SELF-LABELLING VIA SIMULTANEOUS CLUSTERING AND REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Combining clustering and representation learning is one of the most promising approaches for unsupervised learning of deep neural networks. However, doing so naively leads to ill posed learning problems with degenerate solutions. In this paper, we propose a novel and principled learning formulation that addresses these issues. The method is obtained by maximizing the information between labels and input data indices. We show that this criterion extends standard cross-entropy minimization to an optimal transport problem, which we solve efficiently for millions of input images and thousands of labels using a fast variant of the Sinkhorn-Knopp algorithm. The resulting method is able to self-label visual data so as to train highly competitive image representations without manual labels. Compared to the best previous method in this class, namely DeepCluster, our formulation minimizes a single objective function for both representation learning and clustering; it also significantly outperforms DeepCluster in standard benchmarks.

1 INTRODUCTION

Learning from unlabelled data can dramatically reduce the cost of deploying algorithms to new applications, thus amplifying the impact of machine learning in the real world. Self-supervision is an increasingly popular framework for learning without labels. The idea is to define pretext learning tasks can be constructed from raw data alone, but that still result in neural networks that transfer well to useful applications.

Much of the research in self-supervision has focused on designing new pre-text tasks. However, given supervised data such as ImageNet (Deng et al., 2009), the standard classification objective of minimizing the cross-entropy loss still results in better pre-training than any of such methods (for a certain amount of data and model complexity). This suggests that the task of classification may be sufficient for pre-training networks, provided that suitable data labels are available. In this paper, we thus focus on the problem of obtaining the labels automatically by designing a self-labelling algorithm.

Learning a deep neural network together while discovering the data labels can be viewed as *simultaneous clustering and representation learning*. The latter can be approached by combining cross-entropy minimization with an off-the-shelf clustering algorithm such as K-means. This is precisely the approach adopted by the recent DeepCluster method (Caron et al., 2018), which achieves excellent results in unsupervised representation learning. However, combining representation learning, which is a discriminative task, with clustering, which is usually based on a generative objective (Bishop, 2006), is not at all trivial. In particular, we show that the combination of cross-entropy minimization and K-means as adopted by DeepCluster cannot be described as the consistent optimization of an overall learning objective; instead, there exist degenerate solutions that the algorithm avoids via particular implementation choices.

In order to address this technical shortcoming, in this paper we contribute a new principled formulation for simultaneous clustering and representation learning. The starting point is to minimize the cross-entropy loss for learning the deep network *as well as* the data labels. This is often done in semi-supervised learning and multiple instance learning. However, when applied naively to the unsupervised case, it immediately leads to a degenerate solution where all data points are mapped to the same cluster.

We solve this issue by adding the constraint that the labels must induce an equipartition of the data, which we show to maximize the information between data indices and labels. We also show that the resulting label assignment problem is the same as optimal transport, and can therefore be solved in polynomial time as a linear program. However, since we want to scale the algorithm to millions of data points and thousands of labels, standard transport solvers are inadequate. Thus, we also propose to use a fast version of the Sinkhorn-Knopp algorithm for finding an approximate solution to the transport problem efficiently at scale, using fast matrix-vector algebra.

Compared to methods such as DeepCluster, the new formulation is more principled and allows to more easily demonstrate properties such as convergence. Most importantly, via extensive experimentation we show that our new approach leads to significantly superior results, achieving the new state-of-the-art for representation learning for clustering-based approaches.

2 Related Work

Our paper relates to two broad areas of research: (a) self-supervised representation learning, and (b) more specifically, training a deep neural network using pseudo-labels, i.e. the assignment of a label to each image. We discuss closely related works for each.

Self-supervised learning: A wide variety of methods that do not require manual annotations have been proposed for the self-training of deep convolutional neural networks. These methods use various cues and proxy tasks namely, in-painting (Pathak et al., 2016), patch context and jigsaw puzzles (Doersch et al., 2015; Noroozi & Favaro, 2016; Noroozi et al., 2018; Mundhenk et al., 2017), clustering (Caron et al., 2018; Jiabo Huang & Zhu, 2019), noise-as-targets (Bojanowski & Joulin, 2017), colorization (Zhang et al., 2016; Larsson et al., 2017), generation (Jenni & Favaro, 2018; Ren & Lee, 2018; Donahue et al., 2017; Donahue & Simonyan, 2019), predictive coding (Oord et al., 2018; Hénaff et al., 2019), geometry (Dosovitskiy et al., 2016), predicting transformations (Gidaris et al., 2018; Zhang et al., 2019) and counting (Noroozi et al., 2017). In (Feng et al., 2019), predicting rotation (Gidaris et al., 2018) is combined with instance retrieval (Wu et al., 2018).

The idea is that the pretext task can be constructed automatically and easily from images alone. Thus, methods often modify information in the images and require the network to recover them. In-painting or colorization techniques fall in this category. However these methods have the downside that the features are learned on modified images which potentially harms the generalization to unmodified ones. For example, colorization uses a gray scale image as input, thus the network cannot learn to extract color information, which can be important for other tasks.

