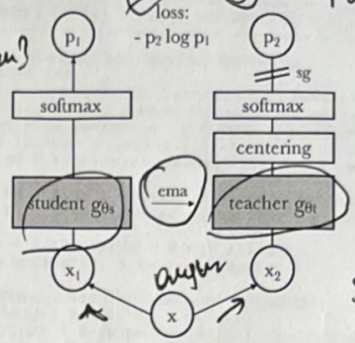


Figure 2: **Self-distillation with no labels.** We illustrate DINO in the case of one single pair of views (x_1, x_2) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters. The output of the teacher network is centered with a mean computed over the batch. Each networks outputs a K dimensional feature that is normalized with a temperature softmax over the feature dimension. Their similarity is then measured with a cross-entropy loss. We apply a stop-gradient (sg) operator on the teacher to propagate gradients only through the student. The teacher parameters are updated with an exponential moving average (ema) of the student parameters.

2. Related work

Self-supervised learning. A large body of work on self-supervised learning focuses on discriminative approaches coined *instance classification* [12, 20, 33, 73], which considers each image a different class and trains the model by discriminating them up to data augmentations. However, explicitly learning a classifier to discriminate between all images [20] does not scale well with the number of images. Wu *et al.* [73] propose to use a noise contrastive estimator (NCE) [32] to compare instances instead of classifying them. A caveat of this approach is that it requires comparing features from a large number of images simultaneously. In practice, this requires large batches [12] or memory banks [33, 73]. Several variants allow automatic grouping of instances in the form of clustering [2, 8, 9, 36, 42, 74, 80, 85].

Recent works have shown that we can learn unsupervised features without discriminating between images. Of particular interest, Grill *et al.* [30] propose a metric-learning formulation called BYOL, where features are trained by matching them to representations obtained with a momentum encoder. Methods like BYOL work even without a momentum encoder, at the cost of a drop of performance [16, 30]. Several other works echo this direction, showing that one can match more elaborate representations [26, 27], train features matching them to a uniform distribution [6] or by using whitening [23, 81]. Our approach takes its inspiration from BYOL but operates with a different similarity matching loss

and uses the exact same architecture for the student and the teacher. That way, our work completes the interpretation initiated in BYOL of self-supervised learning as a form of Mean Teacher self-distillation [65] with no labels.

Self-training and knowledge distillation. Self-training aims at improving the quality of features by propagating a small initial set of annotations to a large set of unlabeled instances. This propagation can either be done with hard assignments of labels [41, 78, 79] or with a soft assignment [76]. When using soft labels, the approach is often referred to as knowledge distillation [7, 35] and has been primarily designed to train a small network to mimic the output of a larger network to compress models. Xie *et al.* [76] have shown that distillation could be used to propagate soft pseudo-labels to unlabelled data in a self-training pipeline, drawing an essential connection between self-training and knowledge distillation. Our work builds on this relation and extends knowledge distillation to the case where no labels are available. Previous works have also combined self-supervised learning and knowledge distillation [25, 63, 13, 47], enabling self-supervised model compression and performance gains. However, these works rely on a *pre-trained* fixed teacher while our teacher is dynamically built during training. This way, knowledge distillation, instead of being used as a post-processing step to self-supervised pre-training, is directly cast as a self-supervised objective. Finally, our work is also related to *codistillation* [1] where student and teacher have the same architecture and use distillation during training. However, the teacher in *codistillation* is also distilling from the student, while it is updated with an average of the student in our work.

3. Approach

3.1. SSL with Knowledge Distillation

The framework used for this work, DINO, shares the same overall structure as recent self-supervised approaches [10, 16, 12, 30, 33]. However, our method shares also similarities with knowledge distillation [35] and we present it under this angle. We illustrate DINO in Figure 2 and propose a pseudo-code implementation in Algorithm 1.

Knowledge distillation is a learning paradigm where we train a student network g_{θ_s} to match the output of a given teacher network g_{θ_t}, parameterized by θ_s and θ_t respectively. Given an input image x, both networks output probability distributions over K dimensions denoted by P_s and P_t. The probability P is obtained by normalizing the output of the network g with a softmax function. More precisely,

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K} \exp(g_{\theta_s}(x)^{(k)}/\tau_s)}, \quad (1)$$

with $\tau_s > 0$ a temperature parameter that controls the sharpness of the output distribution, and a similar formula holds for P_t with temperature τ_t. Given a fixed teacher network g_{θ_t}, we learn to match these distributions by minimizing the cross-entropy loss w.r.t. the parameters of the student network θ_s:

$$\min_{\theta_s} H(P_t(x), P_s(x)), \quad (2)$$

where $H(a, b) = -a \log b$.

