Divide and Contrast: Self-supervised Learning from Uncurated Data

Yonglong Tian *
MIT

Olivier J. Hénaff
DeepMind

Aaron van den Oord
DeepMind

Abstract

Self-supervised learning holds promise in leveraging large amounts of unlabeled data, however much of its progress has thus far been limited to highly curated pre-training data such as ImageNet. We explore the effects of contrastive learning from larger, less-curated image datasets such as YFCC, and find there is indeed a large difference in the resulting representation quality. We hypothesize that this curation gap is due to a shift in the distribution of image classes—which is more diverse and heavy-tailed—resulting in less relevant negative samples to learn from. We test this hypothesis with a new approach, Divide and Contrast (DnC), which alternates between contrastive learning and clustering-based hard negative mining. When pretrained on less curated datasets, DnC greatly improves the performance of self-supervised learning on downstream tasks, while remaining competitive with the current state-of-the-art on curated datasets.

1. Introduction

Recent developments in self-supervised learning have shown that it is possible to learn high-level representations of object categories from unlabeled images [40, 44, 92, 12, 101], phonetic information from speech [73, 83] and language understanding from raw text [21, 105]. The most studied benchmark in self-supervised learning is ImageNet [20], where representations learned from unlabeled images can surpass supervised representations, both in terms of their data-efficiency and transfer-learning performance [13, 35].

One of the caveats with self-supervised learning on ImageNet is that it is not completely "self-supervised". The training set of ImageNet, on which the representations are learned, is heavily curated and required extensive human effort to create [20]. In particular, ImageNet contains many fine-grained classes (such as subtly different dog breeds), each one containing roughly the same number of images. While this *consistency* may facilitate the learning of high-level visual representations, limiting self-supervised learn-

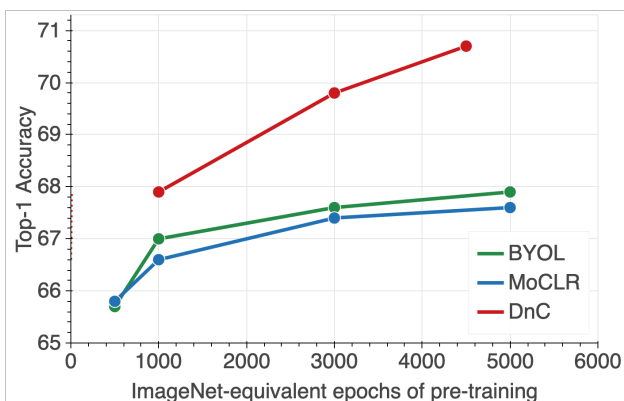*Work done while interning at DeepMind.

Figure 1. Linear evaluation on ImageNet of representations learned from a large-scale uncurated dataset using ResNet-50. Divide and Contrast (DnC) is better able to handle the diverse and long-tailed distribution of images and improves more with longer training. X-axis represents total computation, in ImageNet-equivalent epochs.

ing to such curated datasets risks biasing their development towards methods which require this consistency, limiting their applicability to more diverse downstream tasks and larger datasets for pre-training.

In this paper we assess how well recent self-supervised learning methods perform on downstream tasks (including ImageNet) when they are pre-trained on significantly less curated datasets, such as YFCC100M [91]. We observe a notable drop in performance of over 9% Top-1 accuracy (from 74.3% to 65.3%) for a ResNet50 model trained with the current state-of-the-art in self-supervised learning.

We hypothesize that this *curation gap* is due to the heavy-tailed nature of images collected in the wild, which present much more diverse content, breaking the *global consistency* exploited in previous datasets. We test this hypothesis with a new method, Divide and Contast (DnC), which attempts to recover *local consistency* in subsets of the larger, uncurated dataset, such that self-supervised learning methods can learn high-level features that are specific to each subset. We find that such semantically coherent subsets can be straightforwardly obtained by clustering the representations of standard self-supervised models.

Divide and Contrast (DnC) proceeds by training individ-

ual "expert" models on each subset and distilling them into a single model. As a result, DnC can be used in combination with any self-supervised learning technique, and requires the same amount of computation, as each expert is trained for significantly less time. Finally, this computation is trivially parallelized, allowing it to be scaled to massive datasets.

The remainder of this paper is structured as follows. We first review related work in self-supervised learning. We then present a new stronger baseline (MoCLR) which improves over current contrastive methods, matching the performance of the current state-of-the-art (BYOL [35]). Next we present the main method, Divide and Contrast, and how this model can be used together with any SSL method. In the experiments we evaluate the different hypotheses that support DnC, and compare its ability to learn from uncurated dataset with existing methods.

2. Related Work

Recent self-supervised representation learning generally includes three types of methods: *generative models* that directly model the data distribution, *pretext tasks* that are manually designed according to the data, and *contrastive learning* that contrasts positive pairs with negative pairs.

Generative models. While the primary goal of generative models such as GAN [30, 14] or VAE [51] is to model the data distribution (e.g., sample new data or estimate likelihood), the encoder network can also extract good representations [80]. Recent state of the art generative models for representation learning include BiGAN [24] and BigBiGAN [25], which learn a bidirectional mapping between the latent codes and the images, and iGPT [11] which trains an autoregressive model on raw pixels.

Pretext tasks. Good representations may also be learned by solving various pretext tasks. Examples include denoising [97], relative patch prediction [22], image inpainting [77], noise prediction [5], colorization [112, 113, 98], Jigsaw [72], exemplar modeling [26], motion segmentation [76], image transformation prediction [29, 111], tracking [100], or even the combination of multiple tasks [23]. Another line of methods generates pseudo labels by clustering features [8, 9, 47, 110, 1]. Most recently, SeLa [106] jointly clusters images and balances the clusters. SwAV [10] learns representatons by having different views of the same image assigned to the same cluster. Another work [46] directly optimizes the transferability of representation by integrating clustering with meta-learning.

Contrastive learning. Contrastive learning is a widely-used generic method. The loss function for contrastive learning has evolved from early margin-based binary classification [37], to triplet loss [84], and to recent k-pair loss [86, 73]. The core idea lying at the heart of the recent series of self-supervised contrastive learning methods [101, 73, 44, 92, 118, 3, 40, 68, 12, 15, 94, 13, 58, 7] is to maximize the agreement between two "views" of the same image while repulsing "views" from different images. Such views can be created by color decomposition [92], patch cropping [73, 44, 3], data augmentation [12, 13, 87], or image segmentation [43, 96, 114]. Indeed, contrastive learning is very general such that it can be easily adapted to different data types. Examples include different frames of video [73, 117, 85, 38, 31, 39], point clouds [104], multiple sensory data [69, 18, 78], text and its context [67, 105, 62, 53], or video and language [88, 66, 59]. A set of other work [2, 94, 115, 103, 95, 79, 99] focuses on providing empirical and theoretical understanding of contrastive learning. Recently a non-contrastive method BYOL [35] applies a momentum-encoder to one view and predicts its output from the other, inspired by bootstrapping RL [36]. Finally, contrastive learning has also been applied to supervised image classification [49], image translation [74], knowledge distillation [93, 81], and adversarial learning [50].

This paper is also related to knowledge distillation [45]. In [45], several expert models were also trained in parallel on a large scale dataset, and then distilled into a single model. While labels are assumed available in [45] to partition the dataset and distill into a single model, we are dealing with self-supervised learning without supervision. Our distillation procedure is also inspired by FitNet [82].

Lastly, while self-supervised representation learning on uncurated datasets is largely unexplored, there are a few prior attempts [9, 34]. In [9], clustering is applied to generate training targets, and in order to capture the long-tailed distribution of images in the uncurated YFCC100m [91], a hierachical formulation is proposed. The work of [34] benchmarked pretext-based self-supervised methods in a large scale setting, e.g., jigSaw, colorization and rotation prediction, and found that these pretext tasks are not 'hard' enough to take full advantage of large scale data. Concurrent work SEER [32] directly scales up SwAV with larger models and datasets.