Slightly less related are methods that use additional information to learn features. Here, often temporal information is used in the form of videos. Typical pretext tasks are based on temporal-context (Misra et al., 2016; Wei et al., 2018; Lee et al., 2017; Sermanet et al., 2018), spatio-temporal cues (Isola et al., 2015; Gao et al., 2016; Wang et al., 2017), foreground-background separation via video segmentation (Pathak et al., 2017), optical-flow (Gan et al., 2018; Mahendran et al., 2018), future-frame synthesis (Srivastava et al., 2015), audio prediction from video (de Sa, 1994; Owens et al., 2016), audio-video alignment (Arandjelović & Zisserman, 2017), ego-motion estimation (Jayaraman & Grauman, 2015), slow feature analysis with higher order temporal coherence (Jayaraman & Grauman, 2016), transformation between frames (Agrawal et al., 2015) and patch tracking in videos (Wang & Gupta, 2015).

Pseudo-labels for images: In the self-supervised domain, we find a spectrum of methods that either give each data point a unique label (Wu et al., 2018; Dosovitskiy et al., 2016) or train on a flexible number of labels with k-means (Caron et al., 2018), with mutual information (Ji et al., 2018) or with noise (Bojanowski & Joulin, 2017). In (Noroozi et al., 2018) a large network is trained with a pretext task and a smaller network is trained via knowledge transfer of the clustered data. Finally, (Bach & Harchaoui, 2008; Vo et al., 2019) use convex relaxations to regularized affine-transformation invariant linear clustering, that does not scale to larger datasets.

Our contribution is a simple method that combines a novel pseudolabel extraction procedure from raw data alone and the training of a deep neural network using a standard cross-entropy loss.

3 METHOD

We will first derive our self-labelling method, then interpret the method as optimizing labels and targets of a cross-entropy loss and finally analyze similarities and differences with other clustering based methods.

3.1 SELF-LABELLING

Neural network pre-training is often achieved via a supervised data classification task. Formally, consider a deep neural network $\boldsymbol{x} = \Phi(I)$ mapping data I (e.g. images) to feature vectors $\boldsymbol{x} \in \mathbb{R}^D$. The model is trained using a dataset (e.g. ImageNet) of N data points $I_1, \ldots, I_N$ with corresponding labels $y_1, \ldots, y_N \in \{1, \ldots, K\}$, drawn from a space of K possible labels. The model is followed by a *classification head* $h : \mathbb{R}^D \to \mathbb{R}^K$, usually consisting of a single linear layer, converting the feature vector into a vector of class scores. The class scores are mapped to class probabilities via the softmax operator:

$$p(y = \cdot|\boldsymbol{x}_i) = \operatorname{softmax}(h \circ \Phi(\boldsymbol{x}_i)).$$

The model and head parameters are learned by minimizing the average cross-entropy loss

$$E(p|y_1, \ldots, y_N) = -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i). \tag{1}$$

Training with objective (1) requires a labelled dataset. When labels are unavailable, we require a *self-labelling* mechanism to assign the labels automatically.

In semi-supervised learning, self-labelling is often achieved by jointly optimizing (1) with respect to the model $h \circ \Phi$ and the labels $y_1, \ldots, y_N$. This can work if at least part of the labels is known, thus constraining the optimization. However, in the fully unsupervised case, it leads to a degenerate solution: eq. (1) is trivially minimized by assigning all data points to a single (arbitrary) label.

To address this issue, we first rewrite eq. (1) by encoding the labels as posterior distributions $q(y|\boldsymbol{x}_i)$:

$$E(p, q) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{y=1}^{K} q(y|\boldsymbol{x}_i) \log p(y|\boldsymbol{x}_i). \tag{2}$$

If we set the posterior distributions $q(y|\boldsymbol{x}_i) = \delta(y - y_i)$ to be deterministic, the formulations in eqs. (1) and (2) are equivalent, in the sense that $E(p, q) = E(p|y_1, \ldots, y_N)$. In this case, optimizing q is the same as reassigning the labels, which leads to the degeneracy. To avoid this, we add the constraint that the label assignments must partition the data in equally-sized subsets. Formally, the learning objective objective[1] is thus:

$$\min_{p,q} E(p, q) \quad \text{subject to} \quad \forall y : q(y|\boldsymbol{x}_i) \in \{0, 1\} \text{ and } \sum_{i=1}^{N} q(y|\boldsymbol{x}_i) = \frac{N}{K}. \tag{3}$$

The constraints mean that each data point $\boldsymbol{x}_i$ is assigned to exactly one label and that, overall, the N data points are split uniformly among the K classes.

