Supplementary Material for 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

A. Additional Results

k-NN classification. In Tab. 1, we evaluate the frozen representations given by ResNet-50 or ViT-small pre-trained with DINO with two evaluation protocols: linear or k-NN. For both evaluations, we extract representations from a pre-trained network without using any data augmentation. Then, we perform classification either with weighted k-NN or with a linear regression learned with `cyanure` library [18]. In Tab. 1 we see that ViT-S accuracies are better than accuracies obtained with RN50 both with a linear or a k-NN classifier. However, the performance gap when using the k-NN evaluation is much more significant than when considering linear evaluation. For example on ImageNet 1%, ViT-S outperforms ResNet-50 by a large margin of $+14.1\%$ with k-NN evaluation. This suggests that transformers architectures trained with DINO might offer more model flexibility that benefits the k-NN evaluation. K-NN classifiers have the great advantage of being fast and light to deploy, without requiring any domain adaptation. Overall, ViT trained with DINO provides features that combine particularly well with k-NN classifiers.

Table 1: **k-NN and linear evaluation for ViT-S/16 and ResNet-50 pre-trained with DINO.** We use ImageNet-1k [24] ("Inet"), Places205 [32], PASCAL VOC [13] and Oxford-102 flowers ("FLOWERS") [19]. ViT trained with DINO provides features that are particularly k-NN friendly.

	Logistic			k-NN		
	RN50	ViT-S	Δ	RN50	ViT-S	Δ
Inet 100%	72.1	75.7	3.6	67.5	74.5	7.0
Inet 10%	67.8	72.2	4.4	59.3	69.1	9.8
Inet 1%	55.1	64.5	9.4	47.2	61.3	14.1
Pl. 10%	53.4	52.1	-1.3	46.9	48.6	1.7
Pl. 1%	46.5	46.3	-0.2	39.2	41.3	2.1
VOC07	88.9	89.2	0.3	84.9	88.0	3.1
FLOWERS	95.6	96.4	0.8	87.9	89.1	1.2
Average Δ			2.4			**5.6**

Table 2: **ImageNet classification with different pretraining.** Top-1 accuracy on ImageNet for supervised ViT-B/16 models using different pretrainings or using an additional pretrained convnet to guide the training. The methods use different image resolution ("res.") and training procedure ("tr. proc."), i.e., data augmentation and optimization. "MPP" is *Masked Patch Prediction*.

Pretraining				
method	data	res.	tr. proc.	Top-1
Pretrain on additional data				
MMP	JFT-300M	384	[12]	79.9
Supervised	JFT-300M	384	[12]	84.2
Train with additional model				
Rand. init.	-	224	[28]	83.4
No additional data nor model				
Rand. init.	-	224	[12]	77.9
Rand. init.	-	224	[28]	81.8
Supervised	ImNet	224	[28]	81.9
DINO	ImNet	224	[28]	82.8

Self-supervised ImageNet pretraining of ViT. In this experiment, we study the impact of pretraining a supervised ViT model with our method. In Tab. 2, we compare the performance of supervised ViT models that are initialized with different pretraining or guided during training with an additional pretrained convnet. The first set of models are pretrained with and without supervision on the large curated dataset composed of 300M images. The second set of models are trained with hard knowledge distillation from a pretrained supervised RegNetY [22]. The last set of models do not use any additional data nor models, and are initialized either randomly or after a pretraining with DINO on ImageNet. Compare to random initialization, pretraining with DINO leads to a performance gain of +1%. This is not caused by a longer training since pretraining with supervision instead of DINO does not improve performance. Using self-supervised pretraining reduces the gap with models pretrained on extra

Table 3: **Low-shot learning on ImageNet with frozen ViT features.** We train a logistic regression on frozen features (FROZEN). Note that this FROZEN evaluation is performed *without any finetuning nor data augmentation*. We report top-1 accuracy. For reference, we show previously published results that uses finetuning and semi-supervised learning.

Method	Arch	Param.	Top 1 1%	Top 1 10%
Self-supervised pretraining with finetuning				
UDA [30]	RN50	23	–	68.1
SimCLRv2 [7]	RN50	23	57.9	68.4
BYOL [15]	RN50	23	53.2	68.8
SwAV [5]	RN50	23	53.9	70.2
SimCLRv2 [9]	RN50w4	375	63.0	74.4
BYOL [15]	RN200w2	250	71.2	77.7
Semi-supervised methods				
SimCLRv2+KD [7]	RN50	23	60.0	70.5
SwAV+CT [2]	RN50	23	–	70.8
FixMatch [26]	RN50	23	–	71.5
MPL [20]	RN50	23	–	73.9
SimCLRv2+KD [7]	RN152w3+SK	794	76.6	80.9
Frozen self-supervised features				
DINO -FROZEN	ViT-S/16	21	64.5	72.2

data or distilled from a convnet.

