PARTICLE: Part Discovery and Contrastive Learning for Fine-grained Recognition

Anonymous ICCV submission

Paper ID 11

Abstract

We develop techniques for refining representations for fine-grained classification and segmentation tasks in a self-supervised manner. Current fine-tuning methods based on instance-discriminative contrastive learning are not as effective, possibly due to object pose and background, which are highly discriminatory for instances but act as a nuisance factor for categorization. We present an iterative learning approach that incorporates part-centric equivariance and invariance objectives. First, pixel representations are clustered in a part discovery step, where we analyze the representations from convolutional and vision transformer networks best suited for this. Then, a part-centric learning step aggregates and contrasts representations of parts within an image. We show that this improves the downstream performance on image classification and part segmentation tasks across datasets. For example, under a linear-evaluation scheme, the classification accuracy of a ResNet50 architecture trained using a self-supervised learning approach called DetCon [18] on ImageNet, improves from 35.4% to 42.0% on the Caltech-UCSD birds dataset, from 35.5% to 44.1% on the FGVC aircraft dataset, and 29.7% to 37.4% on Stanford Cars dataset. We also observe significant gains in few-shot part segmentation tasks in these datasets, while in both cases instance-discriminative learning was not as effective. Smaller, yet consistent, improvements are also observed for stronger baseline models based on vision transformers. We present experiments that evaluate the significance of pre-trained networks and techniques for part-discovery for downstream tasks.

1. Introduction

Contrastive learning based on instance discrimination has emerged as a leading self-supervised learning (SSL) technique (*e.g.*, [6, 13, 15, 20, 40]) for a wide range of image understanding tasks. Yet, their performance on fine-grained categorization has been lacking compared to the supervised counterparts especially in the few-shot setting [10, 32]. Instances within a category often appear in a wide variety of poses and backgrounds which is highly discriminative of instances which can be a nuisance factor for categorization. At the same time appearance of parts are discriminative of categories and thus part-centric appearance have often been used to improve performance on fine-grained recognition tasks [3, 23, 33, 39].

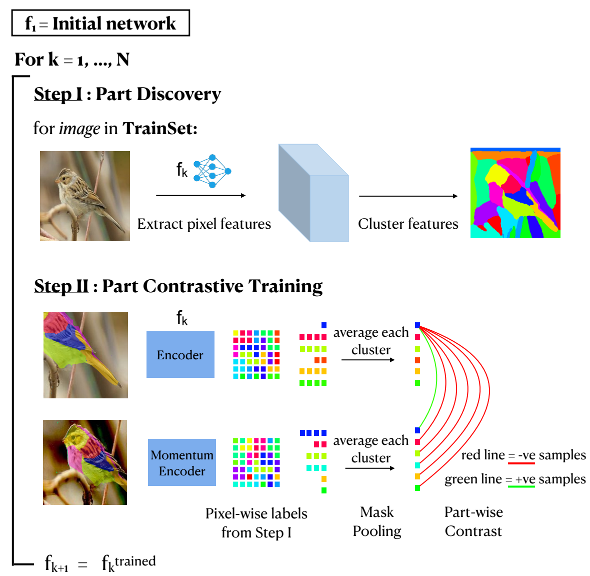

Figure 1. **Self-supervised fine-tuning using part discovery and contrastive learning (PARTICLE).** Given a collection of unlabeled images, at each iteration we cluster pixels features from an initial network to obtain part segmentations (§ 3.1), and fine-tune the network using a contrastive objective between parts (§ 3.2).

Thus we develop an approach for fine-tuning representations that is especially suited for fine-grained classification and segmentation tasks (e.g., recognizing species of birds and segmenting their parts). Our approach shown in Fig. 1 consists of two steps. First, we discover parts within an image by clustering pixel representations using an initial network. This is done by clustering hypercolumn representations of CNNs [7, 14], or patch embedding of vision transformers (Step I). We then train the same network using an objective where we aggregate and contrast pixel representations across parts within the same image (Step II). Similar to prior work (*e.g.*, [5, 7, 18, 34]) we learn invariances and equivariances through data augmentations. The resulting network is then used to re-estimate part segmentations and the entire process repeated (see Algorithm 1). Our approach, for **part di**scovery and **c**ontrastive **le**arning (PARTICLE) can be used to adapt representations to new domains in an entirely self-supervised manner.

We test our approach for adapting residual networks (ResNet50) [17], as well as vision transformers (ViTs) [11] trained on ImageNet [27] using self-supervision techniques to fine-grained domains. We consider two tasks: 1) classification under a linear evaluation, and 2) part segmentation with a few labeled examples. For ResNet50 networks trained with DetCon [18], PARTICLE improves the classification accuracy from 35.4% to 42.0% on Caltech-UCSD birds [38] and 35.5% to 44.1% on FGVC aircrafts [24], closing the gap over ImageNet supervised variant. On part-segmentation our approach leads to significant improvements over both the baseline and supervised ImageNet networks. Similar gains are also observed for networks trained using momentum-contrastive learning (MoCov2 [16]). ViTs, in particular, those trained with DINO [4] are highly effective, surpassing the supervised ResNet50 ImageNet baseline, but our approach improves the classification accuracy from 83.3% to 84.2% on birds, 72.4% to 73.6% on aircrafts, and 72.7% to 73.9% on cars while showing larger gains on the part segmentation tasks. Notably, the same objective (i.e., MoCo, DetCon, or DINO) yield significantly smaller, and sometimes no improvements across the tasks and datasets (Tab. 1), in comparison to PARTICLE.