The objective in eq. (3) is combinatorial in q and thus may appear very difficult to optimize. However, this is an instance of the *optimal transport problem*, which can be solved relatively efficiently. In order to see this more clearly, let $P_{yi} = \log p(y|\boldsymbol{x}_i) \frac{1}{N}$ be the $K \times N$ matrix of scaled log-posterior probabilities estimated by the model. Likewise, let $Q_{yi} = q(y|\boldsymbol{x}_i) \frac{1}{N}$ be the scaled $K \times N$ matrix of label assignments. Using the notation of (Cuturi, 2013), we relax matrix Q to be an element of the *transportation polytope*

$$U(r, c) := \{Q \in \mathbb{R}_+^{K \times N} \mid Q\mathbb{1} = r,\ Q^\top \mathbb{1} = c\}. \tag{4}$$

Here $\mathbb{1}$ are vectors of all ones of the appropriate dimensions, so that r and c are the marginal projections of matrix Q onto its rows and columns, respectively. In our case, we require Q to be a matrix of conditional probability distributions that split the data uniformly, which is captured by:

$$r = \frac{1}{K} \cdot \mathbb{1}, \quad c = \mathbb{1}.$$

[1]We assume for simplicity that K divides N exactly, but the formulation is easily extended to any $N \geq K$ by setting the constraints to either $\lfloor N/K \rfloor$ or $\lfloor N/K \rfloor + 1$, in order to assure that there is a feasible solution.

With this notation, we can rewrite the objective function in eq. (3), up to a constant shift, as

$$E(p, q) = \langle Q, P \rangle + \log K, \tag{5}$$

where $\langle \cdot \rangle$ is the Frobenius dot-product between two matrices. Hence optimizing eq. (3) with respect to the assignments Q is equivalent to solving the problem:

$$\min_{Q \in U(r,c)} \langle Q, P \rangle. \tag{6}$$

This is a *linear program*, and can thus be solved in polynomial time. Furthermore, solving this problem always leads to an integral solution despite having relaxed Q to the continuous polytope $U(r, c)$, guaranteeing the exact equivalence to the original problem.

In practice, however, the resulting linear program is large, involving millions of data points and thousands of classes. Traditional algorithms to solve the transport problem scale badly to instances of this size. We address this issue by adopting a fast version (Cuturi, 2013) of the *Sinkhorn-Knopp algorithm*. This amounts to introducing a regularization term

$$\min_{Q \in U(r,c)} \langle Q, P \rangle - \frac{1}{\lambda} \mathrm{KL}(Q \| rc^\top), \tag{7}$$

where KL is the Kullback-Leibler divergence and $rc^\top$ can be interpreted as a $K \times N$ probability matrix. The advantage of this regularization term is that the minimizer of eq. (7) can be written as:

$$Q = \mathrm{diag}(\alpha) e^{-\lambda P} \mathrm{diag}(\beta) \tag{8}$$

where exponentiation is meant element-wise and α and β are two vectors of scaling coefficients chosen so that the resulting matrix Q is also a probability matrix (see (Cuturi, 2013) for a derivation). The vectors α and β can be obtained, as shown below, via a simple matrix scaling iteration.

For very large λ, optimizing eq. (7) is of course equivalent to optimizing eq. (6), but even for moderate values of λ the two objectives tend to have approximately the same optimizer (Cuturi, 2013). Choosing λ trades off convergence speed with closeness to the original transport problem. In our case, using a fixed λ is appropriate as we are ultimately interested in the final clustering and representation learning results, rather than in solving the transport problem exactly.

Our final algorithm's core can be described as follows. We learn a model $h \circ \Phi$ and a label assignment matrix Q by solving the optimization problem eq. (6) with respect to both Q, which is a probability matrix, and the model $h \circ \Phi$, which determines the predictions $P_{yi} = \log(\mathrm{softmax}_y(h \circ \Phi(\boldsymbol{x}_i)))$. We do so by alternating the two steps:

Step 1: representation learning. Given the current label assignment Q, the model is updated by minimizing eq. (6) with respect to (the parameters of) $h \circ \Phi$. This is the same as training the model using the common cross-entropy loss for classification.

Step 2: self-labelling. Given the current model $h \circ \Phi$, we compute the log probabilities P. Then, we find Q using eq. (8) by iterating the updates (Cuturi, 2013)

$$\forall y : \alpha_y \leftarrow [e^{-\lambda P} \beta]_y^{-1} \qquad \forall i : \beta_i \leftarrow [\alpha^\top e^{-\lambda P}]_i^{-1}.$$

Each update involves a single matrix-vector multiplication, so it is relatively quick even for millions of data points and thousands of labels. Also note that the parameters α and β can be retained between steps, thus allowing a warm start of Step 2.

3.2 Interpretation

As shown above, the formulation in eq. (2) uses scaled versions of the probabilities. We can interpret these by treating the data *index* i as a random variable with uniform distribution $p(i) = 1/N$ and by rewriting the posteriors $p(y|\boldsymbol{x}_i) = p(y|i)$ and $q(y|\boldsymbol{x}_i) = q(y|i)$ as conditional distributions with respect to the data index i instead of the feature vector $\boldsymbol{x}_i$. With these changes, we can rewrite eq. (5) as

$$E(p, q) = -\sum_{i=1}^{N} \sum_{y=1}^{K} p(y, i) \log q(y, i) = H(p, q), \tag{9}$$

which is the cross-entropy between the joint label-index distributions $p(y, i)$ and $q(y, i)$. The minimum of this quantity w.r.t. q is obtained when $q = p$, in which case $E(p, p) = E(p)$ reduces to the entropy $H(y, i)$ of the random variables y and i. Additionally, since we assumed that $p(i) = 1/N$, the marginal entropy $H(i) = \log N$ is constant and, due to the equipartition condition $p(y) = 1/K$, we have $H(y) = \log K$ is *also* constant. Subtracting these two constants from the entropy yields:

$$E(p) + \text{const.} = H(y, i) - H(y) - H(i) = -I(y, i).$$

Thus we see that minimizing $E(p)$ is the same as maximizing the *mutual information* between the label y and the data index i.