Low-shot learning on ImageNet. We evaluate the features obtained with DINO applied on ViT-S on low-shot learning. In Tab. 3, we report the validation accuracy of a logistic regression trained on frozen features (FROZEN) with 1% and 10% labels. The logistic regression is trained with the `cyanure` library [18]. When comparing models with a similar number of parameters and image/sec, we observe that our features are on par with state-of-the-art semi-supervised models. Interestingly, this performance is obtained by training a multi-class logistic regression on *frozen features, without data augmentation nor finetuning*.

B. Methodology Comparison

We compare the performance of different self-supervised frameworks, MoCo-v2 [8], SwAV [5] and BYOL [15] when using convnet or ViT. In Tab. 4, we see that when trained with ResNet-50 (convnet), DINO performs on par with SwAV and BYOL. However, DINO unravels its potential with ViT-S (ViT), outperforming MoCo-v2, SwAV and BYOL by large margins (+4.3% with linear and +6.2% with k-NN evaluations). In the rest of this section, we perform ablations to better understand the performance of DINO applied to ViT. In particular, we provide a detailed comparison with methods that either use a momentum encoder, namely MoCo-v2 and BYOL, and methods that use multi-crop, namely SwAV.

Table 4: **Methodology comparison for DEIT-small and ResNet-50.** We report ImageNet linear and k-NN evaluations validation accuracy after 300 epochs pre-training. All numbers are run by us and match or outperform published results.

	ResNet-50		ViT-small	
Method	Linear	k-NN	Linear	k-NN
MoCo-v2	71.1	62.9	71.6	62.0
BYOL	72.7	65.4	71.4	66.6
SwAV	74.1	65.4	71.8	64.7
DINO	**74.5**	**65.6**	**76.1**	**72.8**

Relation to MoCo-v2 and BYOL. In Tab. 5, we present the impact of ablating components that differ between DINO, MoCo-v2 and BYOL: the choice of loss, the predictor in the student head, the centering operation, the batch normalization in the projection heads, and finally, the multi-crop augmentation. The loss in DINO is a cross-entropy on sharpened softmax outputs (`CE`) while MoCo-v2 uses the InfoNCE contrastive loss (`INCE`) and BYOL a mean squared error on l2-normalized outputs (`MSE`). No sharpening is applied with the `MSE` criterion. Though, DINO surprisingly still works when changing the loss function to `MSE`, but this significantly alters the performance (see rows (1, 2) and (4, 9)). We also observe that adding a predictor has little impact (1, 3). However, in the case of BYOL, the predictor is critical to prevent collapse (7, 8) which is consistent with previous studies [9, 15]. Interestingly, we observe that the teacher output centering avoids collapse without predictor nor batch normalizations in BYOL (7, 9), though with a significant performance drop which can likely be explained by the fact that our centering operator is designed to work in combination with sharpening. Finally, we observe that multi-crop works particularly well with DINO and MoCo-v2, removing it hurts performance by $2-4\%$ (1 versus 4 and, 5 versus 6). Adding multi-crop to BYOL does not work out-of-the-box (7, 10) as detailed in Appendix E and further adaptation may be required.

Relation to SwAV. In Tab. 6, we evaluate the differences between DINO and SwAV: the presence of the momentum encoder and the operation on top of the teacher output. In absence of the momentum, a copy of the student with a stop-gradient is used. We consider three operations on the teacher output: `Centering`, `Sinkhorn-Knopp` or a `Softmax` along the batch axis. The `Softmax` is similar to a single Sinkhorn-Knopp iteration as detailed in the next paragraph. First, these ablations show that using a momentum encoder significantly improves the performance for ViT (3 versus 6, and 2 versus 5). Second, the momentum encoder also avoids collapse when using only centering (row 1). In the absence

Table 5: **Relation to MoCo-v2 and BYOL.** We ablate the components that differ between DINO, MoCo-v2 and BYOL: the loss function (cross-entropy, `CE`, versus InfoNCE, `INCE`, versus mean-square error, `MSE`), the multi-crop training, the centering operator, the batch normalization in the projection heads and the student predictor. Models are run for 300 epochs with ViT-S/16. We report top-1 accuracy on ImageNet linear evaluation.

	Method	Loss	multi-crop	Center.	BN	Pred.	Top-1
1	DINO	`CE`	✓	✓			76.1
2	–	`MSE`	✓	✓			62.4
3	–	`CE`	✓	✓		✓	75.6
4	–	`CE`		✓			72.5
5	MoCov2	`INCE`			✓		71.4
6		`INCE`	✓		✓		73.4
7	BYOL	`MSE`			✓	✓	71.4
8	–	`MSE`			✓		0.1
9	–	`MSE`		✓			52.6
10	–	`MSE`	✓		✓	✓	64.8

Table 6: **Relation to SwAV.** We vary the operation on the teacher output between centering, a softmax applied over the batch dimension and the Sinkhorn-Knopp algorithm. We also ablate the Momentum encoder by replacing it with a hard copy of the student with a stop-gradient as in SwAV. Models are run for 300 epochs with ViT-S/16. We report top-1 accuracy on ImageNet linear evaluation.