We also systematically evaluate the effectiveness of various representations for part discovery. Parts generated by color and texture features are less effective than hypercolumns. Hypercolumns are critical to obtain good parts for ResNets, which explains our improvements over related work such as ODIN [19] and PICIE [8] which are based on clustering final-layer features. On Birds, we find that parts obtained via ground-truth keypoints and figure-ground masks also lead to a significantly better categorization performance, and PARTICLE approaches this oracle baseline. For ViTs we find that last layer "key" features of patches are effective and hypercolumns are not as critical, perhaps because resolution is maintained throughout the feature hierarchy. These differences are highlighted in Tab. 1, Tab. 2, and Fig. 2. Our approach is also relatively efficient as it takes only ≈2× the amount of time to train MoCo and is ≈5× faster than ODIN for ResNet50.

2. Related Work

Fine-grained Recognition using SSL. Cole *et al*. [10] show that self-supervised CNNs trained on ImageNet do not perform well on fine-grained domains compared to their supervised counterparts in the "low-data" regime. Prior work [10, 31, 32] has also investigated the role of domain shifts on the generalization concluding that high domain similarity is critical for good transfer. Our work aims to mitigate these issues by showing that the performance of ImageNet self-supervised representations can be improved by fine-tuning the representations using iterative part-discovery and contrastive learning on moderately sized datasets (≤ 10k images). Recent work in self-supervised learning using vision transformers (ViTs) such as DINO [4] show remarkable results for fine-grained classification. DINO performs as well as supervised ImageNet ViT models and much better than supervised ImageNet ResNet50 models [21]. Our experiments show that PARTICLE still offers improvements, especially on aircrafts where the domain shift is larger.

Part Discovery Methods. Our approach for part discovery is motivated by work that shows that hypercolumns extracted from generative [36, 41] or contrastively [7, 28] trained networks, as well as ViTs [1, 9] lead to excellent transfer on landmark discovery or part segmentation tasks. Among techniques for part discovery on fine-grained domains the most related ones include Sanchez *et al*. [30] who use a supervised keypoint detector to adapt to the target domain. Aygun *et al*. [2] boost landmark correspondence using an objective that captures finer distances in feature space. The focus of this line of work has been on part discovery, but our goal is to also evaluate how part discovery impacts fine-grained classification. Better techniques for part discovery are complementary to our approach.

Pixel Contrastive Learning. Several pixel-level SSL approaches have been proposed for image segmentation or object detection tasks. Our approach for part-centric learning is based on DetCon [18] which learns by clustering pixels based on color and texture [12]. They show improved detection and semantic segmentation performance compared to image-level SSL on standard benchmarks. We adopt the underlying objective due to its computational efficiency, but instead use pixel representations based on deep networks. ODIN [19] uses k-means clustering on the last-layer features of a discovery network to find object clusters to guide a contrastive objective of a separate representation network. The training is based on the student-teacher learning framework of BYOL [13]. Similarly, PiCIE [8] considers global clustering of pixel level features within a dataset and trains a network using photometric invariance and geometric equivariance on the segmentation task. Much of the focus of the above work has been on tasks on coarse domains (e.g., ImageNet or COCO), while our work considers fine-grained image classification and part segmentation tasks. Notably, we find that unlike hypercolumns, the last layer features of a ResNet often used to discover objects do not contain finer demarcations that constitute parts of objects in fine-grained domains (see Fig. 3 for some examples).

3. Method

Problem and Evaluation. We consider the problem of learning representations on fine-grained domains (e.g., Birds or Aircrafts) for image categorization and part segmentation tasks. We consider a setting where the dataset is moderately sized (e.g., $\leq$ 10,000 unlabeled images) and the goal is to adapt a SSL pre-trained representation trained on ImageNet. This represents a practical setting where one might have access to a large collection of unlabeled images from a generic domain and a smaller collection of domain-specific images. For evaluation we consider classification performance under a linear evaluation scheme (i.e., using multi-class logistic regression on frozen features), or part segmentation given a few ($\approx$ 100) labeled examples.

Approach. Given an initial network, our training procedure iterates between a part discovery step and a part-centric learning step outlined in Algorithm 1 and Fig. 1. In § 3.1 we outline various methods to obtain parts and compare them to baselines based on low-level features as well as keypoints and figure-ground masks when available. The latter serves as an oracle "upper bound" on the performance of the approach. In § 3.2 we present the part-level contrastive learning framework which discriminates features across parts within the same image under photometric and geometric transformations.

3.1. Part Discovery Methods

CNNs. Hypercolumn representations of CNNs have been widely used to extract parts of an object. A deep network of n layers (or blocks) can be written as $\Phi(\mathbf{x}) = \Phi^{(n)} \circ \Phi^{(n-1)} \circ \cdots \circ \Phi^{(1)}(\mathbf{x})$. A representation $\Phi(\mathbf{x})$ of size $H' \times W' \times K$ can be spatially interpolated to input size $H \times W \times K$ to produce a pixel representation $\Phi_I(\mathbf{x}) \in \mathbb{R}^{H \times W \times K}$. We use bilinear interpolation and normalize these features using a ℓ_2 norm. The hypercolumn representation of layers $l_1, l_2, \ldots, l_n$ is obtained by concatenating interpolated features from corresponding layers i.e.

$$\Phi_I(\mathbf{x}) = \|\Phi_I^{(l_1)}(\mathbf{x})\|_2 \oplus \|\Phi_I^{(l_2)}(\mathbf{x})\|_2 \oplus \cdots \oplus \|\Phi_I^{(l_n)}(\mathbf{x})\|_2$$

We then use k-means clustering of features within the *same image* to generate part segmentation. We choose the layers based on a visual inspection and keep it fixed across datasets. Further details are in § 5.1.