In our formulation, the maximization above is carried out under the equipartition constraint. We can instead relax this constraint and directly maximize the information $I(y, i)$. However, by rewriting information as the difference $I(y, i) = H(y) - H(y|i)$, we see that the optimal solution is given by $H(y|i) = 0$, which states each data point i is associated to only one label deterministically, and by $H(y) = \ln K$, which is another way of stating the equipartition condition.

In other words, our learning formulation can be interpreted as maximizing the information between data indices and labels while explicitly enforcing the equipartition condition, which is implied by maximizing the information in any case. Compared to minimizing the entropy alone, maximizing information avoids degenerate solutions as the latter carry no mutual information between labels y and indices i.

3.3 Relation to simultaneous representation learning and clustering

In the discussion above, self-labelling amounts to assigning discrete labels to data and can thus be interpreted as clustering. Most of the traditional clustering approaches are *generative*. For example, K-means takes a dataset $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ of vectors and partitions it into K classes in order to minimize the reconstruction error

$$E(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, y_1, \dots, y_N) = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{\mu}_{y_i}\|^2 \tag{10}$$

where $y_i \in \{1, \dots, K\}$ are the data-to-cluster assignments and $\boldsymbol{\mu}_y$ are means approximating the vectors in the corresponding clusters. The K-means energy can thus be interpreted as the average data reconstruction error.

It is natural to ask whether a clustering method such as K-means, which is generative, could be combined with representation learning, which is discriminative. In this setting, the feature vectors $\boldsymbol{x} = \Phi(I)$ are extracted by the neural network Φ from the input data I. Unfortunately, optimizing a loss such as eq. (10) with respect to the clustering and representation parameters is meaningless: in fact, the obvious solution is to let the representation send all the data points to the same constant feature vector and setting all the means to coincide with it, in which case the K-means reconstruction error is zero (minimal).

Nevertheless, DeepCluster (Caron et al., 2018) *does* successfully combine K-means with representation learning. DeepCluster can be related to our approach as follows. Step 1 of the algorithm, namely representation learning via cross-entropy minimization, is exactly the same. Step 2, namely self-labelling, differs: where we solve an optimal transport problem to obtain the pseudo-labels, they do so by running K-means on the feature vectors extracted by the neural network.

DeepCluster does have an obvious degenerate solution: we can assign all data points to the same label and learn a constant representation, achieving simultaneously a minimum of the cross-entropy loss in Step 1 and of the K-means loss in Step 2. The reason why DeepCluster avoids this pitfall is due to the particular interaction between the two steps. First, during Step 2, the features $\boldsymbol{x}_i$ are fixed so K-means cannot pull them together. Instead, the means spread to cover the features as they are, resulting in a balanced partitioning. Second, during the classification step, the cluster assignments y_i are fixed, and optimizing the features $\boldsymbol{x}_i$ with respect to the cross-entropy loss tends to separate them. Lastly, the method in (Caron et al., 2018) also uses other heuristics such as sampling the training data inversely to their associated clusters' size, leading to further regularization.

However, a downside of DeepCluster is that it does not have a single, well-defined objective to optimize, which means that it is difficult to characterize its convergence properties. By contrast, in our formulation, both Step 1 and Step 2 optimize *the same objective*, with the advantage that convergence to a (local) optimum is guaranteed.

3.4 Augmenting self-labelling via data transformations

Methods such as DeepCluster extend the training data via augmentations. In vision problems, this amounts to (heavily) distorting and cropping the input images at random. Augmentations are applied so that the neural network is encouraged to learn a labelling function which is *transformation invariant*. In practice, this is crucial to learn good clusters and representations, so we adopt it here. This is achieved by setting $P_{yi} = \mathbb{E}_t[\log \operatorname{softmax}_y h \circ \Phi(t\boldsymbol{x}_i)]$ where the transformations t are sampled at random. In practice, in Step 1 (representation learning), this is implemented via the application of the random transformations to data batches during optimization via SGD, which is corresponds to the usual data augmentation scheme for deep neural networks.

3.5 Multiple simultaneous self-labellings

Intuitively, the same data can often be clustered in many equally good ways. For example, visual objects can be clustered by color, size, typology, viewpoint, and many other attributes. Since our main objective is to use clustering to learn a good data representation Φ, we consider a multi-task setting in which the same representation is shared among several different clustering tasks, which can potentially capture different and complementary clustering axis.

In our formulation, this is easily achieved by considering *multiple heads* (Ji et al., 2018) $h_1, \ldots, h_T$, one for each of T clustering tasks (which may also have a different number of labels). Then, we optimize a sum of objective functions of the type eq. (6), one for each task, while sharing the parameters of the feature extractor Φ among them.