	Method	Momentum	Operation	Top-1
1	DINO	✓	`Centering`	76.1
2	–	✓	`Softmax(batch)`	75.8
3	–	✓	`Sinkhorn-Knopp`	76.0
4	–		`Centering`	0.1
5	–		`Softmax(batch)`	72.2
6	SwAV		`Sinkhorn-Knopp`	71.8

of momentum, centering the outputs does not work (4) and more advanced operations are required (5, 6). Overall, these ablations highlight the importance of the momentum encoder, not only for performance but also to stabilize training, removing the need for normalization beyond centering.

Details on the `Softmax(batch)` variant. The iterative Sinkhorn-Knopp algorithm [10] used in SwAV [5] is implemented simply with the following PyTorch style code.

```
# x is n-by-K
# tau is Sinkhorn regularization param
x = exp(x / tau)
for _ in range(num_iters): # 1 iter of Sinkhorn
  # total weight per dimension (or cluster)
  c = sum(x, dim=0, keepdim=True)
  x /= c
  # total weight per sample
  n = sum(x, dim=1, keepdim=True)
  # x sums to 1 for each sample (assignment)
  x /= n
```

When performing a single Sinkhorn iteration (`num_iters=1`) the implementation can be highly simplified into only two lines of code, which is our `softmax(batch)` variant:

```
x = softmax(x / tau, dim=0)
x /= sum(x, dim=1, keepdim=True)
```

We have seen in Tab. 6 that this highly simplified variant of SwAV works competitively with SwAV. Intuitively, the `softmax` operation on the batch axis allows to select for each dimension (or "cluster") its best matches in the batch.

Validating our implementation. We observe in Tab. 4 that our reproduction of BYOL, MoCo-v2, SwAV matches or outperforms the corresponding published numbers with ResNet-50. Indeed, we obtain 72.7% for BYOL while [15] report 72.5% in this 300-epochs setting. We obtain 71.1% for MoCo after 300 epochs of training while [8] report 71.1% after 800 epochs of training. Our improvement compared to the implementation of [8] can be explained by the use of a larger projection head (3-layer, use of batch-normalizations and projection dimension of 256).

Concurrent work CsMI. The concurrent work CsMI [31] also exhibits strong performance with simple k-NN classifiers on ImageNet, even with convnets. As DINO, CsMI combines a momentum network and multi-crop training, which we have seen are both crucial for good k-NN performance in our experiments with ViTs. We believe studying this work would help us identifying more precisely the components important for good k-NN performance and leave this investigation for future work.

C. Projection Head

Similarly to other self-supervised frameworks, using a projection head [6] improves greatly the accuracy of our method. The projection head starts with a n-layer multi-layer perceptron (MLP). The hidden layers are 2048d and are with gaussian error linear units (GELU) activations. The last layer of the MLP is without GELU. Then we apply a ℓ_2 normalization and a weight normalized fully connected layer [9, 25] with K dimensions. This design is inspired from the projection head with a "prototype layer" used in SwAV [5]. We do not apply batch normalizations.

BN-free system. 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. In this table we evaluate the impact

ViT-S, 100 epochs	heads w/o BN	heads w/ BN
k-NN top-1	69.7	68.6

of adding BN in the heads. We observe that adding BN in the projection heads has little impact, showing that BN is not important in our framework. *Overall, when applying DINO to ViT, we do not use any BN anywhere, making the system entirely BN-free.* This is a great advantage of DINO + ViT to work at state-of-the-art performance without requiring any BN. Indeed, training with BN typically slows down trainings considerably, especially when these BN modules need to be synchronized across processes [16, 5, 4, 15].

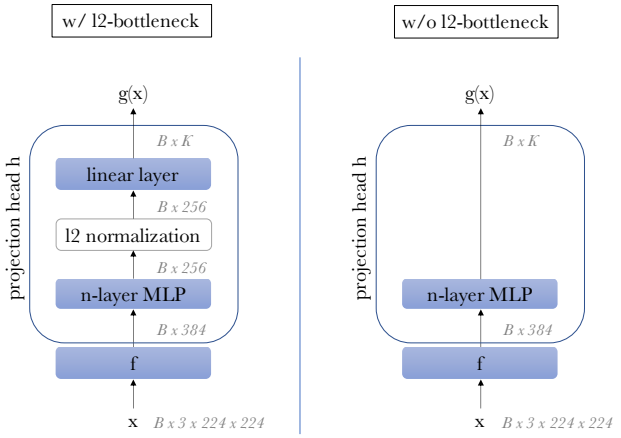


Figure 1: **Projection head design w/ or w/o l2-norm bottleneck.**

L2-normalization bottleneck in projection head. We illustrate the design of the projection head with or without l2-normalization bottleneck in Fig. 1. We evaluate the accuracy