ViTs. Unlike CNNs, ViTs maintain constant spatial resolution throughout the feature hierarchy allowing one to obtain relatively high resolution pixel representations from the last layer. DINO [4] shows that the self-attention of the "`[cls]` token" has a strong figure-ground distinction. Last layer 'key' features of DINO have also been used to obtain part segmentations [1]. Motivated by this and our initial experiments that did not indicate better results using features across multiple layers, we consider the last layer 'key' features to extract pixel representations.

Algorithm 1 Part Discovery and Contrast Learning

Require: $D := \{\mathbf{X}\}$ ▷ Unlabeled images
Require: f, params={#iters, #clusters} ▷ Initial network, params

function PARTDISCOVERY(x, f)
  FREEZEWEIGHTS(f)
  h = NORMFEATURES($f(x)$) ▷ Forward pass as in § 3.1
  y = KMEANS(h, #clusters)
return y
end function

$f_1 \leftarrow f$ ▷ Initialize network
for $k \leftarrow 1$ to #iters **do**
  $\mathbf{Y} = \{\}$ ▷ Initialize labels
  for $x \in \mathbf{X}$ **do** ▷ On each example individually
    y = PARTDISCOVERY(x, f_k)
    $\mathbf{Y} \leftarrow append(y)$ ▷ Part labels
  end for
  $f_{k+1} \leftarrow$ PARTCONTRAST($\mathbf{X}$, $\mathbf{Y}$, f_k) ▷ Training § 3.2
end for

Baseline: Color and Texture. We extract parts using a classical image segmentation algorithm based on pixel color and texture – Felzenzwalb Huttenlocher [12]. The parameters used to generate segmentations are described in §4.

Baseline: Keypoints and Masks. As an oracle baseline we generate parts clustering based on keypoints or figure-ground masks. On birds dataset we assign each foreground pixel to the nearest keypoint (using a Voronoi tessellation) while all background pixels are assigned a background category. For Aircrafts, we consider the figure-ground mask as a binary segmentation (see Datasets, §4 for details).

Analysis. Fig. 2 visualizes the part clusters obtained using various techniques and pre-trained models. Hypercolumns extracted from pre-trained ResNet50 using DetCon produces slightly better visual results than from MoCo. Previous work, ODIN and PICIE cluster last-layer features which are rather coarse and not well aligned with object parts as shown in Fig. 3. This might explain the relatively weaker performance of ODIN on our benchmarks compared to our approach that uses hypercolumns (31.19 vs 34.31 on CUB classification fine-tuned over MoCo ImageNet - more in suppl.). Parts using color and texture are often not as effective, conflating foreground and background. The bottom row shows the clusters obtained using "side information", i.e., keypoints for birds and figure-ground for airplanes.

3.2. Part Contrastive Learning

Given an image $\mathbf{x}$ and an encoder f we a obtain a representation $\mathbf{y} = f(\mathbf{x})$ where $\mathbf{y} \in \mathbb{R}^{H \times W \times K}$ for CNNs and