3. Divide and Contrast

Though Divide and Contrast can be used in combination with any self-supervised learning technique, in this paper we will combine it with recent state of the art techniques (BYOL, SimCLR, MoCo), such that the model can be compared to a strong baseline and make the experiments relevant with respect to recent developments in the literature. We will start by introducing our baseline, MoCLR, which is a simple hybrid based on BYOL [35], SimCLR [12] and MoCo [40], and as a contrastive method outperforms SimCLR v2 [13], achieving similar performance to BYOL (by using a momentum encoder similar to BYOL and MoCo). Even though DnC can be coupled with BYOL, empirically we have found it to work better with methods that use a
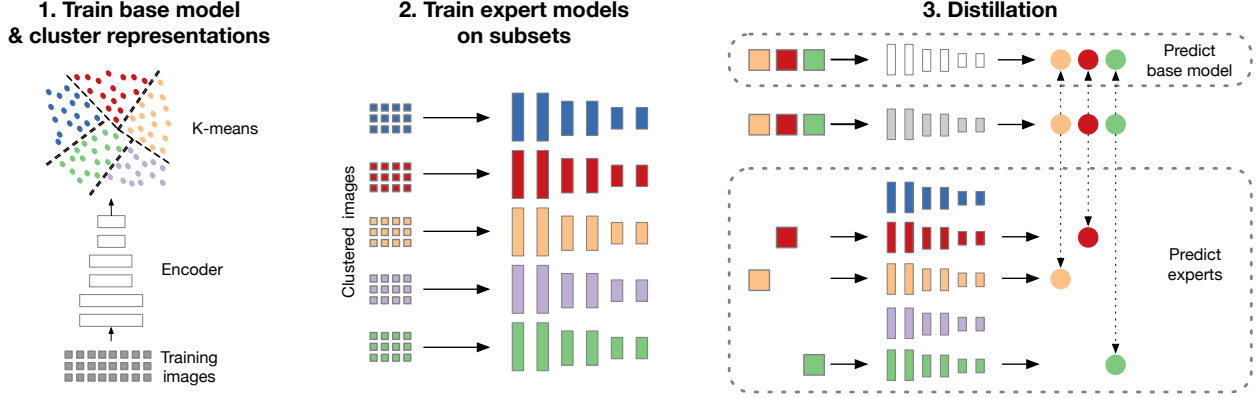
Figure 2. Overview of Divide and Contrast (DnC). DnC can be used in conjunction with any self-supervised learning method (we use MoCLR, an improvement to SimCLR). In the first step a self-supervised learning method is trained on the whole dataset, which we call the base model. The image representations of the base model are then clustered with k-means into 5, 10 or more groups. In the second step, the clustered dataset is then used to train an expert model on each of the image clusters. In the third step the experts and base model are distilled into a single model by predicting their representations. By splitting the dataset into semantically-similar subsets, contrastive methods need to pay more attention to the differences between the images in those clusters and learn more specific representations.

contrastive loss.

3.1. An Improved Contrastive Baseline: MoCLR

MoCLR roughly uses a similar setup to SimCLR and BYOL, we will briefly describe the main components of this setup and highlight the differences.

Augmenting two views. Given an image x and two distributions of image augmentations $\mathcal{T}$ and $\mathcal{T}'$, two *views* are created $v \triangleq t(x)$ and $v' \triangleq t'(x)$ by respectively applying random image augmentations $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}'$ from these distributions. The augmentations $\mathcal{T}$ and $\mathcal{T}'$ here are exactly the same as in BYOL [35].

Architecture. The first augmented view v is fed into an online encoder $f(\cdot)$, followed up with an MLP projection head $g(\cdot)$ to produce a projection z. Similarly, an exponential moving average of the online encoder and projection head, also known as momentum encoder [40] or mean teacher [90], is applied on the second view v' to generate z'. The MLP head consists of two layers with a hidden dimension of 4096 and output size of 256, similar to BYOL [35].

Loss function. Given a batch $\mathcal{B}$, we follow the InfoNCE loss [73] with a cosine similarity function $s(\cdot, \cdot)$ and a scalar temperature value τ:

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \log \frac{e^{s(z_i, z'_i)/\tau}}{e^{s(z_i, z'_i)/\tau} + \sum_{j \in \mathcal{B}/i} e^{s(z_i, z'_j)/\tau}} \quad (1)$$

We symmetrize the loss $\mathcal{L}_{\text{NCE}}$ by separately feeding v' to the online network and v to the momentum encoder, resulting in $\widetilde{\mathcal{L}}_{\text{NCE}}$. The final loss is $\mathcal{L} = \mathcal{L}_{\text{NCE}} + \widetilde{\mathcal{L}}_{\text{NCE}}$.

The difference between MoCLR and other standard methods are as follows. Compared with SimCLR [12], we use a momentum encoder, and double the size of the projection head (from 2048 to 4096 for the hidden layer, and

Table 1. Evaluating the MoCLR baseline which is used in the proposed Divide And Contrast (DnC) model. This comparison is on the ImageNet linear classification benchmark. MoCLR is a contrastive method which achieves similar performance as BYOL, while requiring only a small change to SimCLR (see Section 3.1).

Method	Epochs	Top-1	Top-5
SimCLR [12]	1000	69.3	89.0
SimCLR v2 [13]	1000	71.7	90.4
MoCo v3 [16]	800	73.8	-
BYOL [35]	1000	74.3	91.6
MoCLR (ours)	1000	74.3	92.2

from 128 to 256 for the output layer). In comparison with BYOL [35], we remove the predictor head and use the contrastive loss instead of the mean squared prediction loss. While concurrent work MoCo v3 [16] inherits from BYOL the asymmetric "projector & predictor" design (the online encoder has an additional predictor compared to the momentum network), our MoCLR removes the predictor for simplicity.

In our experiments we set the batch size to 4096 and do not use a memory buffer [101, 92, 40]. As we will show in our experiments, with these simple changes our *MoCLR* baseline trained for 1,000 epochs outperforms SimCLR v1/v2 [13] and is on par with BYOL [35], see Table 1.

3.2. Divide and Contrast

The motivation behind Divide and Contrast is that, when training on diverse, large-scale datasets, the density of informative negatives will be sparse if we sample randomly from the whole dataset. Instead, if we contrast locally be-
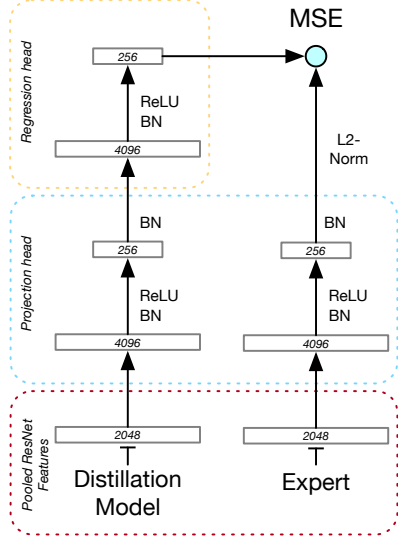
3

Figure 3. Detailed architectural overview of the distillation of the expert's features. The distillation model and experts are applied to the same augmented image view.

tween semantically-similar classes, the sampled negatives will be more informative and the learned model will capture a more discriminative representation.

As visualized in Figure 2, the training of our DnC model consists of three stages:

(1) We first train a MoCLR model on the given dataset for N_1 epochs (though other self-supervised learning methods can be used as well). We will call it the **base** model. We use the base model to extract representations for a set of samples in the training set, and cluster them in to K clusters. With these clusters we partition the dataset into K subsets.

(2) For each subset, we train a separate MoCLR model from scratch, which we call **expert** models. In this stage, we distribute a *total* computational budget of N_2 epochs (measured on the whole dataset) to these expert models, proportionally to their corresponding cluster sizes.

(3) Finally, given a base model that captures the general knowledge of the dataset and expert models focusing on locally similar categories, we distill knowledge from these models into a **distillation** model. In this stage, we train for N_3 epochs.

The encoder-architecture for the base model, expert models, and the distillation model are all identical. Therefore, the computational footprint can roughly be measured by summing up the training epochs across all three stages, resulting in a total training of $N_1 + N_2 + N_3$ epochs (except for the clustering overhead and extra forward pass during distillation, which we discuss later).

3.3. Distillation

To leverage the information learned by each of the different experts and more general information from the base model, we distill their representation into one model in the last stage of training. During the distillation we use a single augmented image (instead of the 2-view setup) in combination with a simple regression loss to predict the representations in these models (no contrastive loss).

The distillation model's architecture is mostly identical to the other models and is visualized in Figure 3. All have a backbone encoder $f(\cdot)$ and an MLP projection head $g(\cdot)$. On top of the projection head in the distillation model there are $K + 1$ regression networks: $r_k(\cdot)$, $k = 1, ..., K$, one to predict each of the K expert models and another regression network $r_b(\cdot)$ to predict the base model. The architecture of these regressors is the same as the projecton head, except that we remove the final global BatchNorm after the last output layer.