# proj. head linear layers	1	2	3	4
w/ l2-norm bottleneck	–	62.2	68.0	69.3
w/o l2-norm bottleneck	61.6	62.9	0.1	0.1

of DINO models trained with or without l2-normalization bottleneck and we vary the number of linear layers in the projection head. With l2 bottleneck, the total number of linear layers is $n + 1$ (n from the MLP and 1 from the weight normalized layer) while without bottleneck the total number of linear layers is n in the head. In this table, we report ImageNet top-1 k-NN evaluation accuracy after 100 epochs pre-training with ViT-S/16. The output dimensionality K is set to 4096 in this experiment. We observe that DINO training fails without the l2-normalization bottleneck when increasing the depth of the projection head. L2-normalization bottleneck stabilizes the training of DINO with deep projection head. We observe that increasing the depth of the projection head improves accuracy. Our default is to use a total of 4 linear layers: 3 are in the MLP and one is after the l2 bottleneck.

Output dimension. In this table, we evaluate the effect of varying the output dimensionality K. We observe that a

K	1024	4096	16384	65536	262144
k-NN top-1	67.8	69.3	69.2	69.7	69.1

large output dimensionality improves the performance. We note that the use of l2-normalization bottleneck permits to use a large output dimension with a moderate increase in the total number of parameters. Our default is to use K equals to 65536 and $d = 256$ for the bottleneck.

GELU activations. By default, the activations used in ViT are gaussian error linear units (GELU). Therefore, for consistency within the architecture, we choose to use GELU also in the projection head. We evaluate the effect of using ReLU instead of GELU in this table and observe that changing the activation unit to ReLU has relatively little impact.

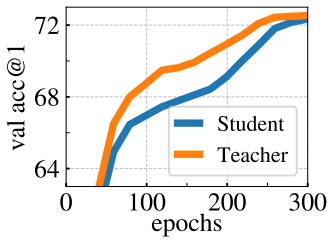
ViT-S, 100 epochs	heads w/ GELU	heads w/ ReLU
k-NN top-1	69.7	68.9

D. Additional Ablations

We have detailed in the main paper that the combination of centering and sharpening is important to avoid collapse in DINO. We ablate the hyperparameters for these two operations in the following. We also study the impact of training length and some design choices for the ViT networks.

Building different teachers from the student. In Fig. 2(right), we compare different strategies to build the teacher from previous instances of the student besides the momentum teacher. First we consider using the student network from a previous epoch as a teacher. This strategy has been used in the memory bank of Wu *et al*. [29] and as a form of hard-distillation in Caron *et al*. [3] and Asano *et al*. [1]. 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. 2. A key observation is that this teacher constantly outperforms the student during the



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

Figure 2: 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.

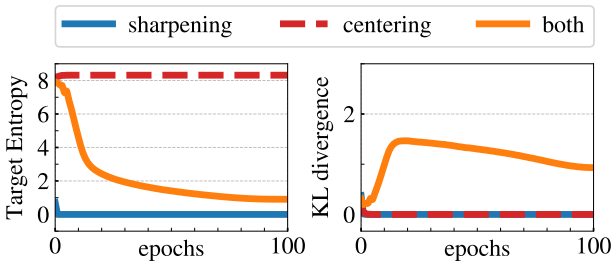


Figure 3: **Collapse study. (left)**: evolution of the teacher's target entropy along training epochs; **(right)**: evolution of KL divergence between teacher and student outputs.

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 [16, 15], 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 [21, 23] 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 [17]. 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 [27].

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}:

$$H(P_t, P_s) = h(P_t) + D_{KL}(P_t|P_s). \quad (1)$$

Table 7: **Time and memory requirements.** We show total running time and peak memory per GPU ("mem.") when running DeiT-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

A KL equal to zero indicates a constant output, and hence a collapse. In Fig. 3, 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).

Compute requirements In Tab. 7, we detail the time and GPU memory requirements when running DeiT-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. 7 that using multi-crop improves the accuracy / running-time trade-off 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× 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× to 10× 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 [15, 5]. Our code is available to train self-supervised ViT on a limited number of GPUs.

Training with small batches In Tab. 8, we study the impact of the batch size on the features obtained with DINO. We also study the impact of the smooth parame-

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

Table 8: **Effect of batch sizes.** Top-1 with k-NN for models trained for 100 epochs without multi-crop.

ter m used in the centering update rule in Appendix D. We scale the learning rate linearly with the batch size [14]: $lr = 0.0005 * \text{batchsize}/256$. Tab. 8 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.

Online centering. We study the impact of the smoothing parameters in the update rule for the center c used in the output of the teacher network. The convergence is robust

m	0	0.9	0.99	0.999
k-NN top-1	69.1	69.7	69.4	0.1

to a wide range of smoothing, and the model only collapses when the update is too slow, i.e., $m = 0.999$.