4 Experiments

In this section, we will evaluate the quality of the learned representations. We first ablate our hyper-parameters and then compare to the state of the art in self-supervised learning, where we find that our method is the best clustering-based feature learner and overall second best on many benchmarks.

4.1 Linear probes and baseline architecture

In order to quantify if a neural network has learned useful feature representations, we follow the standard approach of using linear probes (Zhang et al., 2017). This amounts to solving a difficult task, such as ImageNet classification, by training a linear classifier on top of pre-trained feature representations, which are kept fixed. Linear classifiers heavily rely on the quality of the representation since their discriminative power is low.

We apply linear probes to all intermediate convolutional blocks of networks and train on the ImageNet LSVRC-12 (Deng et al., 2009) and MIT Places (Zhou et al., 2014) datasets, which are the standard benchmarks for evaluation in self-supervised learning. Our base encoder architecture is AlexNet (Krizhevsky et al., 2012), since this is most often used in other unsupervised learning work for the purpose of benchmarking. We insert the probes right after the ReLU layer in each of the five blocks, and denote these entry points `conv1` to `conv5`. Applying the linear probes at each convolutional layer allows studying the quality of the representation learned at different depths of the network. While linear probes are conceptually straightforward, there are several technical details that can affect the final accuracy. We detail the exact setup in the Appendix.

4.2 Ablation

Our method contains two major hyper-parameters. As any clustering method, the number of clusters K (or an equivalent parameter) needs to be defined. Additionally, the number of clustering heads T can be specified. Due to the simplicity of the approach, no other parameters such as balancing losses are needed. In our experiments we specify K and T by denoting the method as "Self-Label$[K \times T]$".

In Table 1 we compare the different choices of T and K. Since the `conv1` and `conv2` do not contain much semantic information (Asano et al., 2019) we evaluate the later layers here. It is clear that increasing the number of heads from $T = 1$ (a-c) to $T = 10$ (e) is the strongest performance gain, while increasing the number of clusters K (i) does not help. The performance gain from increasing the number of times we optimize the label assignment (#opts) yields diminishing returns > 80 (a-c).

There are two versions of AlexNet, differing in the number of the first two convolutional filters: (Krizhevsky et al., 2012) uses (96, 256), while (Krizhevsky, 2014) uses (64, 192), both of which are

Table 1: **Ablation study.** Top-1 linear classification performance on the ILSVRC-2012 validation set. * denotes a larger AlexNet variant and $^+$ a ResNet-50. Self-Label-Sup is a model retrained with the final labels from the corresponding Self-Label model, indicating that once the labels are found, they recover the performance. Res-Sup is trained using the labels obtained by a Self-Label ResNet-50.

	Method	#opt.	c3	c4	c5
(a)	Self-Label* [3k $\times$ 1]	40	42.7	43.4	39.2
(b)	Self-Label* [3k $\times$ 1]	80	43.0	44.7	40.9
(c)	Self-Label* [3k $\times$ 1]	160	42.4	44.6	40.7
(d)	Self-Label [3k $\times$ 1]	100	41.6	41.3	39.5
(e)	Self-Label* [3k $\times$ 10]	100	44.5	46.7	43.3
(f)	Self-Label-Sup* [3k $\times$ 10]	0	43.6	46.5	43.3
(g)	Self-Label$^+$ [3k $\times$ 1]	80	Top-1: 50.5		
(h)	Self-Label-Res-Sup* [3k $\times$ 1]	0	43.3	45.0	42.2
(i)	Self-Label* [10k $\times$ 1]	80	41.9	42.7	37.6

Table 2: **PascalVOC finetuning.** VOC2007-Classification %mAP, VOC2007-Detection %mAP and VOC2012-Segmentation %mIU. We **bold** the best result in each layer and <u>underline</u> the second best. * denotes a larger AlexNet variant.

	PascalVOC Task		
Method	**Cls.**	**Det.**	**Seg.**
ImageNet labels	79.9	59.1	48.0
Random	53.3	43.4	–
Random Rescaled	56.6	45.6	32.6
BiGAN	60.1	46.9	35.2
Context*	65.3	51.1	–
Context 2	69.6	55.8	41.4
CC+VGG	72.5	56.5	42.6
RotNet	73.0	54.4	39.1
DeepCluster*	73.4	55.4	<u>45.1</u>
RotNet+retrieval*	<u>74.7</u>	**58.0**	**45.9**
Self-Label* [3k $\times$ 10]	**74.9**	<u>55.9</u>	43.7

used in related work. We have denoted methods using the larger AlexNet with * in all tables. In the ablation we find that the smaller variant (d) does show worse performance across all layers.

To assess the quality of the learned label assignment, we train a model from scratch (f) with the final labels derived from a previous training (e) and without optimizing the label assignment. We find that this recovers the original performance. This is an interesting result, indicating that the quality of the features depends on the final label assignment and not on the intermediate "label-paths" during training. Since the labels are independent of the network architecture, one can use them to pre-train any architecture without running the actual method. To verify this assumption, we use the labels obtained from a Self-Label ResNet-50 (g) and train an AlexNet (h) and find it performing even better than the directly trained AlexNet (b). For this reason we will publish our self-labels for the ImageNet dataset together with the code and trained model.