In the following, we detail how we adapt the problem in Eq. (2) to self-supervised learning. First, we construct different distorted views, or crops, of an image with multi-crop strategy [10]. More precisely, from a given image, we generate a set V of different views. This set contains two *global* views, x_1^g and x_2^g and several *local* views of smaller resolution. All crops are passed through the student while only the *global* views are passed through the teacher, therefore encouraging "local-to-global" correspondences. We minimize the loss:

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{\substack{x' \in V \\ x' \neq x}} H(P_t(x), P_s(x')). \quad (3)$$

This loss is general and can be used on any number of views, even only 2. However, we follow the standard setting for multi-crop by using 2 global views at resolution 224^2 covering a large (for example greater than 50%) area of the original image, and several local views of resolution 96^2 covering only small areas (for example less than 50%) of the original image. We refer to this setting as the basic parametrization of DINO, unless mentioned otherwise.

Both networks share the same architecture g with different sets of parameters θ_s and θ_t. We learn the parameters θ_s by minimizing Eq. (3) with stochastic gradient descent.

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
   x1, x2 = augment(x), augment(x) # random views

   s1, s2 = gs(x1), gs(x2) # student output n-by-K
   t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

   loss = H(t1, s2)/2 + H(t2, s1)/2
   loss.backward() # back-propagate

   # student, teacher and center updates
   update(gs) # SGD
   gt.params = l*gt.params + (1-l)*gs.params
   C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
   t = t.detach() # stop gradient
   s = softmax(s / tps, dim=1)
   t = softmax((t - C) / tpt, dim=1) # center + sharpen
   return - (t * log(s)).sum(dim=1).mean()
```

Table 1: **Networks configuration.** "Blocks" is the number of Transformer blocks, "dim" is channel dimension and "heads" is the number of heads in multi-head attention. "# tokens" is the length of the token sequence when considering 224^2 resolution inputs, "# params" is the total number of parameters (without counting the projection head) and "im/s" is the inference time on a NVIDIA V100 GPU with 128 samples per forward.

model	blocks	dim	heads	#tokens	#params	im/s
ResNet-50	–	2048	–	–	23M	1237
ViT-S/16	12	384	6	197	21M	1007
ViT-S/8	12	384	6	785	21M	180
ViT-B/16	12	768	12	197	85M	312
ViT-B/8	12	768	12	785	85M	63

Teacher network. Unlike knowledge distillation, we do not have a teacher g_{θ_t} given *a priori* and hence, we build it from past iterations of the student network. We study different update rules for the teacher in Section 5.2 and show that freezing the teacher network over an epoch works surprisingly well in our framework, while copying the student weight for the teacher fails to converge. Of particular interest, using an exponential moving average (EMA) on the student weights, i.e., a momentum encoder [33], is particularly well suited for our framework. The update rule is $\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s$, with λ following a cosine schedule from 0.996 to 1 during training [30]. Originally the momentum encoder has been introduced as a substitute for a queue in contrastive learning [33]. However, in our framework, its role differs since we do not have a queue nor a contrastive loss, and may be closer to the role of the mean teacher used in self-training [65]. Indeed, we observe that this teacher performs a form of model ensembling similar to Polyak-Ruppert averaging with an exponential decay [51, 59]. Using Polyak-Ruppert averaging for model ensembling is a standard practice to improve the performance of a model [38]. We observe that this teacher has better performance than the student throughout the training, and hence, guides the training of the student by providing target features of higher quality. This dynamic was not observed in previous works [30, 58].

Network architecture. The neural network g is composed of a backbone f (ViT [19] or ResNet [34]), and of a projection head h: $g = h \circ f$. The features used in downstream tasks are the backbone f output. The projection head consists of a 3-layer multi-layer perceptron (MLP) with hidden dimension 2048 followed by ℓ_2 normalization and a weight normalized fully connected layer [61] with K dimensions, which is similar to the design from SwAV [10]. We have tested other projection heads and this particular design appears to work best for DINO (Appendix C). We do not use a predictor [30, 16], resulting in the exact same architecture in both student and teacher networks. Of particular interest, we note that unlike standard convnets, ViT architectures do not use batch normalizations (BN) by default. Therefore, when applying DINO to ViT we do not use any BN also in the projection heads, making the system *entirely BN-free*.