Sharpening. We enforce sharp targets by tuning the teacher softmax temperature parameter τ_t. In this table, we observe that a temperature lower than 0.06 is required to avoid collapse. When the temperature is higher than 0.06,

τ_t	0	0.02	0.04	0.06	0.08	$0.04 \rightarrow 0.07$
k-NN top-1	43.9	66.7	69.6	68.7	0.1	69.7

the training loss consistently converges to $ln(K)$. However, we have observed that using higher temperature than 0.06 does not collapse if we start the training from a smaller value and increase it during the first epochs. In practice, we use a linear warm-up for τ_t from 0.04 to 0.07 during the first 30 epochs of training. Finally, note that $\tau \rightarrow 0$ (extreme sharpening) correspond to the `argmax` operation and leads to one-hot hard distributions.

Longer training. We observe in this table that longer training improves the performance of DINO applied to ViT-Small. This observation is consistent with self-supervised results

DINO ViT-S	100-ep	300-ep	800-ep
k-NN top-1	70.9	72.8	74.5

obtained with convolutional architectures [6]. We note that in our experiments with BYOL on ViT-S, training longer than 300 epochs has been leading to worse performance compare our 300 epochs run. For this reason we report BYOL for 300 epochs in the main paper while SwAV, MoCo-v2 and DINO are trained for 800 epochs.

Self-attention maps from supervised versus self-supervised learning. We evaluate the masks obtained by thresholding the self-attention maps to keep 80% of the mass. We compare the Jaccard similarity between the

ViT-S/16 weights	
Random weights	22.0
Supervised	27.3
DINO	45.9
DINO w/o multicrop	45.1
MoCo-v2	46.3
BYOL	47.8
SwAV	46.8

ground truth and these masks on the validation images of PASCAL VOC12 dataset for different ViT-S trained with different frameworks. The properties that self-attention maps from ViT explicitly contain the scene layout and, in particular, object boundaries is observed across different self-supervised methods.

Impact of the number of heads in ViT-S. We study the impact of the number of heads in ViT-S on the accuracy and throughput (images processed per second at inference time on a singe V100 GPU). We find that increasing the number

# heads	dim	dim/head	# params	im/sec	k-NN
6	384	64	21	1007	72.8
8	384	48	21	971	73.1
12	384	32	21	927	73.7
16	384	24	21	860	73.8

of heads improves the performance, at the cost of a slighlty worse throughput. In our paper, all experiments are run with the default model presented in [28], i.e. with 6 heads only.

E. Multi-crop

In this Appendix, we study a core component of DINO: multi-crop training [5].

Range of scales in multi-crop. For generating the different views, we use the `RandomResizedCrop` method from `torchvision.transforms` module in PyTorch. We sample two global views with scale range $(s, 1)$ before

$(0.05, s)$, $(s, 1)$, s:	0.08	0.16	0.24	0.32	0.48
k-NN top-1	65.6	68.0	69.7	69.8	69.5

resizing them to 224^2 and 6 local views with scale sampled

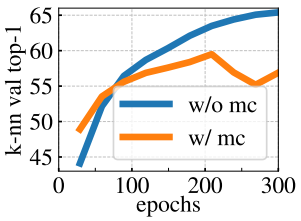
in the range $(0.05, s)$ resized to 96^2 pixels. Note that we arbitrarily choose to have non-overlapping scaling range for the global and local views following the original design of SwAV. However, the ranges could definitely be overlapping and experimenting with finer hyperparameters search could lead to a more optimal setting. In this table, we vary the parameter s that controls the range of scales used in multi-crop and find the optimum to be around 0.3 in our experiments. We note that this is higher than the parameter used in SwAV which is of 0.14.

Multi-crop in different self-supervised frameworks. We compare different recent self-supervised learning frameworks, namely MoCo-v2 [8], BYOL [15] and SwAV [5] with ViT-S/16 architecture. For fair comparisons, all models are

crops	2×224^2		$2 \times 224^2 + 6 \times 96^2$	
eval	k-NN	linear	k-NN	linear
BYOL	66.6	71.4	59.8	64.8
SwAV	60.5	68.5	64.7	71.8
MoCo-v2	62.0	71.6	65.4	73.4
DINO	**67.9**	**72.5**	**72.7**	**75.9**

pretrained either with two 224^2 crops or with multi-crop [5] training, i.e. two 224^2 crops and six 96^2 crops for each image. We report k-NN and linear probing evaluations after 300 epochs of training. Multi-crop does not benefit all frameworks equally, which has been ignored in benchmarks considering only the two crops setting [9]. The effectiveness of multi-crop depends on the considered framework, which positions multi-crop as a core component of a model and not a simple "add-ons" that will boost any framework the same way. Without multi-crop, DINO has better accuracy than other frameworks, though by a moderate margin (1%). Remarkably, DINO benefits the most from multi-crop training ($+3.4\%$ in linear eval). Interestingly, we also observe that the ranking of the frameworks depends on the evaluation protocol considered.