As we show in the appendix, the labels identified by our algorithm are highly meaningful and group visually similar concepts in the same clusters, often even capturing whole Imagenet classes.

4.3 Linear Probes Benchmarks

To compare to the state of the art and concurrent work, we evaluate several architectures using linear probes on public benchmark datasets.

AlexNet. The main benchmark for feature learning methods is linear probing of an AlexNet on ImageNet. In Table 3 we compare the performance across layers also on the Places dataset. We find that across both datasets our method outperforms DeepCluster at every layer. From our ablation study in Table 1 we also note that even our single head variant $[3k \times 1]$ outperforms DeepCluster, which searches for the optimal number of clusters resulting $10k$ clusters. Furthermore we find that our method is either first or second best in all layers and datasets, and the best method that utilizes a combination of two self-supervised modalities, which are known to increase performance (Doersch & Zisserman, 2017) but blur the sources of gain. Barring this combining method, we improve upon the latest single self-supervision benchmark, Auto-Encoding-Transformations (AET) by 2.6%

Larger models. Training better models than AlexNets is not yet standardized in the feature learning community. In Table 4 we compare a ResNet-50 trained with our method to other works. We perform better than all other methods except the computationally very expensive BigBiGAN.

4.4 Fine-tuning: Classification, object detection and semantic segmentation

Finally, since pre-training is usually aimed at improving down-stream tasks, we evaluate the quality of the learned features by fine-tuning the model for three distinct tasks on the Pascal VOC benchmark. In

Table 3: **Linear probing evaluation - AlexNet.** A linear classifier is trained on the (downsampled) activations of each layer in the pretrained model. We **bold** the best result in each layer and underline the second best. The best layer is highlighted in blue. * denotes a larger AlexNet variant. The numbers are taken from (Feng et al., 2019) except those with ‡, which are taken from their original publications. See Table 6 in the Appendix for a full version of this table.

	ILSVRC-12					Places				
Method	c1	c2	c3	c4	c5	c1	c2	c3	c4	c5
ImageNet supervised	19.3	36.3	44.2	48.3	50.5	22.7	34.8	38.4	39.4	38.7
Places supervised	-	-	-	-	-	22.1	35.1	40.2	43.3	44.6
Random	11.6	17.1	16.9	16.3	14.1	15.7	20.3	19.8	19.1	17.5
Inpainting, (Pathak et al., 2016)	14.1	20.7	21.0	19.8	15.5	18.2	23.2	23.4	21.9	18.4
BiGAN, (Donahue et al., 2017)	17.7	24.5	31.0	29.9	28.0	22.0	28.7	31.8	31.3	29.7
Instance retrieval, (Wu et al., 2018)	16.8	26.5	31.8	34.1	35.6	18.8	24.3	31.9	34.5	33.6
RotNet, (Gidaris et al., 2018)	18.8	31.7	38.7	38.2	36.5	21.5	31.0	35.1	34.6	33.7
DeepCluster (RGB)*‡, (Caron et al., 2018)	18.0	32.5	39.2	37.2	30.6	-	-	-	-	-
AND*‡,(Jiabo Huang & Zhu, 2019)	15.6	27.0	35.9	39.7	37.9	-	-	-	-	-
DeepCluster*, (Caron et al., 2018)	13.4	32.3	41.0	39.6	38.2	23.8	32.8	37.3	36.0	31.0
AET*‡,(Zhang et al., 2019)	19.3	35.4	44.0	43.6	42.4	22.1	32.9	37.1	36.2	34.7
RotNet+retrieval*, (Feng et al., 2019)	22.2	**38.2**	**45.7**	**48.7**	**48.3**	**25.5**	**36.0**	**40.1**	**42.2**	**41.3**
Self-Label* [3k × 10]	**22.4**	36.9	44.5	46.8	43.3	26.4	35.0	39.6	41.1	38.5

Table 4: **Linear probing evaluation - ResNet.** ResNet-50 and ResNet-101 perform comparably in this task. We have separated much larger architectures such as RevNet-50×4 and ResNet-170. Results from from (Donahue & Simonyan, 2019) and (Kolesnikov et al., 2019). We **bold** the best result in each layer and underline the second best. See Table 7 in the Appendix for a full version of this table.

