Supplementary Material for "Divide and Contrast: Self-supervised Learning from Uncurated Data"

Yonglong Tian
MIT

Olivier J. Hénaff
DeepMind

Aaron van den Oord
DeepMind

A. Image Augmentations

For a fair comparison, we used exactly the same image augmentations as BYOL [13] (which are a subset of the ones presented in SimCLR [6]):

- 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 1.

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 1. Parameters used to generate image augmentations during training, which are exactly the same as those in [13].

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 $\sim$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

Table 2. 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

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 2 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 3 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 3. Training time for 3000 epochs on ImageNet when using 512 TPU v3 cores.

	BYOL	MoCLR	DnC
training time (hours)	$\sim$24	$\sim$24	$\sim$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 [35] with a cosine decay learning rate schedule [25] 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

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

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 [12] with the batch size (LearningRate $=$ BaseLearningRate $\times$ 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 [36, 30, 14, 6]. 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 [6, 13], namely Food-101 [3], CIFAR-10/100 [20], Birdsnap [2], SUN397 [33], Stanford Cars [19], FGVC Aircraft [26], PASCAL VOC 2007 classification task [10], Describable Textures (DTD) [9], Oxford-IIIT Pets [29], Caltech-101 [11] and Oxford 102 Flowers [28]. As in [6, 13], 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 [17, 18, 6, 13]. 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 [24] architecture as [14]. 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 [24]).

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 [15] with the FPN [23] backbone, with cross-replica BN tuned, similar as that in MoCo [14]. We fine-tune all layers end-to-end. We finetune on the `train2017` set ($\sim$118k images) and evaluate on `val2017`. We use the standard "1x schedule".

We directly use the public Cloud TPU implementation without modification[1]. 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 [21]. 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 [13] 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 [21] to account for the larger batch size).

[1]https://github.com/tensorflow/tpu/tree/master/models/official/detection

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 4, we demonstrate that DnC can improve SimCLR significantly and also benefit BYOL, when both pre-training and evaluating on ImageNet.

Table 4. 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 5, 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 5. 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**

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 6.

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

Table 6. 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 [34]	400	61.5	84.0
DeepClusterV2* [5]	800	75.2	-
SwAV* [5]	800	75.3	-
Contrastive learning with designed architecture:			
AMDIM [1]	150	68.1	-
CMC [30]	240	70.6	89.7
Contrastive learning with ResNet-50:			
NPID [32]	200	56.5	-
Local Agg. [37]	200	58.8	-
CPC v2 [16]	-	63.8	85.3
MoCo [14]	200	60.6	-
PIRL [27]	800	67.4	-
PCL [22]	200	67.6	-
SimCLR [6]	1,000	69.3	89.0
PIC [4]	1,600	70.8	90.0
MoCo v2 [8]	800	71.1	-
SimCLR v2 [7]	1,000	71.7	90.4
InfoMin Aug. [31]	800	73.0	91.1
BYOL [13]	1,000	74.3	91.6
BYOL [13]	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**

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 7. 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 7. 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 8.

Table 8. Ablation 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 [7], 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 9. Models are all pre-trained using ImageNet data.

Table 9. Semi-supervised results by fine-tuning from the first layer of the projection head, following [6]. 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 10, 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 11, we provide the complete results of transfer learning on COCO detection, Pascal VOC semantic segmentation, and NYU depth estimation.

References

[1] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv:1906.00910*, 2019. 4

[2] 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. 2, 5

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014. 2, 5

[4] Yue Cao, Zhenda Xie, Bin Liu, Yutong Lin, Zheng Zhang, and Han Hu. Parametric instance classification for unsupervised visual feature learning. *NeurIPS*, 2020. 4

Table 10. 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 [3], CIFAR-10/100 [20], Birdsnap [2], SUN397 [33], Stanford Cars [19], FGVC Aircraft [26], PASCAL VOC 2007 [10], Describable Textures (DTD) [9], Oxford-IIIT Pets [29], Caltech-101 [11] and Oxford 102 Flowers [28].

		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 11. 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

[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. 4

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning

of visual representations. *arXiv:2002.05709*, 2020. 1, 2, 4

[7] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020. 4

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

[9] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 2, 5

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 2, 5

[11] 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. 2, 5

[12] 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. 2

[13] 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, 4

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2, 3, 4

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 3

[16] 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. 4

[17] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *CVPR*, 2019. 2

[18] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, 2019. 2

[19] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. 2, 5

[20] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 2, 5

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

[22] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv:2005.04966*, 2020. 4

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 3

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 3

[25] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR*, 2017. 2

[26] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv:1306.5151*, 2013. 2, 5

[27] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. *arXiv:1912.01991*, 2019. 4

[28] 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. 2, 5

[29] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 2, 5

[30] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv:1906.05849*, 2019. 2, 4

[31] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. In *NeurIPS*, 2020. 4

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

[33] 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. 2, 5

[34] Asano YM., Rupprecht C., and Vedaldi A. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 4

[35] Yang You, Igor Gitman, and Boris Ginsburg. Scaling SGD batch size to 32k for imagenet training. *arXiv:1708.03888*, 2017. 2

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

[37] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. *arXiv:1903.12355*, 2019. 4