Training BYOL with multi-crop. When applying multi-crop to BYOL with ViT-S, we observe the transfer performance is higher than the baseline without multi-crop for the first training epochs. However, the transfer performance



growth rate is slowing down and declines after a certain amount of training. We have performed learning rate, weight decay, multi-crop parameters sweeps for this setting and systematically observe the same pattern. More precisely, we experiment with $\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}\}$ for learning rate base values, with $\{0.02, 0.05, 0.1\}$ for weight decay and with different number of small crops: $\{2, 4, 6\}$. All our runs are performed with synchronized batch normalizations in the heads. When using a low learning rate, we did not observe the performance break point, i.e. the transfer performance was improving continually during training, but the overall accuracy was low. We have tried a run with multi-crop training on ResNet-50 where we also observe the same behavior. Since integrating multi-crop training to BYOL is not the focus of this study we did not push that direction further. However, we believe this is worth investigating why multi-crop does not combine well with BYOL in our experiments and leave this for future work.

F. Evaluation Protocols

F.1 k-NN classification

Following the setting of Wu *et al*. [29], we evaluate the quality of features with a simple weighted k Nearest Neighbor classifier. We freeze the pretrained model to compute and store the features of the training data of the downstream task. To classify a test image x, we compute its representation and compare it against all stored training features T. The representation of an image is given by the output `[CLS]` token: it has dimensionality $d = 384$ for ViT-S and $d = 768$ for ViT-B. The top k NN (denoted $\mathcal{N}_k$) are used to make a prediction via weighted voting. Specifically, the class c gets a total weight of $\sum_{i \in \mathcal{N}_k} \alpha_i \mathbf{1}_{c_i=c}$, where α_i is a contribution weight. We use $\alpha_i = \exp(T_i x / \tau)$ with τ equals to 0.07 as in [29] which we do not tune. We evaluate different values for k and find that $k = 20$ is consistently leading to the best accuracy across our runs. This evaluation protocol does not require hyperparameter tuning, nor data augmentation and can be run with only one pass over the downstream dataset.

F.2 Linear classification

Following common practice in self-supervised learning, we evaluate the representation quality with a linear classifier. The projection head is removed, and we train a supervised linear classifier on top of frozen features. This linear classifier is trained with SGD and a batch size of 1024 during 100 epochs on ImageNet. We do not apply weight decay. For each model, we sweep the learning rate value. During training, we apply only random resizes crops (with default parameters from PyTorch `RandomResizedCrop`) and horizontal flips as data augmentation. We report central-crop top-1 accuracy. When evaluating convnets, the common practice is to perform global average pooling on the final

feature map before the linear classifier. In the following, we describe how we adapt this design when evaluating ViTs.

ViT-S representations for linear eval. Following the *feature-based* evaluations in BERT [11], we concatenate the `[CLS]` tokens from the l last layers. We experiment

concatenate l last layers	1	2	4	6
representation dim	384	768	1536	2304
ViT-S/16 linear eval	76.1	76.6	77.0	77.0

with the concatenation of a different number l of layers and similarly to [11] we find $l = 4$ to be optimal.

ViT-B representations for linear eval. With ViT-B we did not find that concatenating the representations from the last l layers to provide any performance gain, and consider the final layer only ($l = 1$). In this setting, we adapt the

pooling strategy	`[CLS]` tok. only	concatenate `[CLS]` tok. and avgpooled patch tok.
representation dim	768	1536
ViT-B/16 linear eval	78.0	78.2

pipeline used in convnets with global average pooling on the output patch tokens. We concatenate these pooled features to the final `[CLS]` output token.

G. Self-Attention Visualizations

We provide more self-attention visualizations in Fig. 4. The images are randomly selected from COCO validation set, and are not used during training of DINO.

H. Class Representation

As a final visualization, we propose to look at the distribution of ImageNet concepts in the feature space from DINO. We represent each ImageNet class with the average feature vector for its validation images. We reduce the dimension of these features to 30 with PCA, and run t-SNE with a perplexity of 20, a learning rate of 200 for 5000 iterations. We present the resulting class embeddings in Fig. 5. Our model recovers structures between classes: similar animal species are grouped together, forming coherent clusters of birds (top) or dogs, and especially terriers (far right).