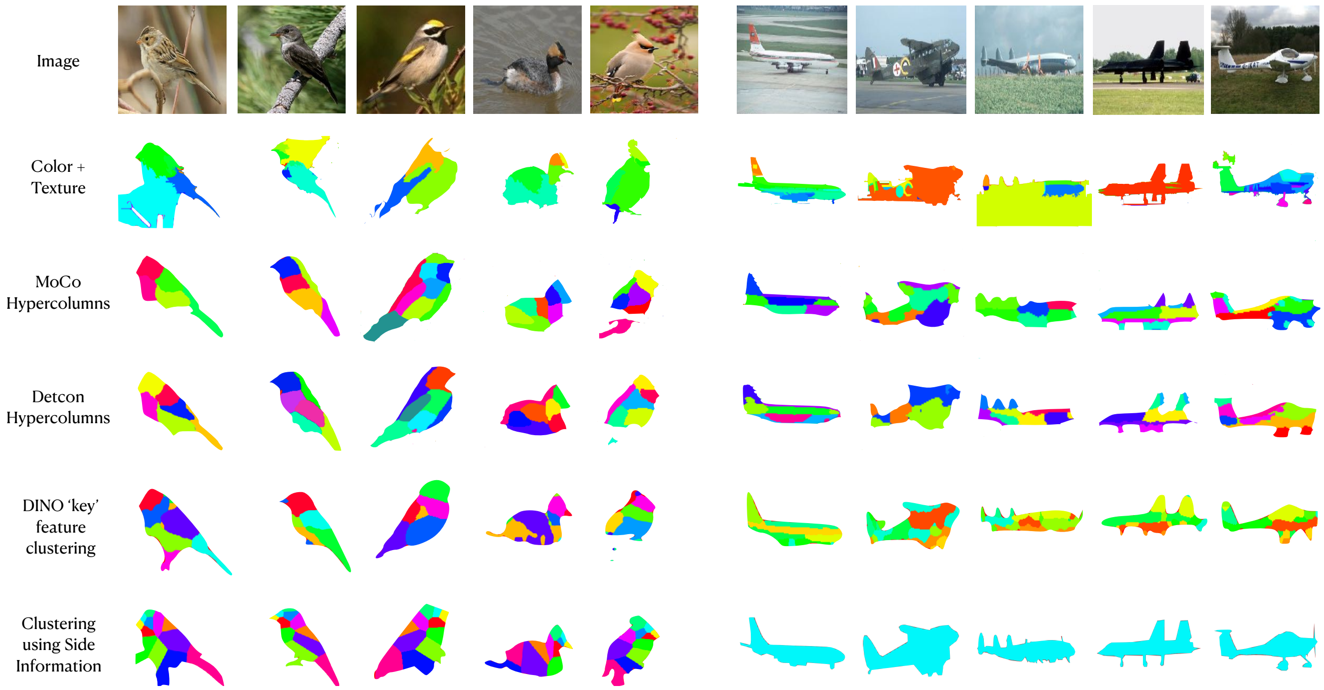

Figure 2. **Visualization of the parts obtained by clustering representations.** Clusters based on color and texture representations often conflate the object with the background. Clustering using hypercolumn features from ResNet50 trained using MoCo or DetCon are more aligned with semantic parts. For example, parts such as the head, tail, wing and breast in birds are distinct, and align with clusters generated using *ground truth* keypoints and figure-ground masks. DINO ViT representations are qualitatively similar. For Aircrafts, the only side information available is the figure-ground mask. *Note that for the purpose of this visualization we manually mask out the clusters in the background. Refer to Fig. 3 last column to see the background clusters.*

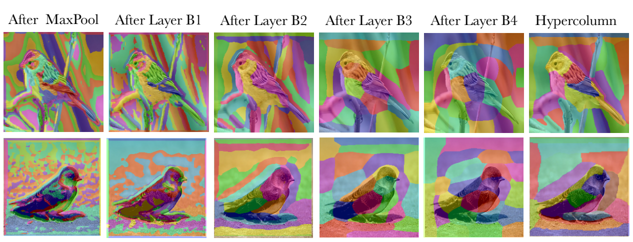

Figure 3. **Clusters features from various layers of a ResNet50.** The shallower layer (left) features are similar to those based on colour and texture. As we go deeper (from left to right), the parts are more distinctive (e.g., layer B2 and B3). Layer B4, the layer before the final average pooling, fails to produce meaningful clusters. Hypercolumns (last column) clusters often result in distinct parts. This ResNet50 was trained using DetCon on ImageNet.

$\mathbf{y} \in \mathbb{R}^{(P+1)\times K}$ for ViTs where (P+1) is the number of patches and the `[cls]` token. We consider the representation before the last Average Pooling layer in a ResNet50 network and the last layer output tokens only for the patches in case of ViT. Given the segmentation of the image $\mathbf{x}$ obtained in the previous step, we downsample it using nearest neighbour interpolation to get $\mathbf{s}$ so that we have a mask value m associated with each spatial location (i,j) in $\mathbf{y}$. A mask pooled feature vector for every mask value m can be obtained as:

$$\mathbf{y}_m = \frac{\sum_{i,j} \mathbb{1}(\mathbf{s}[i,j]=m) * \mathbf{y}[i,j]}{\sum_{i,j} \mathbb{1}(\mathbf{s}[i,j]=m)} \tag{1}$$

Given an image we generate two views $\mathbf{x}$ and $\mathbf{x}'$ using various augmentations (see supplementary). Next using Equation 1 we can obtain mask pooled features from both views as $\mathbf{y}_m, \mathbf{y}'_{m'}$ where m, m' are mask indices. Now using a projector MLP g and a predictor MLP q we get:

$$\mathbf{p}_m = q_\theta \circ g_\theta(\mathbf{y}_m) \qquad \mathbf{p}'_{m'} = g_\xi(\mathbf{y}'_{m'}) \tag{2}$$

Note that the second view $\mathbf{x}'$ is passed to a momentum encoder f_ξ, then the mask pooled features are fed to g_ξ. These networks are trained using momentum update whereas $q_\theta, g_\theta, f_\theta$ are trained using backpropagation. All the latents are rescaled so they have norm as $1/\sqrt{\tau}$ where $\tau = 0.1$

Next to contrast across masks we use the following loss function:

$$\mathcal{L} = \sum_m -\log \frac{\exp(\mathbf{p}_m \cdot \mathbf{p}'_m)}{\exp(\mathbf{p}_m \cdot \mathbf{p}'_m) + \sum_n \exp(\mathbf{p}_m \cdot \mathbf{p}'_n)} \tag{3}$$

where $\mathbf{p}'_n$ are the negatives *i.e*. samples from different masks from same image as well as across examples.

4. Datasets and Evaluation Metrics

Here we describe the datasets we use for the part aware contrastive training step and for the downstream tasks of fine-grained classification and few-shot part segmentation.

4.1. Birds

Self-Supervised Training. We use the Caltech-UCSD birds (CUB) [38] dataset that has 11788 images centered on birds with 5994 for training and 5794 for testing. We use the training set images for our contrastive learning part. The CUB dataset provides keypoints, figure-ground masks and classes as annotations. It has labels for 15 keypoints per-image. We remove the left/right distinctions and get a total of 12 keypoints : 'back', 'beak', 'belly', 'breast', 'crown', 'forehead', 'eye', 'leg', 'wing', 'nape', 'tail', 'throat'. Each foreground pixel is assigned a cluster based on the index of the nearest part, while background pixels are assigned their own labels. For clustering using color and texture, we use FH with the scale parameter of 400 and minimum component size of 1000 for this dataset, to get an average of 25 clusters per image. For hypercolumns we use k=25 for k-means clustering.

Classification. We again use the CUB dataset for classification. It has birds from 200 classes. We use the official train-test splits for our experiments and report the per-image accuracy on the test and validation sets.

Few-shot Part Segmentation. We use the PASCUB dataset for part segmentation with 10 part segments introduced by Saha *et al*. [29]. We use the training set consisting of 421 images to train and use the validation (74) and testing (75) sets of the CUB partition to present results. We report the mean intersection-over-union (IoU) on the validation and test sets.