Avoiding collapse. Several self-supervised methods differ by the operation used to avoid collapse, either through contrastive loss [73], clustering constraints [8, 10], predictor [30] or batch normalizations [30, 58]. While our framework can be stabilized with multiple normalizations [10], it can also work with only a centering and sharpening of the momentum teacher outputs to avoid model collapse. As shown experimentally in Section 5.3, centering prevents one dimension to dominate but encourages collapse to the uniform distribution, while the sharpening has the opposite effect. Applying both operations balances their effects which is sufficient to avoid collapse in presence of a momentum teacher. Choosing this method to avoid collapse trades stability for less dependence over the batch: the centering operation only depends on first-order batch statistics and can be interpreted as adding a bias term c to the teacher: $g_t(x) \leftarrow g_t(x) + c$. The center c is updated with an exponential moving average, which allows the approach to work well across different batch sizes as shown in Section 5.5:

$$c \leftarrow mc + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i), \tag{4}$$

where $m > 0$ is a rate parameter and B is the batch size. Output sharpening is obtained by using a low value for the temperature τ_t in the teacher softmax normalization.

3.2. Implementation and evaluation protocols

In this section, we provide the implementation details to train with DINO and present the evaluation protocols used in our experiments.

Vision Transformer. We briefly describe the mechanism of the Vision Transformer (ViT) [19, 70] and refer to Vaswani *et al.* [70] for details about Transformers and to Dosovitskiy *et al.* [19] for its adaptation to images. We follow the implementation used in `DeiT` [69]. We summarize the configuration of the different networks used in this paper in Table 1. The ViT architecture takes as input a grid of non-overlapping contiguous image patches of resolution $N \times N$. In this paper we typically use $N = 16$ ("/16") or $N = 8$ ("/8"). The patches are then passed through a linear layer to form a set of embeddings. We add an extra learnable token to the sequence [18, 19]. The role of this token is to aggregate information from the entire sequence and we attach the projection head h at its output. We refer to this token as the class token `[CLS]` for consistency with

previous works[18, 19, 69], even though it is not attached to any label nor supervision in our case. The set of patch tokens and [CLS] token are fed to a standard Transformer network with a "pre-norm" layer normalization [11, 39]. The Transformer is a sequence of self-attention and feed-forward layers, paralleled with skip connections. The self-attention layers update the token representations by looking at the other token representations with an attention mechanism [4].

Implementation details. We pretrain the models on the ImageNet dataset [60] without labels. We train with the adamw optimizer [44] and a batch size of 1024, distributed over 16 GPUs when using ViT-S/16. The learning rate is linearly ramped up during the first 10 epochs to its base value determined with the following linear scaling rule [29]: $lr = 0.0005 * \text{batchsize}/256$. After this warmup, we decay the learning rate with a cosine schedule [43]. The weight decay also follows a cosine schedule from 0.04 to 0.4. The temperature τ_s is set to 0.1 while we use a linear warm-up for τ_t from 0.04 to 0.07 during the first 30 epochs. We follow the data augmentations of BYOL [30] (color jittering, Gaussian blur and solarization) and multi-crop [10] with a bicubic interpolation to adapt the position embeddings to the scales [19, 69]. The code and models to reproduce our results is publicly available.

Evaluation protocols. Standard protocols for self-supervised learning are to either learn a linear classifier on frozen features [82, 33] or to finetune the features on downstream tasks. For linear evaluations, we apply random resize crops and horizontal flips augmentation during training, and report accuracy on a central crop. For finetuning evaluations, we initialize networks with the pretrained weights and adapt them during training. However, both evaluations are sensitive to hyperparameters, and we observe a large variance in accuracy between runs when varying the learning rate for example. We thus also evaluate the quality of features with a simple weighted nearest neighbor classifier (k-NN) as in [73]. We freeze the pretrain model to compute and store the features of the training data of the downstream task. The nearest neighbor classifier then matches the feature of an image to the k nearest stored features that votes for the label. We sweep over different number of nearest neighbors and find that 20 NN is consistently working the best for most of our runs. This evaluation protocol does not require any other hyperparameter tuning, nor data augmentation and can be run with only one pass over the downstream dataset, greatly simplifying the feature evaluation.