For distillation we use the same augmentation as during self-supervised learning. Given an augmented input image x with clustering id k, we feed it into the distillation model to produce the projection-head output z. Similarly we get z_b and z_k from the base model and the k-th expert model respectively. We also ℓ_2-normalize z_b and z_k to be unit-norm. The distillation objective is then the average of the two mean squared errors:

$$\mathcal{L}(x) = \frac{1}{2} \| r_b(z) - z_b \|_2^2 + \frac{1}{2} \| r_k(z) - z_k \|_2^2 \qquad (2)$$

Note that the outputs of r_b and r_k are *not* ℓ_2 normalized. To make it possible to compare to our baseline methods in terms of the number of epochs trained, we use two augmented views from the same input image and average their losses. This is otherwise not necessary and alternatively one could also increase the batch-size.

The computational cost in this stage is slightly higher than the self-supervised learning stage (e.g., BYOL and MoCLR), as there are now **two** forward passes (for the expert and base model) for each view (not backward pass and gradient computation). In contrast, BYOL and MoCLR only need **one** forward pass from the momentum encoder. However, we found that always feeding a center crop to expert and base models only leads to very marginal drop in performance (instead of an augmented view). This strategy offers the possibility of first doing a single forward pass over the dataset and storing the activations offline.

4. Experiments

In this section, we compare DnC to BYOL and MoCLR by pre-training on two large-scale uncurated datasets and evaluating transfer performance on different downstream tasks.
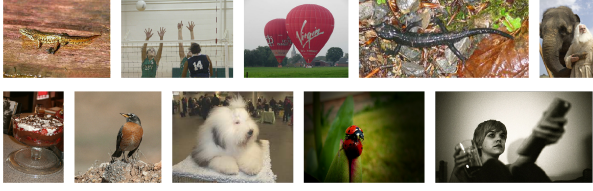
Figure 4. Example ImageNet images

Figure 5. Example YFCC100M images

Datasets. We consider two large-scale uncurated datasets. The first is a private dataset of roughly 300 million images (JFT-300M [89]). For the second dataset we use YFCC100M [91], a public dataset of 95M Flickr images available under the Creative Commons license. Figures 4 and 5 show a visual comparison between images from ImageNet and YFCC100M. ImageNet images often contain the object or animal of interest in the center of the image. ImageNet also does not have a long-tailed distribution (e.g., power law) over object-classes but only considers a specific set of 1000 different classes, which are (roughly) equally represented in the dataset. As a result, specific objects or animals (e.g., common tench, Bedlington terrier, . . .) are over-represented compared to more typically occurring scenes such as human faces and landscapes (which are better represented in YFCC100M).

Settings. ResNet-50 [42] is used in all experiments, unless noted otherwise. For ease of comparison, we report the computational footprint of all experiments in ImageNet-epoch equivalents (*e.g.* 1 "epoch" = 1281167/batch_size iterations). More implementation and optimization details are included in Appendix.

Table 2. We consider three different training schedules for DnC. The number of total training epochs in each stage, as well as the number of clusters, is specified.

Schedules	Base epochs	Experts total epochs	Distillation epochs
1,000 epochs	200	600 (5 clusters)	200
3,000 epochs	1,000	1,500 (5 clusters)	500
4,500 epochs	1,000	3,000 (10 clusters)	500

DnC Schedules. Table 2 shows three training schedules with different number of epochs. For example, in the schedule of 3,000 epochs, we first train the base model for 1,000 epochs, after which we cluster the samples into 5 groups. The 5 experts are trained in parallel on these subsets. We

Table 3. Comparison of self-supervised learning methods pre-trained on uncurated datasets YFCC100M and JFT-300M. For evaluation a linear classifier is trained on ImageNet and Places-365. Computation is measured as ImageNet-equivalent epochs.

Method	Arch	pre-training # epochs	ImageNet Top-1 Acc	Places 365 Top-1 Acc
Concurrent work trained on IG 1B images:				
SEER [32]	R-50	≈1,000	61.6	-
	R-101	≈1,000	65.8	-
Pre-training on YFCC100M:				
MoCLR		1,000	65.1	53.2
BYOL	R-50	1,000	65.3	52.9
MoCLR		3,000	65.7	53.2
BYOL	R-50	3,000	66.6	52.9
DnC		3,000	**67.8**	**54.1**
MoCLR		5,000	66.1	53.5
BYOL	R-50	5,000	67.0	53.2
DnC		4,500	**68.5**	**54.4**
Pre-training on JFT-300M:				
MoCLR		1,000	66.6	52.1
BYOL	R-50	1,000	67.0	51.9
DnC		1,000	**67.9**	**52.5**
MoCLR		3,000	67.4	52.5
BYOL	R-50	3,000	67.6	52.4
DnC		3,000	**69.8**	**53.3**
MoCLR		5,000	67.6	52.4
BYOL	R-50	5,000	67.9	52.4
DnC		4,500	**70.7**	**53.5**
With larger ResNet:				
MoCLR	R-200x2	3,000	74.2	54.6
DnC		3,000	**77.3**	**56.2**

use 1,500 epochs in total, spread out over the experts according to the number of images in each cluster (300 on average per expert). The distillation model is then trained for 500 epochs. See Section D for analysis of run time.

4.1. Linear Evaluation on ImageNet and Places-365

Table 3 shows the results of models pre-trained on YFCC100M and JFT-300M and tested on ImageNet and Places-365 [116] with linear evaluation, *i.e.*, features are frozen and a linear classifier is trained. For JFT-300M the results are also visualized in Figure 1. On the ImageNet linear benchmark we see a large drop in performance compared to pre-training on ImageNet (Table 1): **-9.0%**, **-7.3%** for BYOL-1k and **-9.2%**, **-7.7%** for MoCLR-1k, showing the difficulty of learning representations from uncurated (and more diverse) data.

In these experiments DnC always uses MoCLR-1k for the clustering, and uses the remaining pre-training epochs for the expert training and distillation. Therefore

Table 4. Transfer learning experiments. We evaluate models pre-trained on ImageNet, YFCC100M and JFT-300M with a linear classifier on 12 downstream classification tasks: Food-101 [6], CIFAR-10/100 [56], Birdsnap [4], SUN397 [102], Stanford Cars [55], FGVC Aircraft [65], PASCAL VOC 2007 [27], Describable Textures (DTD) [19], Oxford-IIIT Pets [75], Caltech-101 [28] and Oxford 102 Flowers [71].

		Food-101	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers	Average
YFCC	BYOL-5k	69.1	85.8	66.8	**35.5**	64.1	50.1	**51.9**	82.5	74.5	74.0	**87.6**	95.8	69.8
	MoCLR-5k	68.4	87.6	69.7	30.5	63.9	41.0	46.7	82.4	76.2	68.5	86.0	93.0	67.8
	DnC-4.5k	**72.1**	**88.0**	**71.1**	**35.5**	**67.2**	**52.6**	49.2	**83.7**	**76.5**	**75.9**	87.0	**97.8**	**71.4**
JFT-300M	BYOL-5k	73.3	89.8	72.4	38.2	61.8	64.4	**54.4**	81.3	75.5	77.0	90.1	94.3	72.7
	MoCLR-5k	72.8	90.7	72.5	33.8	62.2	60.6	50.9	81.9	75.3	75.8	89.5	93.8	71.7
	DnC-4.5k	**78.7**	**91.7**	**74.9**	**42.1**	**65.0**	**75.3**	54.1	**83.1**	**76.6**	**86.1**	90.2	**98.2**	**76.3**

Table 5. Fine-tuning pre-trained model for transfer learning experiments, including object detection on COCO dataset, semantic segmentation on Pascal VOC 2012, and depth estimation on NYU v2 dataset. For the evaluation metrics of *rms* and *rel*, lower is better.