4.2. Aircrafts

Self-Supervised Training. We use the OID Aircraft [37] dataset for pre-training. We use the official training split containing 3701 images. Since we do not have keypoint annotations for this dataset, we only use the figure-ground masks as the side information segmentations. For the color and texture we use FH with a scale parameter of 1000 and minimum component size of 1000 and get an average of 30 clusters per image. For clustering using hypercolumns we use k=25 for k-means clustering.

Classification. For classification we use the FGVC Aircraft [24] dataset. It contains 10,000 images belonging to 100 classes. We use the official 'trainval' set to train and the 'test' set for reporting testing results. They contain 6667 and 3333 images respectively. We report the mean per-image accuracy on this dataset.

Few-shot Part Segmentation. We use the Aircraft segmentation subset extracted from OID Aircraft in Saha *et al*. [29]. It contains 4 partially overlapping parts per image. We use the official 150 images for training and 75 each for validation and testing. We report the mean intersection-over-union (IoU) on this dataset.

5. Implementation Details and Baselines

5.1. ImageNet pre-trained SSL CNNs

We consider initialization using two choices of ImageNet self-supervised models both based on a ResNet50 architecture for a uniform comparison. One is based on MoCo and the other is based on DetCon. To obtain part clusters, every image in the dataset is resized to 224×224 and hypercolumn features are extracted from the first MaxPool, BottleNeck Block 1, BottleNeck Block 2 and BottleNeck Block 3 layers. We resample all features to a spatial resolution of 64×64 and concatenate across channel dimension. This results in a $64\times64\times1856$ feature vector. We use sklearn k-means clustering using k=25 and 500 max iterations. We provide an ablation to justify the number of clusters in supplementary. We cluster each image in the dataset independently. We use the same specifications for hypercolumn extraction and clustering while training iterations of discovery and contrast.

5.2. ImageNet pre-trained DINO ViT

We also extend our method to vision transformers. We extract parts from ImageNet pre-trained DINO ViT by clustering the last layer (Layer 11) 'key' features using the method by Amir *et al*. [1]. We fix the number of parts to 7 for birds and 5 for aircrafts. We use the 8×8 patch version of ViT S/8 as it has the largest feature resolution for parts. For fine-tuning DINO ViT using PARTICLE, we apply the part contrastive loss over the output patch tokens of the ViT and add to the DINO student-teacher loss with equal weights. We use 224×224 input image resulting in 28×28 feature vector at every layer.

5.3. Baselines for Self-Supervised Adaptation

To determine the effect of our training strategy over the boost coming from simply fine-tuning on a category specific dataset, we benchmark over some standard baselines. For each of these baselines we fine-tune over the category specific dataset (CUB for birds/OID for aircrafts) while learning using their objective. Below we list the baselines:

MoCo (V2). The Momentum Contrast (MoCo [16]) approach minimizes a InfoNCE loss [25] over a set of unlabeled images. MoCo performs instance level contrast by maintaining a queue of other examples considered negatives and treating transformations of a single image as positives.

DetCon. DetCon uses color and texture features to generate object segmentations using the Felzenszwalb-Huttenlocher [12] algorithm. It uses a ResNet-50 based model to train using pixel contrast based on these object segmentations. Their loss function is the same as in § 3.1.

ODIN. This method has the same training objective of DetCon but creates segmentations by clustering the last layer features of a 'discovery' network using K-means in every iteration. This 'discovery' network is initialized randomly and is trained using momentum update from the main encoder. In Fig. 3 we show that the clusters of the last layer features of even a pre-trained network is not a good representation of object parts. We show a comparison of using ODIN vs other objectives in the Supplementary Material.

DINO ViT. We use the ViT S/8 network which the Small ViT using 8×8 patches, trained with DINO [4]. DINO trains using a student teacher framework where the student is updated by minimizing the cross-entropy between softmax normalized outputs of the student and teacher. The teacher is updated using momentum. DINO is also an instance level contrastive method.

PiCIE. PiCIE [8] learns unsupervised object segmentation by clustering the features of the complete dataset using mini-batch k-means and training using invariance to photometric transformations and equivariance to geometric transformations. For part segmentation, PiCIE does not work well (see supplementary) because it uses only the last downsampled feature space of the encoder which does not have part information (see Fig. 3) and trying to fit object parts from all images to a single set of centroids for the whole dataset results in loss of information.

5.4. Hyper-parameters

Self-Supervised Adaptation. For all baselines and our method based on CNN we finetune the initialized model for 600 epochs with a learning rate of 0.005 with a batch size of 320. We use a SGD optimizer with weight decay of 1.5E-6 and momentum of 0.9. We use a cosine learning rate decay with 10 epochs for warm up. For momentum updates we use a decay of 0.996. For all methods, we train using an image resolution of 224×224. We utilize the augmentations as defined in BYOL [13]. We provide the details in the Supplementary. For adaptation to DINO ViT, we use a learning rate of 1E-7 with cosine decay and a weight decay of 0.4. We train for 100 epochs with a batch size of 64.

Iterative Training. For extracting hypercolumns, we use the same specification as in § 5.1. We train for 20 epochs with a learning rate of 0.05. Rest of the hyperparameters stay the same as in the previous paragraph. For DINO ViT based models, we use a LR of 1E-8 and train for 60 epochs.