Table 2: **Linear and k-NN classification on ImageNet.** We report top-1 accuracy for linear and k-NN evaluations on the validation set of ImageNet for different self-supervised methods. We focus on ResNet-50 and ViT-small architectures, but also report the best results obtained across architectures. * are run by us. We run the k-NN evaluation for models with official released weights. The throughput (im/s) is calculated on a NVIDIA V100 GPU with 128 samples per forward. Parameters (M) are of the feature extractor.

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [12]	RN50	23	1237	69.1	60.7
MoCov2 [15]	RN50	23	1237	71.1	61.9
InfoMin [67]	RN50	23	1237	73.0	65.3
BarlowT [81]	RN50	23	1237	73.2	66.0
OBoW [27]	RN50	23	1237	73.8	61.9
BYOL [30]	RN50	23	1237	74.4	64.8
DCv2 [10]	RN50	23	1237	75.2	67.1
SwAV [10]	RN50	23	1237	**75.3**	65.7
DINO	RN50	23	1237	**75.3**	**67.5**
Supervised	ViT-S	21	1007	79.8	79.8
BYOL* [30]	ViT-S	21	1007	71.4	66.6
MoCov2* [15]	ViT-S	21	1007	72.7	64.4
SwAV* [10]	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	**77.0**	**74.5**
Comparison across architectures					
SCLR [12]	RN50w4	375	117	76.8	69.3
SwAV [10]	RN50w2	93	384	77.3	67.3
BYOL [30]	RN50w2	93	384	77.4	–
DINO	ViT-B/16	85	312	78.2	76.1
SwAV [10]	RN50w5	586	76	78.5	67.1
BYOL [30]	RN50w4	375	117	78.6	–
BYOL [30]	RN200w2	250	123	79.6	73.9
DINO	ViT-S/8	21	180	79.7	**78.3**
SCLRv2 [13]	RN152w3+SK	794	46	79.8	73.1
DINO	ViT-B/8	85	63	**80.1**	77.4

4. Main Results

We first validate the DINO framework used in this study with the standard self-supervised benchmark on ImageNet. We then study the properties of the resulting features for retrieval, object discovery and transfer-learning.

4.1. Comparing with SSL frameworks on ImageNet

We consider two different settings: comparison with the same architecture and across architectures.

Comparing with the same architecture. In top panel of Table 2, we compare DINO with other self-supervised methods with the same architecture, either a ResNet-50 [34] or a ViT-small (which follows the design of DeiT-S [69]). The choice of ViT-S is motivated by its similarity with ResNet-50 along several axes: number of parameters (21M vs 23M),

Table 3: **Image retrieval.** We compare the performance in retrieval of off-the-shelf features pretrained with supervision or with DINO on ImageNet and Google Landmarks v2 (GLDv2) dataset. We report mAP on revisited Oxford and Paris. Pretraining with DINO on a landmark dataset performs particularly well. For reference, we also report the best retrieval method with off-the-shelf features [57].

			$\mathcal{R}$Ox		$\mathcal{R}$Par	
Pretrain	Arch.	Pretrain	M	H	M	H
Sup. [57]	RN101+R-MAC	ImNet	49.8	18.5	74.0	**52.1**
Sup.	ViT-S/16	ImNet	33.5	8.9	63.0	37.2
DINO	ResNet-50	ImNet	35.4	11.1	55.9	27.5
DINO	ViT-S/16	ImNet	41.8	13.7	63.1	34.4
DINO	ViT-S/16	GLDv2	**51.5**	**24.3**	**75.3**	51.6

throughput (1237/sec VS 1007 im/sec) and supervised performance on ImageNet with the training procedure of [69] (79.3% VS 79.8%). We explore variants of ViT-S in Appendix D. First, we observe that DINO performs on par with the state of the art on ResNet-50, validating that DINO works in the standard setting. When we switch to a ViT architecture, DINO outperforms BYOL, MoCov2 and SwAV by +3.5% with linear classification and by +7.9% with k-NN evaluation. More surprisingly, the performance with a simple k-NN classifier is almost on par with a linear classifier (74.5% versus 77.0%). This property emerges only when using DINO with ViT architectures, and does not appear with other existing self-supervised methods nor with a ResNet-50.

Comparing across architectures. On the bottom panel of Table 2, we compare the best performance obtained across architectures. The interest of this setting is not to compare methods directly, but to evaluate the limits of a ViT trained with DINO when moving to larger architectures. While training a larger ViT with DINO improves the performance, reducing the size of the patches ("/8" variants) has a bigger impact on the performance. While reducing the patch size do not add parameters, it still leads to a significant reduction of running time, and larger memory usage. Nonetheless, a base ViT with 8×8 patches trained with DINO achieves 80.1% top-1 in linear classification and 77.4% with a k-NN classifier with $10\times$ less parameters and $1.4\times$ faster run time than previous state of the art [13].

4.2. Properties of ViT trained with SSL

We evaluate properties of the DINO features in terms of nearest neighbor search, retaining information about object location and transferability to downstream tasks.