		COCO detection			COCO instance seg.			PASCAL seg.	NYU v2 depth estimation				
		AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}	mIoU	<1.25	$<1.25^2$	$<1.25^3$	rms↓	rel↓
	ImageNet Super.	39.5	60.1	43.3	35.4	56.9	38.1	74.4	81.1	95.3	98.8	0.573	0.127
YFCC	BYOL-5k	41.1	62.0	45.1	36.6	58.6	38.9	75.5	83.5	96.4	99.0	0.558	0.130
	MoCLR-5k	40.8	61.7	44.8	36.6	58.5	39.0	75.1	**86.7**	**97.4**	**99.3**	**0.503**	**0.117**
	DnC-4.5k	**41.5**	**62.5**	**45.6**	**37.0**	**59.3**	**39.6**	**76.6**	86.2	97.2	**99.3**	0.512	0.121
JFT-300M	BYOL-5k	40.6	61.2	44.3	36.2	58.1	38.8	75.8	84.4	96.5	99.0	0.544	0.129
	MoCLR-5k	41.1	62.0	45.4	36.9	58.9	39.5	76.1	**86.3**	**97.2**	99.3	0.513	0.120
	DnC-4.5k	**41.7**	**62.5**	**45.9**	**37.2**	**59.3**	**39.8**	**76.9**	86.1	**97.2**	**99.4**	0.509	0.119

a good comparison is with MoCLR trained for longer (from scratch). For 3,000 epochs of training, MoCLR-3k improves over MoCLR-1k by **+0.6** and **+0.8** on YFCC100M and JFT-300M respectively, while DnC-3k improves by **+2.7** and **+3.2**. On Places-365 we see similar relative improvements. We also include BYOL-3k for completeness and again see small improvements with longer training for BYOL. For further longer schedules (4,500-5,000 epochs), we notice similar behavior on both YFCC100M and JFT-300M. Besides, we see DnC significantly outperforms concurrent efforts SEER [32] when using ResNet-50. We further test DnC with a larger model (*i.e.*, ResNet-200 with a width multiplier of 2) and observe that DnC outperforms MoCLR by **+3.1**.

4.2. Transfer Learning

In this section, we consider both using frozen representations for fine-grained linear classification and fine-tuning for different downstream tasks.

Fine-grained linear classification. Following Sim-CLR [12] and BYOL [35], we further perform linear classification evaluation on 12 classification datasets (introduced by [54]), to assess whether the learned representation is generic across different image domains (see more details in Section F.1). As shown in Table 4, when pre-training

on YFCC100M or JFT-300M, DnC significantly and consistently outperforms BYOL and MoCLR.

Detection, segmentation, and depth estimation. In Table 5, we evaluate the representation on three different fine-tuning tasks: (1) for object detection and instance segmentation on COCO [61], we train a standard Mask-RCNN [41] using FPN [60] with a 1x schedule, *i.e.*, 12 epochs; (2) for semantic segmentation on VOC2012, we used FCN [63] as in [40]; (3) for depth estimation on NYU-v2 dataset [70], the setup is the same as [35]. In all three tasks, DnC significantly outperforms ImageNet supervised pre-training, *e.g.*, **+2.2** in AP^{bb} and **+1.8** in AP^{mk} for detection, **+2.5** in mIoU for segmentation, and **+5.0** in <1.25 metric for depth prediction. DnC also significantly outperforms both self-supervised baselines when transferring to PASCAL and COCO tasks, and performs on-par with MoCLR while outperforming BYOL for depth estimation.

For more implementation details, please refer to Section F in the Appendix; Complete results for transfer learning are included in Section I.4.

5. Hypothesis and Analysis

The Divide and Contrast (DnC) method hinges on two main hypotheses. The first hypothesis is that clustering activations of powerful self-supervised learning models should

6

Table 6. Evaluating clustered representations from self-supervised learning methods using Top-1 accuracy and Mutual Information (MI) with the class labels. The representations of each method are clustered using k-means with 1000 centroids. To compute Top-1 accuracy, every cluster is mapped to the most frequent class label for the images in that cluster.

Method	Layer	Dimension	Top-1 Acc	MI
SimCLR [12]	pool	2048	30.1	5.64
	hidden	2048	33.3	5.83
	output	128	31.8	5.68
BYOL [35]	pool	2048	39.9	6.23
	hidden	4096	51.0	6.99
	output	256	50.0	6.78
MoCLR (ours)	pool	2048	40.1	6.26
	hidden	4096	**51.6**	**7.19**
	output	256	49.8	7.11

provide us with *locally consistent* clusters of images (*e.g.* having similar class labels). The second is that contrasting against similar (but different) object categories allows self-supervised methods to learn more fine-grained, discriminative representations.

We empirically assess these hypotheses in isolation. Next we compare DnC with current state of the art methods on ImageNet to see how well it performs on standard (curated) datasets, and analyze the design choices of DnC.

5.1. Clustered Representations are Object Categories

Our first hypothesis is that clusters of self-supervised representations are semantically meaningful. To this effect we cluster the representations of various self-supervised learning methods trained on ImageNet with k-means. Specifically, we consider representations from three different layers: the `pool` layer right after the mean pooling, the `hidden` layer of the projection head, and the final `projection`.

We start with 1000-way clustering, and assign every cluster to a single ImageNet class with a simple majority vote to measure the Top-1 Accuracy. We also measure the mutual information between the clustering assignments and the class labels. Table 6 gives an overview of these results. In particular, these methods can group images from the same category surprisingly well, with some representations achieving over 50% Top-1 clustering accuracy. We also notice that the `hidden` layer performs the best for all methods, and thus use this layer for clustering in the DnC method.

To give an orthogonal view with a smaller number of clusters, Figure 6 plots the 5 clusters used in the DnC model for ImageNet (based on the clustering of a MoCLR ResNet-50). Qualitatively, it appears that groups of classes are

Table 7. Linear classification evaluated on the Canine subset of ImageNet (130 classes). The feature extraction models were pre-trained either on ImageNet (Full) or the subset of Canine images without using any labels. All computation is reported in terms of "full imagenet" epochs. We report the difference in performance relative to training on the full dataset for 1,000 epochs. Even though the canine-only models were trained with 5× fewer gradient updates, they largely outperform the self-supervised learning models that were trained on the full dataset. We also observe that contrastive methods (SimCLR, MoCLR) benefit the most.

Method	Pre-training Dataset	Epochs	Top-1 Acc
SimCLR	Full	200	67.4
BYOL	Full	200	70.7
MoCLR	Full	200	68.7
SimCLR	Full	1,000	69.4
BYOL	Full	1,000	76.5
MoCLR	Full	1,000	75.3
SimCLR	Canine	100	72.8 (+3.4)
BYOL	Canine	100	76.0 (-0.5)
MoCLR	Canine	100	76.1 (+0.8)
SimCLR	Canine	200	74.1 (+4.7)
BYOL	Canine	200	77.3 (+0.8)
MoCLR	Canine	200	**77.5 (+2.2)**

jointly assigned to the same cluster. Indeed, the fraction of images in each class that belong to the same cluster is 87.4%, lending further evidence that individual clusters are semantically coherent.

5.2. Training on Semantically Similar Data Subsets

DnC is based on the second hypothesis that training self-supervised learning methods on a subset of images from similar object categories should improve performance on those object classes. Contrastive methods in particular stand to benefit from this procedure, as distinguishing positive samples from negatives from nearby classes might require learning more fine-grained features (similarly to hard-negative mining [84, 48]). On the other hand, it might hinder performance by drawing negative samples that are too similar, including more false-negatives [17].

To test this hypothesis in isolation and gain a better intuitive understanding of our method, we train various self-supervised learning models on the subset of ImageNet classes that belong to the canine family (including dogs, wolves and foxes, 130 classes in total) and compare them to models trained on the full dataset. For all models, we train a linear classifier on the canine-only subset, and evaluate on validation images from the canine subset.

From Table 7 it can be seen that models pre-trained on the canine-only subset perform significantly better than those trained on the entire ImageNet dataset, even though they have significantly less images to learn from and were
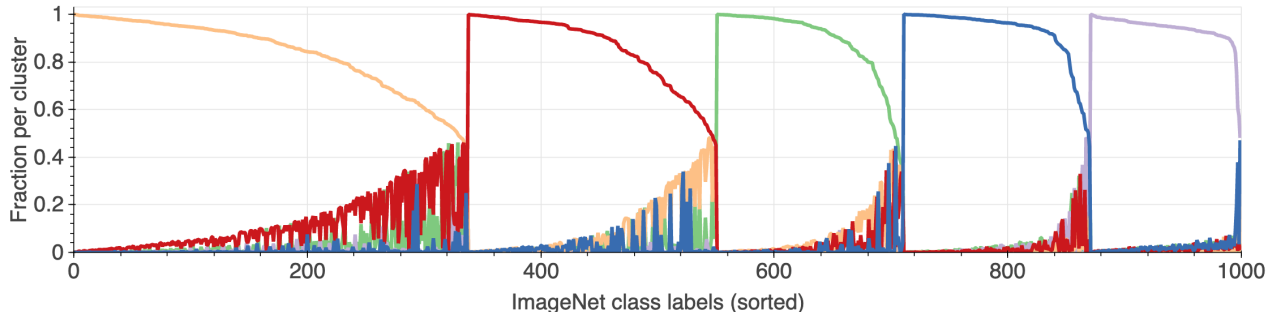
Figure 6. Visualization of the 5-way ImageNet clustering used in DnC. For each ImageNet class we compute the fraction of images that belong to each cluster. For better visualization the x-axis was sorted per cluster. From the figure it is clear that most images in each class belong to a single cluster. The fraction of images in each class that belong to the same cluster is **87.4%**.

trained with 5× less computation.