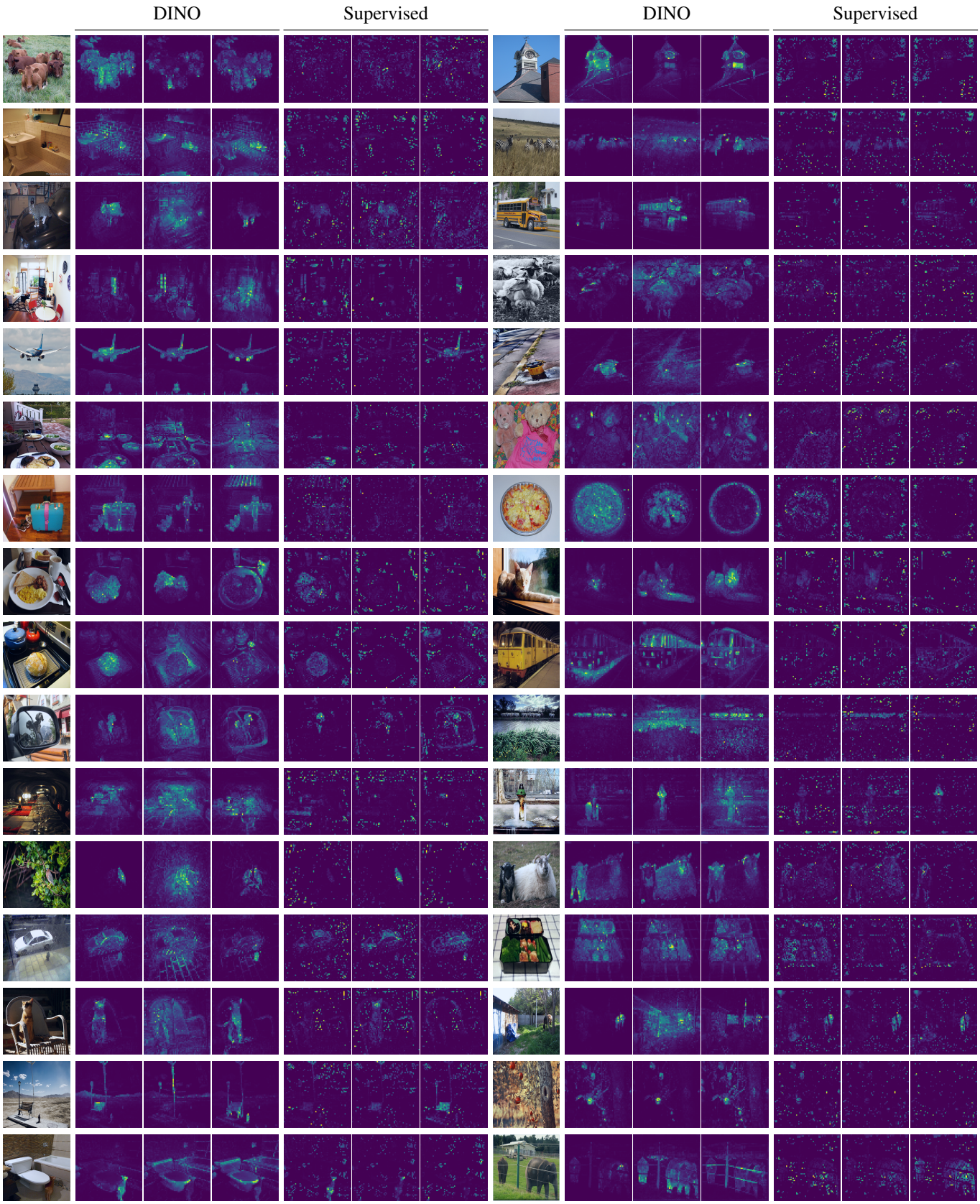


Figure 4: **Self-attention heads from the last layer.** We look at the attention map when using the `[CLS]` token as a query for the different heads in the last layer. Note that the `[CLS]` token is not attached to any label or supervision.