Table 4: **Copy detection.** We report the mAP performance in copy detection on Copydays "strong" subset [21]. For reference, we also report the performance of the multigrain model [5], trained specifically for particular object retrieval.

Method	Arch.	Dim.	Resolution	mAP
Multigrain [5]	ResNet-50	2048	224^2	75.1
Multigrain [5]	ResNet-50	2048	largest side 800	82.5
Supervised [69]	ViT-B/16	1536	224^2	76.4
DINO	ViT-B/16	1536	224^2	81.7
DINO	ViT-B/8	1536	320^2	**85.5**

4.2.1 Nearest neighbor retrieval with DINO ViT

The results on ImageNet classification have exposed the potential of our features for tasks relying on nearest neighbor retrieval. In this set of experiments, we further consolidate this finding on landmark retrieval and copy detection tasks.

Image Retrieval. We consider the revisited [53] Oxford and Paris image retrieval datasets [50]. They contain 3 different splits of gradual difficulty with query/database pairs. We report the Mean Average Precision (mAP) for the Medium (M) and Hard (H) splits. In Table 3, we compare the performance of different *off-the-shelf* features obtained with either supervised or DINO training. We freeze the features and directly apply k-NN for retrieval. We observe that DINO features outperform those trained on ImageNet with labels.

An advantage of SSL approaches is that they can be trained on any dataset, without requiring any form of annotations. We train DINO on the 1.2M clean set from Google Landmarks v2 (GLDv2) [72], a dataset of landmarks designed for retrieval purposes. DINO ViT features trained on GLDv2 are remarkably good, outperforming previously published methods based on off-the-shelf descriptors [68, 57].

Copy detection. We also evaluate the performance of ViTs trained with DINO on a copy detection task. We report the mean average precision on the "strong" subset of the INRIA Copydays dataset [21]. The task is to recognize images that have been distorted by blur, insertions, print and scan, etc. Following prior work [5], we add 10k distractor images randomly sampled from the YFCC100M dataset [66]. We perform copy detection directly with cosine similarity on the features obtained from our pretrained network. The features are obtained as the concatenation of the output `[CLS]` token and of the GeM pooled [54] output patch tokens. This results in a 1536d descriptor for ViT-B. Following [5], we apply whitening on the features. We learn this transformation on an extra 20K random images from YFCC100M, distincts from the distractors. Table 4 shows that ViT trained with DINO is very competitive on copy detection.

Table 5: **DAVIS 2017 Video object segmentation.** We evaluate the quality of frozen features on video instance tracking. We report mean region similarity $\mathcal{J}_m$ and mean contour-based accuracy $\mathcal{F}_m$. We compare with existing self-supervised methods and a supervised ViT-S/8 trained on ImageNet. Image resolution is 480p.

Method	Data	Arch.	$(\mathcal{J}\&\mathcal{F})_m$	$\mathcal{J}_m$	$\mathcal{F}_m$
Supervised					
ImageNet	INet	ViT-S/8	66.0	63.9	68.1
STM [48]	I/D/Y	RN50	81.8	79.2	84.3
Self-supervised					
CT [71]	VLOG	RN50	48.7	46.4	50.0
MAST [40]	YT-VOS	RN18	65.5	63.3	67.6
STC [37]	Kinetics	RN18	67.6	64.8	70.2
DINO	INet	ViT-S/16	61.8	60.2	63.4
DINO	INet	ViT-B/16	62.3	60.7	63.9
DINO	INet	ViT-S/8	**69.9**	**66.6**	**73.1**
DINO	INet	ViT-B/8	**71.4**	**67.9**	**74.9**

Figure 3: **Attention maps from multiple heads.** We consider the heads from the last layer of a ViT-S/8 trained with DINO and display the self-attention for `[CLS]` token query. Different heads, materialized by different colors, focus on different locations that represents different objects or parts (more examples in Appendix).

4.2.2 Discovering the semantic layout of scenes

As shown qualitatively in Figure 1, our self-attention maps contain information about the segmentation of an image. In this study, we measure this property on a standard benchmark as well as by directly probing the quality of masks generated from these attention maps.

Video instance segmentation. In Tab. 5, we evaluate the output patch tokens on the DAVIS-2017 video instance segmentation benchmark [52]. We follow the experimental protocol in Jabri *et al.* [37] and segment scenes with a nearest-neighbor between consecutive frames; we thus do not train any model on top of the features, nor finetune any weights for the task. We observe in Tab. 5 that even though our training objective nor our architecture are designed for dense tasks, the performance is competitive on this benchmark. Since the network is not finetuned, the output of the model must have retained some spatial information. Finally, for this dense recognition task, the variants with small patches ("/8") perform much better (+9.1% $(\mathcal{J}\&\mathcal{F})_m$ for ViT-B).