5.3. ImageNet Results

Even though our main goal is to improve self-supervised learning on uncurated datasets, we asked whether it remains competitive on heavily studied datasets such as ImageNet.

Table 8. Comparison with BYOL and MoCLR on ImageNet linear evaluation benchmark with training budgets of 1000 and 3000 epochs. Top-1 accuracy is reported using ResNet-50.

Method	1000 epochs	3000 epochs	Δ
BYOL	74.3	73.9	-0.4
MoCLR	74.3	74.5	0.2
DnC (ours)	**74.5**	**75.8**	**1.3**

From Table 8 we see that training the baseline MoCLR for 2,000 more epochs does not improve the results by much (+0.2). DnC on the other hand convincingly outperforms the baseline (**+1.3**). Interestingly, DnC even slightly outperforms MoCLR or BYOL when giving a computational budget of 1,000 epochs. Though DnC aims at uncurated datasets, the previous results from Section 5.2 on the canine subset have shown that even on ImageNet it might be beneficial to draw negatives from more similar categories.

5.3.1 Semi-supervised learning

We evaluate the performance of DnC when fine-tuning on a subset of ImageNet's train set. Following the semi-supervised protocol [54, 109, 12, 35], we use the same splits of 1% and 10% ImageNet data as in [12, 35]. As shown in Table 9, DnC consistently outperforms BYOL, SwAV, and MoCLR and Barlow Twins.

5.3.2 Ablations

We provide further experiments for isolating the factors that make DnC work, shown in Table 10. If we train the expert models on the full dataset instead of subsets (similar to an

Table 9. Semi-supervised results with a fraction of ImageNet labels following the protocol of [12, 35]. The encoder is ResNet-50.

method	Top-1			Top-5		
	Label fraction			Label fraction		
	1%	10%	100%	1%	10%	100%
SimCLR [12]	48.3	65.6	76.0	75.5	87.8	93.1
BYOL [35]	53.2	68.8	77.7	78.4	89.0	93.9
SwAV [10]	53.9	70.2	-	78.5	89.9	-
MoCLR	53.0	68.8	77.4	79.1	89.6	94.0
Barlow Tw. [108]	55.0	69.7	-	79.2	89.3	-
DnC	**59.9**	**71.1**	**78.2**	**83.0**	**90.4**	**94.2**

Table 10. Ablating the experts used in DnC: We notice a big drop in performance if the experts models are trained on the whole dataset (ensemble), or on random subsets.

	Partitioning	Experts trained on	Top-1 Acc
DnC	Clustering	local partition	75.8
- local experts	-	full dataset	74.3
- clustering	Randomly	local partition	73.1

Table 11. Evaluation of what models to predict during distillation.

base model	local experts	use center-crop	Top-1 Acc
✓			74.5
	✓		75.2
✓	✓		**75.8**
✓	✓	✓	75.6

ensemble), but with the same computational budget, the resulting model achieves the same performance as the base model (no improvement). Alternatively, splitting the dataset into random subsets hurts the final performance, showing the importance of the clustering used.

In Table 11 we study the distillation process, showing it is important to regress to both the base model and experts. And as discussed in Section 3.3, using center-crops instead of augmented views does not hurt the performance by much.

6. Conclusion

In this paper we have studied how state of the art self-supervised learning methods perform when they are pre-trained on uncurated data – datasets that did not require human annotations or labels to create – as a step towards fully self-supervised learning. We have observed that current methods suffer from a large drop in performance of up to **-9%** when pre-trained on these uncurated datasets. To alleviate this issue, we have proposed Divide and Contrast (DnC) that requires a few simple changes to existing self-supervised learning methods, and which largely outperforms state of the art SSL methods on uncurated datasets, as well as achieving similar or better performance on ImageNet. We hope this work draws more attention to uncurated datasets as a benchmark for self-supervised learning.

References

[1] Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. *arXiv:1911.12667*, 2019. 2

[2] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv:1902.09229*, 2019. 2

[3] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv:1906.00910*, 2019. 2, 15

[4] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *CVPR*, 2014. 6, 14, 17

[5] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. *arXiv:1704.05310*, 2017. 2

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014. 6, 14, 17

[7] Yue Cao, Zhenda Xie, Bin Liu, Yutong Lin, Zheng Zhang, and Han Hu. Parametric instance classification for unsupervised visual feature learning. *NeurIPS*, 2020. 2, 15

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

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

[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. 2, 8, 15

[11] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020. 2

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv:2002.05709*, 2020. 1, 2, 3, 6, 7, 8, 13, 14, 15, 16

[13] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020. 1, 2, 3, 15, 16

[14] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016. 2

[15] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020. 2, 15

[16] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised visual transformers. *arXiv preprint arXiv:2104.02057*, 2021. 3

[17] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. *NeurIPS*, 2020. 7

[18] Soo-Whan Chung, Joon Son Chung, and Hong-Goo Kang. Perfect match: Improved cross-modal embeddings for audio-visual synchronisation. In *ICASSP*, 2019. 2

[19] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 6, 14, 17

[20] 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

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018. 1

[22] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 2

[23] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *ICCV*, 2017. 2

[24] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2017. 2

[25] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *NeurIPS*, 2019. 2

[26] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, 2014. 2

[27] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 6, 14, 17

[28] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR*, 2004. 6, 14, 17

[29] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv:1803.07728*, 2018. 2

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[31] Daniel Gordon, Kiana Ehsani, Dieter Fox, and Ali Farhadi. Watching the world go by: Representation learning from unlabeled videos. *arXiv:2003.07990*, 2020. 2

[32] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021. 2, 5, 6

[33] 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. *arXiv:1706.02677*, 2017. 14

[34] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *ICCV*, 2019. 2

[35] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *NeurIPS*, 2020. 1, 2, 3, 6, 7, 8, 13, 14, 15

[36] Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. *arXiv:2004.14646*, 2020. 2

[37] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 2

[38] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *ICCV Workshop*, 2019. 2

[39] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. *arXiv:2008.01065*, 2020. 2

[40] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 2, 3, 6, 14, 15

[41] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 6, 14

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

[43] Olivier J Hénaff, Skanda Koppula, Jean-Baptiste Alayrac, Aaron van den Oord, Oriol Vinyals, and João Carreira. Ef-ficient visual pretraining with contrastive detection. *arXiv preprint arXiv:2103.10957*, 2021. 2

[44] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv:1905.09272*, 2019. 1, 2, 15

[45] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015. 2

[46] Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. *arXiv:1810.02334*, 2018. 2

[47] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019. 2

[48] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *NeurIPS*, 2018. 7

[49] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *NeurIPS*, 2020. 2

[50] Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. *NeurIPS*, 2020. 2

[51] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013. 2

[52] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *CVPR*, 2019. 14

[53] Lingpeng Kong, Cyprien de Masson d'Autume, Lei Yu, Wang Ling, Zihang Dai, and Dani Yogatama. A mutual information maximization perspective of language representation learning. In *ICLR*, 2020. 2

[54] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, 2019. 6, 8, 14

[55] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. 6, 14, 17

[56] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 6, 14, 17

[57] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, 2016. 15

[58] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv:2005.04966*, 2020. 2, 15

[59] Tianhao Li and Limin Wang. Learning spatiotemporal features via video and text pair discrimination. *arXiv:2001.05691*, 2020. 2

[60] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 6, 14

[61] 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. 6

[62] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *ICLR*, 2018. 2

[63] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 6, 14

[64] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR*, 2017. 14

[65] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv:1306.5151*, 2013. 6, 14, 17

[66] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. *arXiv:1912.06430*, 2019. 2

[67] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013. 2

[68] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. *arXiv:1912.01991*, 2019. 2, 15

[69] Pedro Morgado, Nuno Vasconcelos, and Ishan Misra. Audio-visual instance discrimination with cross-modal agreement. *arXiv:2004.12943*, 2020. 2

[70] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 6

[71] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008. 6, 14, 17

[72] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 2

[73] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018. 1, 2, 3

[74] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *ECCV*, 2020. 2

[75] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 6, 14, 17

[76] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017. 2

[77] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

[78] Mandela Patrick, Yuki M Asano, Ruth Fong, João F Henriques, Geoffrey Zweig, and Andrea Vedaldi. Multi-modal self-supervision from generalized data transformations. *arXiv:2003.04298*, 2020. 2