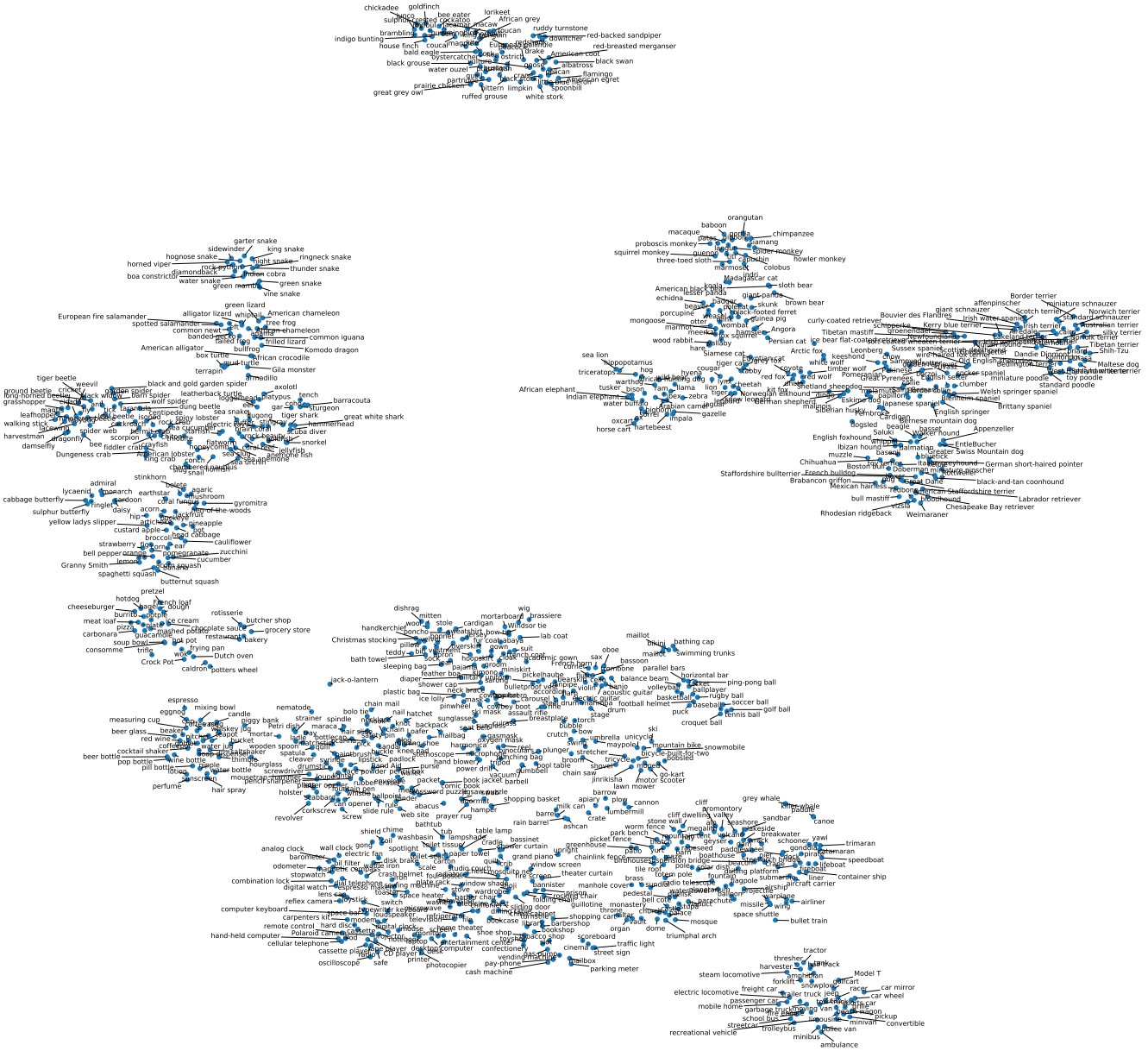


Figure 5: t-SNE visualization of ImageNet classes as represented using DINO. For each class, we obtain the embedding by taking the average feature for all images of that class in the validation set.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 4

[2] Mahmoud Assran, Nicolas Ballas, Lluis Castrejon, and Michael Rabbat. Recovering petaflops in contrastive semi-supervised learning of visual representations. *preprint arXiv:2006.10803*, 2020. 2

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

[4] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019. 4

[5] 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. 2, 3, 4, 5, 6, 7

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *preprint arXiv:2002.05709*, 2020. 3, 6

[7] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020. 2

[8] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *preprint arXiv:2003.04297*, 2020. 2, 3, 7

[9] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *preprint arXiv:2011.10566*, 2020. 2, 3, 7

[10] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013. 3

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *preprint arXiv:1810.04805*, 2018. 8

[12] 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. *preprint arXiv:2010.11929*, 2020. 1

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 1

[14] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *preprint arXiv:1706.02677*, 2017. 6

[15] 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, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020. 2, 3, 4, 5, 7

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 4, 5

[17] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *preprint arXiv:1412.2007*, 2014. 5

[18] Julien Mairal. Cyanure: An open-source toolbox for empirical risk minimization for python, c++, and soon more. *preprint arXiv:1912.08165*, 2019. 1, 2

[19] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 2008. 1

[20] Hieu Pham, Qizhe Xie, Zihang Dai, and Quoc V Le. Meta pseudo labels. *preprint arXiv:2003.10580*, 2020. 2

[21] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 5

[22] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020. 1

[23] David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, 1988. 5

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 1

[25] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *NeurIPS*, 2016. 3

[26] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020. 2

[27] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *preprint arXiv:1703.01780*, 2017. 5

[28] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *preprint arXiv:2012.12877*, 2020. 1, 6

[29] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018. 4, 7

[30] Qizhe Xie, Zihang Dai Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training. *preprint arXiv:1904.12848*, 2020. 2

[31] Haohang Xu, Xiaopeng Zhang, Hao Li, Lingxi Xie, Hongkai Xiong, and Qi Tian. Seed the views: Hierarchical semantic alignment for contrastive representation learning. *arXiv preprint arXiv:2012.02733*, 2021. 3

[32] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NeurIPS*, 2014. 1