Linear Probing. We initialize a ResNet50 encoder with the contrastively trained networks as described above and § 3. We do the evaluation using the input image of resolution 224×224. We store the features before the last Average pooling layer for both train and test sets. We do not use any data augmentation for this. We then use the Logistic Regression method of sklearn, which we train using L-BFGS for 1000 maximum iterations. We choose the best model by evaluating on the validation set. For DINO ViT based models we average over the class token and patch tokens and use the same details as above.

Fine-Tuning. We also report results using fine-tuning in the supplementary where the entire network is trained for 200 epochs with a batch size of 200. We use SGD with a lr of 0.01 and momentum of 0.9. We train for varying number of images in the train set – 1, 3, 8, 15, 30 per class. Only flipping augmentation is used while training, except the low shot versions (1,3 and 8) where we also add random resized cropping and color jitter. For reporting scores on test set, we choose the best checkpoint based on the val set.

Part Segmentation. We add a decoder network consisting of four upsampling layers followed by convolutions to generate part segmentations from the ResNet50 features. We use the best pre-training checkpoint for each experiment obtained in linear probing on validation set. We follow all the parameters for training/evaluation of Saha *et al*. [29]. We fine-tune the entire network for part segmentation. Here we train and test using input images of resolution 256×256 following. We train the network using a cross entropy loss for PASCUB experiments. For Aircrafts, we treat it as a pixel-wise multi-label classification task and use binary cross entropy (BCE) loss. We use Adam optimizer with a learning rate of 0.0001 for 200 epochs. We use flipping and color-jitter augmentations while training. We use the mean IoU metric to report results. During evaluation, we perform 5 fold cross validation to find the best checkpoint using the validation sets and report the mean of them. For DINO ViT based models we rearrange the patch 'key' features of the last layer back to a 3D tensor and use 3 layers of upsampling each of which consists of two 3×3 kernel Convs. We use a learning rate of 1E-5. Other details are same as above.

6. Results

We describe the results of evaluating the baselines and our method across different settings for fine-grained visual classification and few-shot part segmentation. In the following sections, we present a detailed analysis of various factors that affect the performance of baselines and our model.

6.1. PARTICLE Improves Performance Consistently

Tab. 1 shows that our method improves performance across baselines. For each model, we compare PARTI-

Architecture	Method	Caltech-UCSD Birds Cls	Caltech-UCSD Birds Seg	FGVC Aircrafts Cls	OID Aircrafts Seg
	Supervised ImageNet	66.29	47.41 ± 0.88	46.46	54.39 ± 0.52
ResNet50	MoCoV2 (ImageNet)	28.92	46.08 ± 0.55	19.62	51.57 ± 0.98
	MoCoV2 *fine-tuned*	31.17	46.22 ± 0.70	23.99	52.65 ± 0.54
	PARTICLE *fine-tuned*	**36.09**	**47.40 ± 1.06**	**29.13**	**54.74 ± 0.47**
	DetCon (ImageNet)	35.39	47.42 ± 0.92	35.55	53.62 ± 0.67
	DetCon *fine-tuned*	37.15	47.88 ± 1.18	40.74	56.26 ± 0.25
	PARTICLE *fine-tuned*	**41.98**	**50.21 ± 0.85**	**44.13**	**58.99 ± 0.61**
ViT S/8	DINO (ImageNet)	83.36	49.57 ± 1.26	72.37	61.73 ± 0.88
	DINO *fine-tuned*	83.36	49.66 ± 0.98	72.37	61.68 ± 0.71
	PARTICLE *fine-tuned*	**84.15**	**51.40** ± **1.29**	**73.64**	**62.71** ± **0.56**

Table 1. **Performance on downstream tasks.** We present the performance boost that our approach offers over various pre-trained SSL methods with backbone architecture as ResNet-50 or ViT S8. We show results for Birds and Aircrafts datasets. We significantly boost classification accuracy for CNN based models. While DINO is already much better than CNN based models for fine-grained classification, we are still able to improve the performance using our method. The gap in segmentation performance for DINO ViT vs DetCon/MoCo V2 is much less pronounced. Our method contributes steady improvement over all baseline models for segmentation.

Method	CUB Cls	CUB Seg	FGVC Cls	OID Seg
Color+Texture	37.15	47.88	40.74	56.26
Hypercolumns	40.88	49.23	43.99	58.95
Side Information	43.72	50.15	39.03	55.98

Table 2. **Effect of part discovery method.** We compare the performance of one iteration of PARTICLE over the ResNet50 model trained using DetCon. Hyercolumns lead to improved results compared to color and texture, and nearly match the performance obtained by clustering keypoints + figure-ground masks on birds. On airplanes, side information beyond figure-ground is lacking, and PARTICLE performs better.

Method		Iter 0	Iter 1	Iter 2	Iter 3
MoCo	Cls.	28.92	34.31	36.03	**36.09**
	Seg.	46.08	46.39	47.38	**47.40**
DetCon	Cls.	35.39	40.88	**42.00**	41.98
	Seg.	47.42	49.23	50.17	**50.21**

Table 3. **Effect of number of iterations.** We present the performance on CUB dataset over PARTICLE iterations. Iter 0 refers to the performance of the initial model (either MoCo or DetCon). The largest boost is observed in the first iteration, while the performance often saturates after two iterations.

CLE to the ImageNet pre-trained SSL model, and when the model is fine-tuned on the dataset using the objective of the underlying SSL model. We report the results of the best iteration to compare the maximum boost that PARTICLE can contribute. However, most of the improvement is obtained after a single iteration (Tab 3). ResNet50 SSL models lag behind supervised ImageNet models for classification tasks. PARTICLE fine-tuning goes a long way toward bridging this gap. DINO ViT on the other hand performs exceptionally well on fine-grained classification, even outperforming the ImageNet supervised CNNs. Yet, PARTICLE offers consistent improvements. For few-shot part segmentation, PARTICLE offers significant improvement over all baseline SSL models. We present results on an additional domain of Cars in the supplementary.