Probing the self-attention map. In Fig. 3, we show that different heads can attend to different semantic regions of an image, even when they are occluded (the bushes on the third row) or small (the flag on the second row). Visualizations are obtained with 480p images, resulting in sequences of 3601 tokens for ViT-S/8. In Fig. 4, we show that a supervised ViT does not attend well to objects in presence of clutter both qualitatively and quantitatively. We report the Jaccard similarity between the ground truth and segmentation masks obtained by thresholding the self-attention map to keep 60% of the mass. Note that the self-attention maps are smooth and not optimized to produce a mask. Nonetheless, we see a clear difference between the supervised or DINO models with a significant gap in terms of Jaccard similarities. Note that self-supervised convnets also contain information about segmentations but it requires dedicated methods to extract it from their weights [31].

4.2.3 Transfer learning on downstream tasks

In Tab. 6, we evaluate the quality of the features pretrained with DINO on different downstream tasks. We compare with features from the same architectures trained with supervision on ImageNet. We follow the protocol used in Touvron *et al.* [69] and finetune the features on each downstream task. We observe that for ViT architectures, self-supervised pretraining transfers better than features trained with supervision, which is consistent with observations made on convolutional networks [10, 33, 62]. Finally, self-supervised pretraining greatly improves results on ImageNet (+1-2%).

5. Ablation Study of DINO

In this section, we empirically study DINO applied to ViT. The model considered for this entire study is ViT-S. We also refer the reader to Appendix for additional studies.

5.1. Importance of the Different Components

We show the impact of adding different components from self-supervised learning on ViT trained with our framework.

	Random	Supervised	DINO
ViT-S/16	22.0	27.3	45.9
ViT-S/8	21.8	23.7	44.7

Figure 4: **Segmentations from supervised versus DINO.** We visualize masks obtained by thresholding the self-attention maps to keep 60% of the mass. On top, we show the resulting masks for a ViT-S/8 trained with supervision and DINO. We show the best head for both models. The table at the bottom compares the Jaccard similarity between the ground truth and these masks on the validation images of PASCAL VOC12 dataset.

Table 6: **Transfer learning by finetuning pretrained models on different datasets.** We report top-1 accuracy. Self-supervised pretraining with DINO transfers better than supervised pretraining.

	Cifar_{10}	Cifar_{100}	INat_{18}	INat_{19}	Flwrs	Cars	INet
ViT-S/16							
Sup. [69]	**99.0**	89.5	70.7	76.6	98.2	92.1	79.9
DINO	**99.0**	**90.5**	**72.0**	**78.2**	**98.5**	**93.0**	**81.5**
ViT-B/16							
Sup. [69]	99.0	90.8	**73.2**	77.7	98.4	92.1	81.8
DINO	**99.1**	**91.7**	72.6	**78.6**	**98.8**	**93.0**	**82.8**

In Table 7, we report different model variants as we add or remove components. First, we observe that in the absence of momentum, our framework does not work (row 2) and more advanced operations, SK for example, are required to avoid collapse (row 9). However, with momentum, using SK has little impact (row 3). In addtition, comparing rows 3 and 9 highlights the importance of the momentum encoder for performance. Second, in rows 4 and 5, we observe that multi-crop training and the cross-entropy loss in DINO are important components to obtain good features. We also observe that adding a predictor to the student network has little impact (row 6) while it is critical in BYOL to prevent collapse [16, 30]. For completeness, we propose in Appendix B an extended version of this ablation study.

Importance of the patch size. In Fig. 5, we compare the k-NN classification performance of ViT-S models trained

Table 7: **Important component for self-supervised ViT pretraining.** Models are trained for 300 epochs with ViT-S/16. We study the different components that matter for the k-NN and linear ("Lin.") evaluations. For the different variants, we highlight the differences from the default DINO setting. The best combination is the momentum encoder with the multicrop augmentation and the cross-entropy loss. We also report results with BYOL [30], MoCo-v2 [15] and SwAV [10].

	Method	Mom.	SK	MC	Loss	Pred.	k-NN	Lin.
1	DINO	✓	✗	✓	CE	✗	72.8	76.1
2		✗	✗	✓	CE	✗	0.1	0.1
3		✓	✓	✓	CE	✗	72.2	76.0
4		✓	✗	✗	CE	✗	67.9	72.5
5		✓	✗	✓	MSE	✗	52.6	62.4
6		✓	✗	✓	CE	✓	71.8	75.6
7	BYOL	✓	✗	✗	MSE	✓	66.6	71.4
8	MoCov2	✓	✗	✗	INCE	✗	62.0	71.6
9	SwAV	✗	✓	✓	CE	✗	64.7	71.8

SK: Sinkhorn-Knopp, MC: Multi-Crop, Pred.: Predictor
CE: Cross-Entropy, MSE: Mean Square Error, INCE: InfoNCE

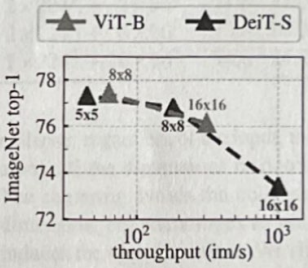


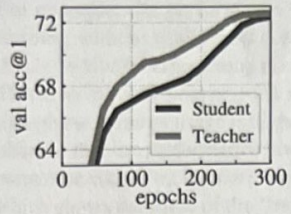
Figure 5: **Effect of Patch Size.** k-NN evaluation as a function of the throughputs for different input patch sizes with ViT-B and ViT-S. Models are trained for 300 epochs.