[79] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv:2007.13916*, 2020. 2

[80] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015. 2

[81] Anyi Rao, Jiaze Wang, Linning Xu, Xuekun Jiang, Qingqiu Huang, Bolei Zhou, and Dahua Lin. A unified framework for shot type classification based on subject centric lens. 2020. 2

[82] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv:1412.6550*, 2014. 2

[83] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv:1904.05862*, 2019. 1

[84] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 2, 7

[85] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018. 2

[86] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, 2016. 2

[87] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv:2004.04136*, 2020. 2

[88] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. *arXiv:1906.05743*, 2019. 2

[89] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017. 5

[90] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 3

[91] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 2016. 1, 2, 5

[92] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv:1906.05849*, 2019. 1, 2, 3, 14, 15

[93] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020. 2

[94] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. In *NeurIPS*, 2020. 2, 15

[95] Christopher Tosh, Akshay Krishnamurthy, and Daniel Hsu. Contrastive learning, multi-view redundancy, and linear models. *arXiv:2008.10150*, 2020. 2

[96] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Unsupervised semantic segmentation by contrasting object mask proposals. *arXiv preprint arXiv:2102.06191*, 2021. 2

[97] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 2

[98] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *ECCV*, 2018. 2

11

[99] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *arXiv:2005.10242*, 2020. 2

[100] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 2

[101] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018. 1, 2, 3, 15

[102] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 6, 14, 17

[103] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. *arXiv:2008.05659*, 2020. 2

[104] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas J Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. *arXiv:2007.10985*, 2020. 2

[105] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019. 1, 2

[106] Asano YM., Rupprecht C., and Vedaldi A. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 2, 15

[107] Yang You, Igor Gitman, and Boris Ginsburg. Scaling SGD batch size to 32k for imagenet training. *arXiv:1708.03888*, 2017. 14

[108] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021. 8

[109] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *ICCV*, 2019. 8

[110] Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. Online deep clustering for unsupervised representation learning. In *CVPR*, 2020. 2

[111] Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. In *CVPR*, 2019. 2

[112] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016. 2, 14

[113] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017. 2

[114] Xiao Zhang and Michael Maire. Self-supervised visual representation learning from hierarchical grouping. *arXiv preprint arXiv:2012.03044*, 2020. 2

[115] Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? *arXiv:2006.06606*, 2020. 2

[116] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017. 5

[117] Chengxu Zhuang, Alex Andonian, and Daniel Yamins. Unsupervised learning from video with deep neural embeddings. *arXiv:1905.11954*, 2019. 2

[118] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. *arXiv:1903.12355*, 2019. 2, 15

A. Image Augmentations

For a fair comparison, we used exactly the same image augmentations as BYOL [35] (which are a subset of the ones presented in SimCLR [12]):

- random resized cropping: a random patch is cropped, whose area is uniformly sampled between $0.08\times$ and $1\times$ that of the raw image, and aspect ratio is logarithmically sampled between $3/4$ and $4/3$. We resize the patch to 224×224 pixels using bicubic interpolation;

- random horizontal flip;

- color jittering: the brightness, contrast, saturation and hue of the image are shifted by a uniformly distributed offset applied on all the pixels of the same image;

- color dropping: randomly convert images to grayscale, computed as $0.2989R + 0.5870G + 0.1140B$;

- Gaussian blurring: a Gaussian kernel of size 23×23 is used, whose standard deviation is uniformly sampled from $[0.1, 2.0]$;

- solarization: an optional color transformation $x \mapsto x \cdot \mathbf{1}_{\{x<0.5\}} + (1-x) \cdot \mathbf{1}_{\{x \geqslant 0.5\}}$ for pixels with values in $[0, 1]$.

Augmentations from the sets $\mathcal{T}$ and $\mathcal{T}'$ are compositions of the above image augmentations, each applied with a predetermined probability. The parameters for $\mathcal{T}$ and $\mathcal{T}'$ are listed in Table 12.

In the evaluation or representation clustering stage, we follow the standard center-crop strategy: resize images to 256 pixels along the shorter side, and crop out the central 224×224 window.

Parameter	$\mathcal{T}$	$\mathcal{T}'$
Random crop probability	1.0	1.0
Flip probability	0.5	0.5
Color jittering probability	0.8	0.8
Brightness adjustment max intensity	0.4	0.4
Contrast adjustment max intensity	0.4	0.4
Saturation adjustment max intensity	0.2	0.2
Hue adjustment max intensity	0.1	0.1
Color dropping probability	0.2	0.2
Gaussian blurring probability	1.0	0.1
Solarization probability	0.0	0.2

Table 12. Parameters used to generate image augmentations during training, which are exactly the same as those in [35].

B. Pre-trainning Datasets

ImageNet. We split out 10009 images from the train set as our local validation set, and use the remaining 1271158

images for both unsupervised pre-training and linear classifier training. After selecting hyper-parameters based on the performance of the local validation set, we report accuracy on the official validation set consisting of 50000 images.
JFT-300M. The JFT-300M dataset contains 301.7 millions of images in total.
YFCC100M. YFCC-100M is a widely used uncurated dataset that includes ∼95 millions of images, which are all used in our pre-training.

C. Clustering Representations

We apply the vanilla k-means algorithm on the representations extracted from the `hidden` layer of the projection network, with cosine similarity as a distance metric. When pre-training on ImageNet, we use all training images for clustering; when pre-training on JFT-300M and YFCC, we randomly sample 1.5 million images for clustering and extracting the centroids, and then use these centroids to assign clustering labels to all images.

D. Run Time Analysis of DnC

While DnC has three stages of training, its computational complexity or running time is similar as other state-of-the-art approaches trained for the same number of epochs. As a illustration, we compare the training FLOPs of DnC with other methods such as BYOL, MoCLR, and SwAV. As discussed in the main paper, the base or expert training stage of DnC has exactly the same FLOPs as the chosen base approach, *i.e.*, MoCLR here. The only different lies in the third stage, where DnC requires one additional forward pass. Therefore, for DnC we compute a weighted average of FLOPs across three stages (we use the normalized number of training epochs as weights). Table 13 summarizes the comparison with other approaches: DnC is comparable with BYOL and MoCLR, while SwAV costs more flops because it uses eight views per image per step.

Besides, we also run BYOL, MoCLR and DnC on ImageNet for 3000 epochs to compare the running time. Table 14 reports the comparison when using 512 TPU v3 cores. DnC only introduces <5% extra training time, compared with BYOL and MoCLR. Besides, the time cost for clustering the representations is small, *e.g.*, it takes about 20-30 minutes to extract representations on the training set and cluster them into groups, even only with 8 V100 GPUs on a single node.

Table 13. Training FLOPs of different methods per image when using ResNet-50.

	SwAV	BYOL	MoCLR	DnC
training FLOPS	38.4B	24.7B	24.7B	25.4B

Table 14. Training time for 3000 epochs on ImageNet when using 512 TPU v3 cores.

	BYOL	MoCLR	DnC
training time (hours)	∼24	∼24	∼25

E. Optimization

Unsupervised pre-training. All the hyper-parameters for optimization directly follow BYOL, except for base learning rate (which we discuss in the next paragraph). Specifically, we use LARS optimizer [107] with a cosine decay learning rate schedule [64] and a warm-up period of 10 epochs for all unsupervised pre-training. In addition, we use a global weight decay parameter of $1.5 \cdot 10^{-6}$ while excluding the biases and batch normalization parameters from both LARS adaptation and weight decay. For the momentum encoder, its parameters θ_{EMA} are updated by $\theta_{\text{EMA}} = \tau\theta_{\text{EMA}} + (1 - \tau)\theta$, where θ are the parameters of the online encoder. The EMA parameter τ starts from $\tau_{\text{base}} = 0.996$ and is increased to one during training. Following BYOL, we set

$$\tau \triangleq 1 - (1 - \tau_{\text{base}}) \cdot (\cos(\pi k/K) + 1)/2 \qquad (3)$$

with k the current training step and K the maximum number of training steps.

Specifically for the base learning rate, we used 0.2 for BYOL as in the original paper (we sweep over $\{0.2, 0.3, 0.4\}$ for 1000 epochs pre-training on ImageNet to confirm 0.2 is the best). For MoCLR, we found 0.3 is slightly better than 0.2, and therefore we kept using 0.3 for MoCLR and all stages of DnC (The only exception is that for DnC with 1000 epoch schedule, we increase the base learning rate to 0.5 to compensate for short training of models at each stage). The final learning rate is scaled linearly [33] with the batch size (LearningRate = BaseLearningRate × BatchSize/256).