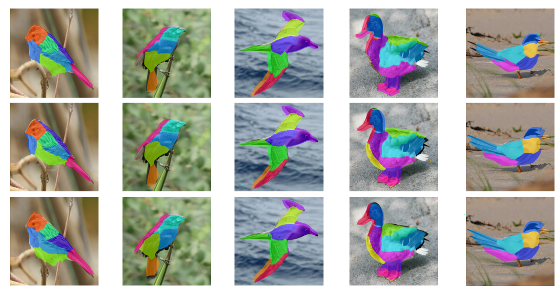

Figure 4. **Effect of Iterative training on clustering.** For the first bird as an example, the first iteration captures the boundary of the wing, head and belly better. The second iteration introduces a new middle part.

Performance of DINO. ImageNet pre-trained DINO is exceptionally good in fine-grained classification. It performs better than ImageNet pre-trained DetCon in classification tasks, however the difference is not as large for the part segmentation tasks. We believe that this can be attributed to DINO's strong figure-ground decomposition and the structure of it's feature space that makes it effective for linear and nearest-neighbor classification [4, 21].

6.2. Effect of Clustering Method

As we described earlier, Fig. 2 shows a qualitative comparison of clusters obtained using various representations described in § 3.1. Tab 2 shows the quantitative performance of various clustering methods on classification and segmentation tasks. Hypercolumn features from ImageNet pre-trained DetCon beats the performance of color + texture features. However, it lags behind the side information oracle in the case of birds, since the weak supervision of keypoints and figure-ground mask results in better part discovery. This indicates that better part discovery methods

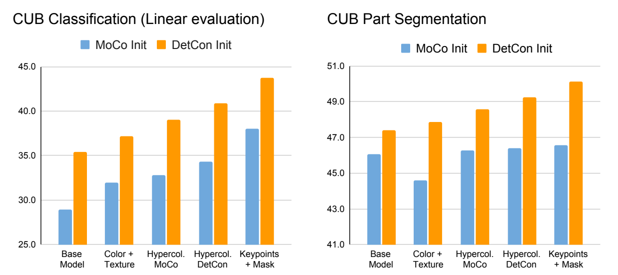

Figure 5. **Effect of initialization and adaption.** The left panel shows the classification performance (Linear evaluation) while the right panel shows the part segmentation performance on the CUB dataset. In each panel we show the result of initializing the representation network using MoCo and DetCon, and various ways to obtain part segmentation via clustering.

could lead to improvements in classification tasks.

6.3. Effect of Iterative Training

We vary the number of outer iterations on our model from zero, i.e., the initialization, to three, which consists of three iterations of part discovery and representation learning over the entire dataset. Results are shown in Tab. 3. For both initializations we did not find significant improvements beyond the second iteration on Birds. On Aircrafts the improvements over iterations were smaller (also see Table 1, $1\times$ vs. $3\times$). Fig. 4 shows how the clustering changes over iterations. To produce consistent clusters across images, *i.e.*, to avoid the randomness of k-means, we initialize the successive clustering for k-means using the previous partition and continue k-means for 500 iterations.

6.4. Effect of Initialization

Fig. 5 compares the effect of initializing weights with either MoCo V2 or DetCon ImageNet pre-trained weights. We compare performance on both classification and segmentation for various clustering techniques. The initial DetCon model has a higher performance than MoCo on both tasks. The boost observed follows the same trend for both initialization strategies. For Part Segmentation again the base DetCon ImageNet performs better than MoCo, however the trend of the boost over base model is not same for both initializations. Starting with a MoCo initialization the fine-tuned models do not see an adequate boost, whereas in the case of DetCon initialization the fine-tuned models see significant boost over the base DetCon model.

6.5. Comparison to ImageNet supervised CNNs

Tab. 1 shows that our ResNet50 based methods improve over ImageNet supervised models for few-shot part segmentation on both Birds and Aircrafts datasets. The ImageNet pre-trained SSL baselines are close to ImageNet supervised in the case of Birds and slightly worse on Aircrafts. However, using our methods leads to a significant boost over the pre-trained SSL methods. This once again suggests that the current CNN based SSL approaches are quite effective at learning parts, but are limited in their ability to recognize categories. The aircrafts dataset has a larger domain gap from the ImageNet dataset and our CNN based methods achieve closer performance to ImageNet supervised ResNet50 models. Our linear evaluation score reaches close to ImageNet supervised for Aircrafts ($\sim$2 points gap) unlike for Birds where there is still a gap of about $\sim$24 points. ImageNet already has a large number of classes of birds and has been trained for classification, which gives it a large advantage on a fine-grained bird classification dataset. The improvement in part segmentation of our method over ImageNet supervised ResNet-50 remains similar for both Birds and Aircrafts.

6.6. Efficiency of Various Methods

CNNs. Training MoCo is fastest since it performs image level contrast. Both DetCon and our method (one iteration) take the same amount of time which is less than $2\times$ that of MoCo. Note that we train each baseline and our method for 600 epochs. Since we use relatively small datasets to train, our approach takes less than 11 hours on 8 2080TI GPUs for the first iteration. We train the next iterations only for 20 epochs which takes around 20 minutes on the same GPU setup (total of 40 minutes for 2 extra iterations).
ViTs. For the first iteration, we train for 100 epochs which takes less than 2 hours on 8 2080TI GPUs. For the next iteration we train for 60 epochs which takes about an hour in the same setting.