with different patch sizes, 16×16, 8×8 and 5×5. We also compare to ViT-B with 16×16 and 8×8 patches. All the models are trained for 300 epochs. We observe that the performance greatly improves as we decrease the size of the patch. It is interesting to see that performance can be greatly improved without adding additional parameters. However, the performance gain from using smaller patches comes at the expense of throughput: when using 5×5 patches, the throughput falls to 44 im/s, *vs* 180 im/s for 8×8 patches.

5.2. Impact of the choice of Teacher Network

In this ablation, we experiment with different teacher network to understand its role in DINO. We compare models trained for 300 epochs using the k-NN protocol.

Building different teachers from the student. In Fig. 6(right), we compare different strategies to build the teacher from previous instances of the student besides the



Teacher	Top-1
Student copy	0.1
Previous iter	0.1
Previous epoch	66.6
Momentum	72.8

Figure 6: Top-1 accuracy on ImageNet validation with k-NN classifier. **(left)** Comparison between the performance of the momentum teacher and the student during training. **(right)** Comparison between different types of teacher network. The momentum encoder leads to the best performance but is not the only viable option.

momentum teacher. First we consider using the student network from a previous epoch as a teacher. This strategy has been used in a memory bank [73] or as a form of clustering hard-distillation [8, 2, 14]. Second, we consider using the student network from the previous iteration, as well as a copy of the student for the teacher. In our setting, using a teacher based on a recent version of the student does not converge. This setting requires more normalizations to work. Interestingly, we observe that using a teacher from the previous epoch does not collapse, providing performance in the k-NN evaluation competitive with existing frameworks such as MoCo-v2 or BYOL. While using a momentum encoder clearly provides superior performance to this naive teacher, this finding suggests that there is a space to investigate alternatives for the teacher.

Analyzing the training dynamic. To further understand the reasons why a momentum teacher works well in our framework, we study its dynamic during the training of a ViT in the left panel of Fig. 6. A key observation is that this teacher constantly outperforms the student during the training, and we observe the same behavior when training with a ResNet-50 (Appendix D). This behavior has not been observed by other frameworks also using momentum [33, 30], nor when the teacher is built from the previous epoch. We propose to interpret the momentum teacher in DINO as a form of Polyak-Ruppert averaging [51, 59] with an exponentially decay. Polyak-Ruppert averaging is often used to simulate model ensembling to improve the performance of a network at the end of the training [38]. Our method can be interpreted as applying Polyak-Ruppert averaging during the training to constantly build a model ensembling that has superior performances. This model ensembling then guides the training of the student network [65].

5.3. Avoiding collapse

We study the complementarity role of centering and target sharpening to avoid collapse. There are two forms of collapse: regardless of the input, the model output is uniform along all the dimensions or dominated by one dimension. The centering avoids the collapse induced by a dominant dimension, but encourages an uniform output. Sharpening induces the opposite effect. We show this complementarity by decomposing the cross-entropy H into an entropy h and the Kullback-Leibler divergence ("KL") D_{KL}:

$$H(P_t, P_s) = h(P_t) + D_{KL}(P_t|P_s). \quad (5)$$

A KL equal to zero indicates a constant output, and hence a collapse. In Fig. 7, we plot the entropy and KL during training with and without centering and sharpening. If one operation is missing, the KL converges to zero, indicating a collapse. However, the entropy h converges to different values: 0 with no centering and $-\log(1/K)$ with no sharpening, indicating that both operations induce different form of collapse. Applying both operations balances these effects (see study of the sharpening parameter τ_t in Appendix D).

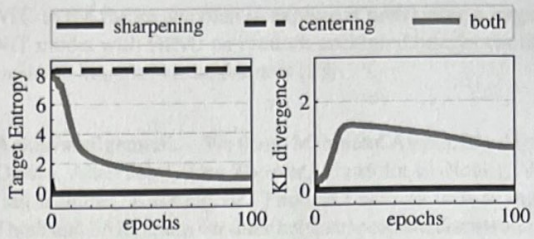


Figure 7: **Collapse study. (left)**: evolution of the teacher's target entropy along training epochs; **(right)**: evolution of KL divergence between teacher and student outputs.

Table 8: **Time and memory requirements.** We show total running time and peak memory per GPU ("mem.") when running ViT-S/16 DINO models on two 8-GPU machines. We report top-1 ImageNet val acc with linear evaluation for several variants of multi-crop, each having a different level of compute requirement.

	100 epochs		300 epochs		
multi-crop	top-1	time	top-1	time	mem.
2×224^2	67.8	15.3h	72.5	45.9h	9.3G
$2\times224^2 + 2\times96^2$	71.5	17.0h	74.5	51.0h	10.5G
$2\times224^2 + 6\times96^2$	73.8	20.3h	75.9	60.9h	12.9G