Linear evaluation on ImageNet/Places-365. On top of the global pooling layer of the frozen pre-trained encoder, we train a supervised 1000- or 365-way linear classifier, as in [112, 92, 40, 12]. We optimize the cross-entropy loss using SGD with Nesterov momentum over 80 epochs, using a batch size of 1024 and a cosine learning rate decay schedule. We sweep the base learning rate (of batch size 256) over $\{0.4, 0.3, 0.2, 0.1, 0.05\}$ for models pre-trained on ImageNet, and $\{1.0, 0.6, 0.4, 0.2, 0.1\}$ for models pre-trained on JFT-300M and YFCC. We chose the best learning rate on a local validation set split out from the ImageNet train set, and report the accuracy on the official ImageNet validation set.

F. Transfer to Other Datasets

F.1. Implementation: fine-grained linear classificaton

We perform transfer via linear classification and fine-tuning on the same set of datasets as in [12, 35], namely Food-101 [6], CIFAR-10/100 [56], Birdsnap [4], SUN397 [102], Stanford Cars [55], FGVC Aircraft [65], PASCAL VOC 2007 classification task [27], Describable Textures (DTD) [19], Oxford-IIIT Pets [75], Caltech-101 [28] and Oxford 102 Flowers [71]. As in [12, 35], we used the validation sets specified by the dataset creators to select hyperparameters for FGVC Aircraft, PASCAL VOC 2007, DTD, and Oxford 102 Flowers. On other datasets, we use the validation examples as test set, and hold out a subset of the training examples as validation set while performing hyperparameter tuning.

We follow the linear evaluation protocol of [52, 54, 12, 35]. We train a regularized multinomial logistic regression classifier on top of the frozen representation without data augmentation. Images are resized to 224 pixels along the shorter side and cropped by the center 224×224 pixels. We minimize the cross-entropy objective using L-BFGS with ℓ_2-regularization, where we select the regularization parameters from a range of 45 logarithmically-spaced values between 10^{-6} and 10^5.

F.2. Implementation: Pascal VOC segmentation

Following BYOL, we use the same fully-convolutional network (FCN)-based [63] architecture as [40]. The backbone consists of the convolutional layers in ResNet-50. The 3×3 convolutions in the conv5 blocks use dilation 2 and stride 1. This is followed by two extra 3×3 convolutions with 256 channels, each followed by batch normalization and ReLU activations, and a 1×1 convolution for per-pixel classification. The dilation is set to 6 in the two extra 3×3 convolutions. The total stride is 16 (FCN-16s [63]).

Similar as BYOL, we train on the train2012 set and report results on val2012. Hyperparameters are selected on a 2119 images held-out validation set. Training is done with random scaling (by a ratio in $[0.5, 2.0]$), cropping, and horizontal flipping. The crop size is 513. Inference is performed on the $[513, 513]$ central crop. For training we use a batch size of 16 and weight decay of 0.0001. We select the base learning rate by sweeping across 5 logarithmically spaced values between 10^{-3} and 10^{-1}. The learning rate is multiplied by 0.1 at the 70-th and 90-th percentile of training. We train for 30000 iterations, and average the results on 5 seeds.

F.3. Implementation: COCO detection

We use the standard Mask R-CNN [41] with the FPN [60] backbone, with cross-replica BN tuned, similar

as that in MoCo [40]. We fine-tune all layers end-to-end. We finetune on the `train2017` set (~118k images) and evaluate on `val2017`. We use the standard "1x schedule".

We directly use the public Cloud TPU implementation without modification. Specifically, we use a batch size of 64 images split over 16 workers. We linearly warmup the learning rate to 0.3 for the first 500 iterations, and drop it twice by a factor of 2, after $\frac{2}{3}$ and $\frac{8}{9}$ of the total training steps.

F.4. Implementation: NYU v2 depth estimation

Similar as BYOL, we follow the same protocol as in [57]. With a standard ResNet-50 backbone, we feed the conv5 features into 4 fast up-projection blocks with respective filter sizes 512, 256, 128, and 64. We use a reverse Huber loss function for training.

The original NYU Depth v2 frames of size [640, 480] are down-sampled by a factor 0.5 and center-cropped to [304, 228] pixels. Input images are randomly horizontally flipped and the same set of color transformations as in [35] are applied. We train for 7500 steps with batch size 256, weight decay 0.0005, and learning rate 0.16 (scaled linearly from the setup of [57] to account for the larger batch size).

G. DnC with other Self-supervised Methods

While the main paper demonstrate the effectiveness of DnC with MoCLR, we found DnC can potentially improves other state-of-the-art self-supervised approaches as well. In Table 15, we demonstrate that DnC can improve SimCLR significantly and also benefit BYOL, when both pre-training and evaluating on ImageNet.

Table 15. Applying DnC with other self-supervised methods, when pre-training and evaluating on ImageNet.

Method	w/ DnC	Epochs	Accuracy (%)	Δ
SimCLR		1000	69.4	
		5000	70.2	+0.8
	✓	3000	73.0	+3.6
BYOL		1000	74.3	
		3000	73.9	-0.5
	✓	3000	75.1	+0.8

Besides, we notice that DnC with BYOL gets a larger improvement when pre-training on the uncurated dataset YFCC. As shwon in Table 16, naively extending BYOL from 1000 to 5000 epochs only increases the performance by 1.7%, while DnC-4500 leverages the computation more efficiently and improves the accuracy by 3.4%.

Table 16. Applying DnC with BYOL on YFCC pre-training. Evaluation is conducted on ImageNet linear evaluation.

Method	w/ DnC	Epochs	Accuracy (%)	Δ
BYOL		1000	65.3	
		3000	66.6	+1.3
		5000	67.0	+1.7
	✓	3000	67.9	+2.6
	✓	4500	68.7	+3.4

Table 17. Linear evaluation benchmark on ImageNet (self-supervised pre-training is also conducted on ImageNet). * indicates using eight views for pre-training.

Method	Epochs	Top-1 Acc	Top-5 Acc
Clustering methods with ResNet-50:			
SeLa [106]	400	61.5	84.0
DeepClusterV2* [10]	800	75.2	-
SwAV* [10]	800	75.3	-
Contrastive learning with designed architecture:			
AMDIM [3]	150	68.1	-
CMC [92]	240	70.6	89.7
Contrastive learning with ResNet-50:			
NPID [101]	200	56.5	-
Local Agg. [118]	200	58.8	-
CPC v2 [44]	-	63.8	85.3
MoCo [40]	200	60.6	-
PIRL [68]	800	67.4	-
PCL [58]	200	67.6	-
SimCLR [12]	1,000	69.3	89.0
PIC [7]	1,600	70.8	90.0
MoCo v2 [15]	800	71.1	-
SimCLR v2 [13]	1,000	71.7	90.4
InfoMin Aug. [94]	800	73.0	91.1
BYOL [35]	1,000	74.3	91.6
BYOL [35]	3,000	73.9	92.2
MoCLR (ours)	1,000	74.3	92.2
MoCLR (ours)	3,000	74.5	92.3
DnC (ours)	1,000	74.5	92.2
DnC (ours)	3,000	**75.8**	**92.8**

H. Comparing with SoTA on ImageNet

Though ImageNet linear evaluation benchmark (both pre-training and evaluating on ImageNet) is not the main focus of this paper, we still provides a comparison between DnC and recent SoTA methods, as shown in Table 17.

I. Additional Ablations and Results

I.1. Length of the distillation stage

While the distillation stage introduces additional FLOPs compared to the base or expert training stage, this stage can

be short. In this section, we conduct ablation on the number of epochs for distillation stage. We train DnC on ImageNet following the the DnC-3k schedule (*i.e.*, 1000 epochs for base training and 1500 epochs for experts). We vary the number of epochs used for distillation and report the linear evaluation accuracy in Table 18. Short distillation schedule such as 60 epochs can yield 74.0%, as long as a larger learning rate is utilized to compensate for the smaller number of gradient steps.

Table 18. Abalation on length of the distillation stage.

Epoch	60	100	200	300	500
Learning rate	0.45	0.45	0.35	0.35	0.3
Accuracy (%)	74.0	74.9	75.1	75.4	75.8

I.2. ℓ_2-normalization on regressor output

We study whether it's better to normalize the output of the regressor by ℓ_2-normalization in the distillation stage. We conducted this ablation on ImageNet, and found normalizing the output of the regressor actually hurts the performance a bit, as shown in Table 19.