7. Conclusion

We show that clustering and contrasting parts obtained through ImageNet self-supervised networks is an effective way to adapt them on small to moderately sized fine-grained datasets without any supervision. While we observe significant improvements on part segmentation tasks, even outperforming supervised ImageNet ResNets, we also show consistent improvements over the significantly better ViT models. On the Airplanes dataset where the domain gap over ImageNet is larger, our approach leads to larger gains. The analysis shows that current self-supervised models (including our own) are very effective at learning pose and parts. Moreover, conditioning and contrasting the discovered parts allows the model to learn diverse localized representations allowing better generalization to the classification tasks. However, a big limitation of the approach is that it requires a good initial model to discover parts, and the approach may not generalize to significantly different domains. Future work will explore if parts extracted from generic large-scale models lead to better guidance for part and feature learning, and will aim to characterize the effect of domain shifts on the effectiveness of transfer. We will also publicly release pre-trained models and codebase to reproduce the results upon acceptance.

References

[1] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *ECCVW What is Motion For?*, 2022. 2, 3, 5

[2] Mehmet Aygün and Oisin Mac Aodha. Demystifying unsupervised semantic correspondence estimation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXX*, pages 125–142. Springer, 2022. 2

[3] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014. 1

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2, 3, 6, 7

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

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

[7] Zezhou Cheng, Jong-Chyi Su, and Subhransu Maji. On equivariant and invariant learning of object landmark representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9897–9906, 2021. 1, 2

[8] Jang Hyun Cho, Utkarsh Mall, Kavita Bala, and Bharath Hariharan. Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16794–16804, 2021. 2, 6, 11

[9] Subhabrata Choudhury, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Unsupervised part discovery from contrastive reconstruction. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 28104–28118. Curran Associates, Inc., 2021. 2

[10] Elijah Cole, Xuan Yang, Kimberly Wilber, Oisin Mac Aodha, and Serge Belongie. When does contrastive visual representation learning work? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14755–14764, 2022. 1, 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[12] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004. 2, 3, 6

[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. *Advances in neural information processing systems*, 33:21271–21284, 2020. 1, 2, 6

[14] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015. 1

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

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 2, 5

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[18] Olivier J Hénaff, Skanda Koppula, Jean-Baptiste Alayrac, Aaron Van den Oord, Oriol Vinyals, and João Carreira. Efficient visual pretraining with contrastive detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10086–10096, 2021. 1, 2

[19] Olivier J Hénaff, Skanda Koppula, Evan Shelhamer, Daniel Zoran, Andrew Jaegle, Andrew Zisserman, João Carreira, and Relja Arandjelović. Object discovery and representation networks. *arXiv preprint arXiv:2203.08777*, 2022. 2

[20] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019. 1

[21] Menglin Jia, Bor-Chun Chen, Zuxuan Wu, Claire Cardie, Serge Belongie, and Ser-Nam Lim. Rethinking nearest neighbors for visual classification. *arXiv preprint arXiv:2112.08459*, 2021. 2, 7

[22] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 14

[23] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015. 1

[24] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 2, 5

[25] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 5

[26] Kitsuchart Pasupa, Phongsathorn Kittiworapanya, Napasin Hongngern, and Kuntpong Woraratpanya. Evaluation of deep learning algorithms for semantic segmentation of car parts. *Complex & Intelligent Systems*, pages 1–13, May 2021. 14

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 2

[28] Oindrila Saha, Zezhou Cheng, and Subhransu Maji. Ganorcon: Are generative models useful for few-shot segmentation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9991–10000, 2022. 2

[29] Oindrila Saha, Zezhou Cheng, and Subhransu Maji. Improving few-shot part segmentation using coarse supervision. In *Computer Vision – ECCV 2022*, pages 283–299, Cham, 2022. Springer Nature Switzerland. 5, 6

[30] Enrique Sanchez and Georgios Tzimiropoulos. Object landmark discovery through unsupervised adaptation. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[31] Jong-Chyi Su, Zezhou Cheng, and Subhransu Maji. A realistic evaluation of semi-supervised learning for fine-grained classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12966–12975, 2021. 2

[32] Jong-Chyi Su, Subhransu Maji, and Bharath Hariharan. When does self-supervision improve few-shot learning? In *European conference on computer vision*, pages 645–666. Springer, 2020. 1, 2

[33] Luming Tang, Davis Wertheimer, and Bharath Hariharan. Revisiting pose-normalization for fine-grained few-shot recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[34] James Thewlis, Samuel Albanie, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of landmarks by descriptor vector exchange. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6361–6371, 2019. 1

[35] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers amp; distillation through attention. In *International Conference on Machine Learning*, volume 139, pages 10347–10357, July 2021. 13

[36] Nontawat Tritrong, Pitchaporn Rewatbowornwong, and Supasorn Suwajanakorn. Repurposing gans for one-shot semantic part segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4475–4485, 2021. 2

[37] A. Vedaldi, S. Mahendran, S. Tsogkas, S. Maji, B. Girshick, J. Kannala, E. Rahtu, I. Kokkinos, M. B. Blaschko, D. Weiss, B. Taskar, K. Simonyan, N. Saphra, and S. Mohamed. Understanding objects in detail with fine-grained attributes. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 5

[38] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2, 5

[39] Xiu-Shen Wei, Yi-Zhe Song, Oisin Mac Aodha, Jianxin Wu, Yuxin Peng, Jinhui Tang, Jian Yang, and Serge Belongie. Fine-grained image analysis with deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1

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

[41] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10145–10155, 2021. 2