$2\times224^2 + 10\times96^2$	74.6	24.2h	76.1	72.6h	15.4G

5.4. Compute requirements

In Tab. 8, we detail the time and GPU memory requirements when running ViT-S/16 DINO models on two 8-GPU machines. We report results with several variants of multi-crop training, each having a different level of compute requirement. We observe in Tab. 8 that using multi-crop improves the accuracy / running-time tradeoff for DINO runs.

For example, the performance is 72.5% after 46 hours of training without multi-crop (i.e. 2×224^2) while DINO in $2\times224^2+10\times96^2$ crop setting reaches 74.6% in 24 hours only. This is an improvement of $+2\%$ while requiring $2\times$ less time, though the memory usage is higher ($15.4G$ versus $9.3G$). We observe that the performance boost brought with multi-crop cannot be caught up by more training in the 2×224^2 setting, which shows the value of the "local-to-global" augmentation. Finally, the gain from adding more views diminishes (+.2% form $6\times$ to $10\times$ 96^2 crops) for longer trainings.

Overall, training DINO with Vision Transformers achieves 76.1 top-1 accuracy using two 8-GPU servers for 3 days. This result outperforms state-of-the-art self-supervised systems based on convolutional networks of comparable sizes with a significant reduction of computational requirements [30, 10]. Our code is available to train self-supervised ViT on a limited number of GPUs.

5.5. Training with small batches

bs	128	256	512	1024
top-1	57.9	59.1	59.6	59.9

Table 9: **Effect of batch sizes.** Top-1 with k-NN for models trained for 100 epochs without multi-crop.

In Tab. 9, we study the impact of the batch size on the features obtained with DINO. We also study the impact of the smooth parameter m used in the centering update rule of Eq. 4 in Appendix D. We scale the learning rate linearly with the batch size [29]: $lr = 0.0005 * \text{batchsize}/256$. Tab. 9 confirms that we can train models to high performance with small batches. Results with the smaller batch sizes ($bs = 128$) are slightly below our default training setup of $bs = 1024$, and would certainly require to re-tune hyperparameters like the momentum rates for example. Note that the experiment with batch size of 128 runs on only 1 GPU. We have explored training a model with a batch size of 8, reaching 35.2% after 50 epochs, showing the potential for training large models that barely fit an image per GPU.

6. Conclusion

In this work, we have shown the potential of self-supervised pretraining a standard ViT model, achieving performance that are comparable with the best convnets specifically designed for this setting. We have also seen emerged two properties that can be leveraged in future applications: the quality of the features in k-NN classification has a potential for image retrieval where ViT are already showing promising results [22]. The presence of information about the scene layout in the features can also benefit weakly supervised image segmentation. However, the main result of this paper is that we have evidences that self-supervised learning could be the key to developing a BERT-like model based on ViT. In the future, we plan to explore if pretraining a large ViT model with DINO on random uncurated images could push the limits of visual features [28].

Acknowledgement. We thank Mahmoud Assran, Matthijs Douze, Allan Jabri, Jure Zbontar, Alaaeldin El-Nouby, Y-Lan Boureau, Kaiming He, Thomas Lucas as well as the Thoth and FAIR teams for their help, support and discussions around this project. Julien Mairal was funded by the ERC grant number 714381 (SOLARIS project) and by ANR 3IA MIAI@Grenoble Alpes (ANR-19-P3IA-0003).

References

[1] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018. 3

[2] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 2, 9

[3] Mahmoud Assran, Nicolas Ballas, Lluis Castrejon, and Michael Rabbat. Recovering petaflops in contrastive semi-supervised learning of visual representations. *preprint arXiv:2006.10803*, 2020. 14

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *preprint arXiv:1409.0473*, 2014. 5

[5] Maxim Berman, Hervé Jégou, Vedaldi Andrea, Iasonas Kokkinos, and Matthijs Douze. MultiGrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*, 2019. 6

[6] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. In *ICML*, 2017. 2

[7] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *SIGKDD*, 2006. 3

[8] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 2, 4, 9, 16

[9] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019. 2, 16

[10] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 1, 2, 3, 4, 5, 7, 8, 10, 14, 15, 16, 17, 18

[11] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. The best of both worlds: Combining recent advances in neural machine translation. *preprint arXiv:1804.09849*, 2018. 5

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *preprint arXiv:2002.05709*, 2020. 2, 3, 5, 16, 17