Table 19. Abalation on whether ℓ_2-normalize the output of regressors r_b and $r_k (k = 1, 2, ...K)$.

ℓ_2-normalization	Accuracy
Yes	75.4
No	75.8

I.3. Semi-supervised learning with projection layer

As found in SimCLR v2 [13], fine-tuning from the hidden layer of projection head gives better semi-supervised accuracy. In this seciton, we also report the semi-supervised accuracy fine-tuned from the hidden layer of the projection head in Table 20. Models are all pre-trained using ImageNet data.

Table 20. Semi-supervised results by fine-tuning from the first layer of the projection head, following [12]. The encoder is ResNet-50.

method	Top-1		Top-5	
	Label fraction		Label fraction	
	1%	10%	1%	10%
SimCLR v2	57.9	68.4	82.5	89.2
BYOL	61.9	71.9	83.3	90.7
MoCLR	61.0	71.6	84.2	90.9
DnC	**65.6**	**73.2**	**86.4**	**91.4**

I.4. Complete results of transfer learning

In Table 21, we summarize the transfer learning results on fine-grained linear classification tasks, with different computational budgets in pre-training stage for each method.

In Table 22, we provide the complete results of transfer learning on COCO detection, Pascal VOC semantic segmentation, and NYU depth estimation.

Table 21. Transfer learning experiments. We evaluate models pre-trained on ImageNet, YFCC100M and JFT-300M with a linear classifier on 12 downstream classification tasks: Food-101 [6], CIFAR-10/100 [56], Birdsnap [4], SUN397 [102], Stanford Cars [55], FGVC Aircraft [65], PASCAL VOC 2007 [27], Describable Textures (DTD) [19], Oxford-IIIT Pets [75], Caltech-101 [28] and Oxford 102 Flowers [71].

		Food-101	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers	Average
YFCC	BYOL-1k	67.9	85.0	63.9	31.3	63.4	44.3	47.5	81.8	75.2	71.1	84.0	93.4	67.4
	BYOL-3k	68.8	86.5	66.6	33.2	63.9	46.5	49.8	82.3	73.6	73.9	86.5	95.4	68.9
	BYOL-5k	69.1	85.8	66.8	**35.5**	64.1	50.1	**51.9**	82.5	74.5	74.0	**87.6**	95.8	69.8
	MoCLR-1k	67.7	87.8	69.9	29.4	63.4	41.1	45.6	81.6	75.8	67.7	85.6	92.9	67.4
	MoCLR-3k	67.9	**88.3**	70.2	29.6	63.8	40.7	45.9	82.4	76.0	69.2	85.4	92.3	67.6
	MoCLR-5k	68.4	87.6	69.7	30.5	63.9	41.0	46.7	82.4	76.2	68.5	86.0	93.0	67.8
	DnC-3k	71.9	87.3	70.1	34.4	65.7	48.2	46.3	82.7	75.5	75.7	86.0	96.5	70.0
	DnC-4.5k	**72.1**	88.0	**71.1**	35.5	**67.2**	52.6	49.2	**83.7**	76.5	75.9	87.0	**97.8**	**71.4**
JFT-300M	BYOL-1k	72.7	90.1	71.7	33.9	61.0	62.4	52.1	81.1	74.9	76.0	89.0	94.3	71.6
	BYOL-3k	72.8	89.9	72.5	36,7	62.1	63.3	53.2	81.6	75.5	77.8	89.5	94.5	72.5
	BYOL-5k	73.3	89.8	72.4	38.2	61.8	64.4	**54.4**	81.3	75.5	77.0	**90.1**	94.3	72.7
	MoCLR-1k	71.9	90.4	72.7	32.8	61.3	59.3	51.6	81.5	75.4	74.5	89.3	93.9	71.2
	MoCLR-3k	72.7	90.8	73.0	33.5	62.2	59.8	51.6	81.4	**77.3**	76.2	88.7	93.5	71.7
	MoCLR-5k	72.8	90.7	72.5	33.8	62.2	60.6	50.9	81.9	75.3	75.8	89.5	93.8	71.7
	DnC-3k	74.8	**91.6**	**74.9**	38.2	63.8	68.6	53.4	**83.0**	77.1	82.5	**90.5**	97.2	74.6
	DnC-4.5k	**78.7**	91.7	74.9	**42.1**	**65.0**	**75.3**	54.1	83.1	76.6	**86.1**	90.2	**98.2**	**76.3**

Table 22. Fine-tuning pre-trained model for transfer learning experiments, including object detection on COCO dataset, semantic segmentation on Pascal VOC 2012, and depth estimation on NYU v2 dataset. For the evaluation metrics of *rms* and *rel*, lower is better.

		COCO object detection, 1x schedule						Seg.	NYU v2 depth estimation				
		AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}	mIoU	<1.25	$<1.25^2$	$<1.25^3$	rms↓	rel↓
	ImageNet Super.	39.5	60.1	43.3	35.4	56.9	38.1	74.4	81.1	95.3	98.8	0.573	0.127
ImageNet	BYOL-3k	40.9 (+1.4)	61.9	45.0	36.7 (+1.3)	58.5	39.2	76.3	84.7	97.0	99.1	0.525	0.126
	MoCLR-3k	41.5 (+2.0)	62.3	45.4	37.0 (+1.6)	59.0	39.7	76.2	84.6	97.0	99.3	0.527	0.126
	DnC-3k	41.7 (+2.2)	62.6	45.6	37.3 (+1.9)	59.2	40.1	76.9	85.1	97.0	99.2	0.525	0.124
YFCC	BYOL-1k	40.8 (+1.3)	61.9	45.0	36.4 (+1.0)	58.4	38.8	75.5	85.8	97.2	99.2	0.511	0.122
	BYOL-3k	41.0 (+1.5)	61.6	45.0	36.6 (+1.2)	58.5	39.2	75.5	85.2	96.9	99.0	0.537	0.124
	BYOL-5k	41.1 (+1.6)	62.0	45.1	36.6 (+1.2)	58.6	38.9	75.1	83.5	96.4	99.0	0.558	0.130
	MoCLR-1k	40.2 (+0.7)	61.1	44.2	36.0 (+0.6)	57.8	38.2	75.0	85.7	97.1	99.3	0.515	0.122
	MoCLR-3k	40.7 (+1.2)	61.6	44.4	36.3 (+0.9)	58.3	38.8	75.3	86.6	97.2	99.3	0.502	0.120
	MoCLR-5k	40.8 (+1.3)	61.7	44.8	36.6 (+1.2)	58.5	39.0	75.5	86.7	97.4	99.3	0.503	0.117
	DnC-3k	41.0 (+1.5)	61.6	44.7	36.6 (+1.2)	58.5	39.5	76.1	86.7	97.3	99.3	0.506	0.117
	DnC-4.5k	41.5 (+2.0)	62.5	45.6	37.0 (+1.6)	59.3	39.6	76.6	86.2	97.2	99.3	0.512	0.121
JFT-300M	BYOL-1k	40.5 (+1.0)	61.3	44.4	36.4 (+1.0)	58.2	38.8	75.5	85.8	97.1	99.2	0.519	0.121
	BYOL-3k	40.5 (+1.0)	61.1	44.7	36.4 (+1.0)	57.9	39.2	75.7	85.6	97.0	99.2	0.525	0.122
	BYOL-5k	40.6 (+1.1)	61.2	44.3	36.2 (+0.8)	58.1	38.8	75.8	84.4	96.5	99.0	0.544	0.129
	MoCLR-1k	40.3 (+0.8)	61.0	44.2	36.3 (+0.9)	58.0	38.8	75.7	84.9	96.8	99.2	0.526	0.126
	MoCLR-3k	40.5 (+1.0)	61.2	44.4	36.4 (+1.0)	58.1	39.0	75.8	85.9	97.2	99.3	0.514	0.121
	MoCLR-5k	41.1 (+1.6)	62.0	45.4	36.9 (+1.5)	58.9	39.5	76.1	86.3	97.2	99.3	0.513	0.120
	DnC-3k	41.6 (+2.1)	62.3	45.5	37.2 (+1.8)	59.1	39.8	76.8	86.0	97.3	99.3	0.517	0.119
	DnC-4.5k	41.7 (+2.2)	62.5	45.9	37.2 (+1.8)	59.3	39.8	76.9	86.1	97.2	99.4	0.509	0.119