Method	**Architecture**	**Evaluation details (epochs)**	**Top-1**	**Top-5**
Supervised, (Donahue & Simonyan, 2019)	ResNet-50	Adam, LR sweeps (135)	76.3	93.1
Supervised, (Donahue & Simonyan, 2019)	ResNet-101	Adam, LR sweeps (135)	77.8	93.8
Jigsaw, (Kolesnikov et al., 2019)	ResNet-50	SGD (500)	38.4	–
Rotation, (Kolesnikov et al., 2019)	ResNet-50	SGD (500)	43.8	–
CPC, (Oord et al., 2018)	ResNet-101	SGD (145)	48.7	73.6
BigBiGAN, (Donahue & Simonyan, 2019)	ResNet-50	Adam, LR sweeps (135)	**55.4**	**77.4**
Self-Label [3k × 1]	ResNet-50	SGD (145)	50.5	74.3
other architectures				
Rotation, (Kolesnikov et al., 2019)	RevNet-50×4	SGD (500)	55.4	-
BigBiGAN, (Donahue & Simonyan, 2019)	RevNet-50×4	Adam, LR sweeps (135)	60.8	81.4
Efficient CPC, (Hénaff et al., 2019)	ResNet-170	SGD (145)	61.0	83.0

Table 2 we compare results w.r.t. multi-label classification, object detection and semantic segmentation on PascalVOC (Everingham et al.).

As in the linear probe experiments, we find our method better or close to the best performing method. This shows that our trained convolutional network does not only learn useful feature representations but is also able to perform well on actual down-stream tasks.

5 Conclusion

We present a self-supervised feature learning method that is based on clustering. In contrast to other methods, our method optimizes the same objective during feature learning and during clustering. This becomes possible through a weak assumption that the number of samples should be equal across clusters. This constraint is explicitly encoded in the label assignment step and can be solved for efficiently using a modified Sinkhorn-Knopp algorithm. Our method outperforms all other clustering-based feature learning approaches and the resulting self-labels can be used to learn features for new architectures using simple cross-entropy training.

References

Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proc. ICCV*, pp. 37–45. IEEE, 2015. 2

R. Arandjelović and A. Zisserman. Look, listen and learn. In *Proc. ICCV*, 2017. 2

Yuki M Asano, Christian Rupprecht, and Andrea Vedaldi. Surprising effectiveness of few-image unsupervised feature learning. *arXiv preprint arXiv:1904.13132*, 2019. 6

Francis R. Bach and Zaïd Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (eds.), *Advances in Neural Information Processing Systems 20*, pp. 49–56. Curran Associates, Inc., 2008. 2

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. 1

Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. In *Proc. ICML*, pp. 517–526. PMLR, 2017. 2

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *Proc. ECCV*, 2018. 1, 2, 5, 8, 12, 13, 14

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pp. 2292–2300, 2013. 3, 4, 12

Virginia R de Sa. Learning classification with unlabeled data. In *NIPS*, pp. 112–119, 1994. 2

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 1, 6

Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proc. ICCV*, 2017. 7, 14

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proc. ICCV*, pp. 1422–1430, 2015. 2, 14

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning, 2019. 2, 8, 14

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *Proc. ICLR*, 2017. 2, 8, 14

A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE PAMI*, 38(9):1734–1747, Sept 2016. ISSN 0162-8828. doi: 10.1109/TPAMI.2015.2496141. 2

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 8

Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning by rotation feature decoupling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10364–10374, 2019. 2, 8, 14

Chuang Gan, Boqing Gong, Kun Liu, Hao Su, and Leonidas J Guibas. Geometry guided convolutional neural networks for self-supervised video representation learning. In *Proc. CVPR*, 2018. 2

Rouhan Gao, Dinesh Jayaraman, and Kristen Grauman. Object-centric representation learning from unlabeled videos. In *Proc. ACCV*, 2016. 2

Spyros Gidaris, Praveen Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 2018. 2, 8, 14

Olivier J Hénaff, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 2, 8, 14

Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. In *Proc. ICLR*, 2015. 2

Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In *Proc. ICCV*, 2015. 2

Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *Proc. CVPR*, 2016. 2

Simon Jenni and Paolo Favaro. Self-supervised feature learning by learning to spot artifacts. In *Proc. CVPR*, 2018. 2, 14

Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information distillation for unsupervised image segmentation and clustering. *arXiv preprint arXiv:1807.06653*, 2018. 2, 6

Shaogang Gong Jiabo Huang, Qi Dong and Xiatian Zhu. Unsupervised deep learning by neighbourhood discovery. In *Proceedings of the International Conference on machine learning (ICML)*, 2019. 2, 8, 14

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. *arXiv preprint arXiv:1901.09005*, 2019. 8, 14

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pp. 1106–1114, 2012. 6

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014. 6

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proc. CVPR*, 2017. 2

Hsin-Ying Lee, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequence. In *Proc. ICCV*, 2017. 2

A. Mahendran, J. Thewlis, and A. Vedaldi. Cross pixel optical-flow similarity for self-supervised learning. In *Proc. ACCV*, 2018. 2

Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *Proc. ECCV*, 2016. 2

T Mundhenk, Daniel Ho, and Barry Y. Chen. Improvements to context based self-supervised learning. In *Proc. CVPR*, 2017. 2

T. Nathan Mundhenk, Daniel Ho, and Barry Y. Chen. Improvements to context based self-supervised learning. pp. 9339–9348, 2018. 14

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proc. ECCV*, pp. 69–84. Springer, 2016. 2, 14

Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Proc. ICCV*, 2017. 2, 14

Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *Proc. CVPR*, 2018. 2, 14

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2, 8, 14

Andrew Owens, Phillip Isola, Josh H. McDermott, Antonio Torralba, Edward H. Adelson, and William T. Freeman. Visually indicated sounds. In *Proc. CVPR*, pp. 2405–2413, 2016. 2

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proc. CVPR*, pp. 2536–2544, 2016. 2, 8, 14

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

Zhongzheng Ren and Yong Jae Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *Proc. CVPR*, 2018. 2

Pierre Sermanet et al. Time-contrastive networks: Self-supervised learning from video. In *Proc. Intl. Conf. on Robotics and Automation*, 2018. 2

N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *Proc. ICML*, 2015. 2

Huy V Vo, Francis Bach, Minsu Cho, Kai Han, Yann LeCun, Patrick Pérez, and Jean Ponce. Unsupervised image matching and object discovery as optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8287–8296, 2019. 2

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proc. ICCV*, pp. 2794–2802, 2015. 2

Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *Proc. ICCV*, 2017. 2

D. Wei, J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *Proc. CVPR*, 2018. 2

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018. 2, 8, 12, 14

Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2547–2555, 2019. 2, 8, 14

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Proc. ECCV*, pp. 649–666. Springer, 2016. 2, 14

Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proc. CVPR*, 2017. 6, 12, 14

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pp. 487–495, 2014. 6

A Appendix

A.1 Implementation Details

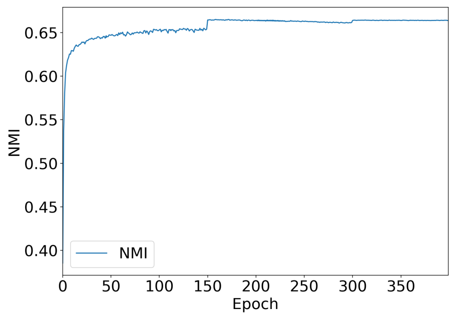
Learning details Unless otherwise noted, we train all our self-supervised models with SGD and intial learning rate 0.05 for 400 epochs with two learning rate drops where we divide the rate by ten at 150 and 300 epochs. We spread our pseudolabel optimizations throughout the whole training process in a logarithmic distribution. We optimize the label assignment at $t_i = \left(\frac{i}{M-1}\right)^2, i \in \{1, \dots, M\}$, where M is the user-defined number of optimizations and t_i is expressed as a fraction of total training epochs. For the Sinkhorn-Kopp optimization we set $\lambda = 25$ as in (Cuturi, 2013). We use standard data augmentations during training that consist of randomly resized crops, horizontal flipping and adding noise, as in (Wu et al., 2018).

Linear Probes - Technical Details. Unfortunately, prior work has used several slightly different setups, so that comparing results between different publications must be done with caution.

In our implementation, we follow the original proposal (Zhang et al., 2017) in pooling each representation to a vector with $9600, 9216, 9600, 9600, 9216$ dimensions for `conv1-5` using adaptive max-pooling, and absorb the batch normalization weights into the preceding convolutions. For evaluation on ImageNet we follow RotNet to train linear probes: images are resized such that the shorter edge has a length of 256 pixels, random crops of 224×224 are computed and flipped horizontally with 50% probability. Learning lasts for 36 epochs and the learning rate schedule starts from 0.01 and is divided by five at epochs 5, 15 and 25. The top-1 accuracy of the linear classifier is then measured on the ImageNet validation subset by extracting 10 crops for each validation image (four at the corners and one at the center along with their horizontal flips) and averaging the prediction scores before the accuracy is computed.

A.2 Further Details

NMI over time In Figure A.1 we find that most learning takes place in the early epochs, and we reach a final NMI value of around 66%. Similarly, we find that due to the updating of the pseudolabels at regular intervals and our data augmentation, the pseudolabel accuracies keep continously rising without overfitting to these labels.



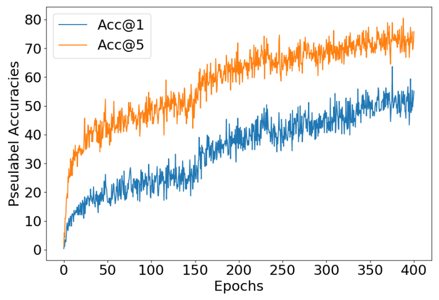


Figure A.1: **Left**: Normalized Mutual Information (NMI) against validation set ImageNet labels. This measure is not used for training but indicates how good a clustering is. **Right**: Pseudolabel accuracies for the training data versus training time. Both plots use the $[10k \times 1]$ AlexNet for comparability with the DeepCluster paper (Caron et al., 2018).

Conv1 filters In Figure A.2 we show the first convolutional filters of two of our trained models. We can find the typical Gabor-like edge detectors as well as color blops and dot-detectors.

Entropy over time In Figure A.3, we show how the distribution of entropy with regards to the true ImageNet labels changes with training time. We find that while at first, all 3000 pseudolabels contain random real ImageNet labels, yielding high entropy of around $6 \approx \ln(400) = \ln(1.2 \cdot 10^6/3000)$. Towards the end of training we arrive at a broad spectrum of entropies with some as low as

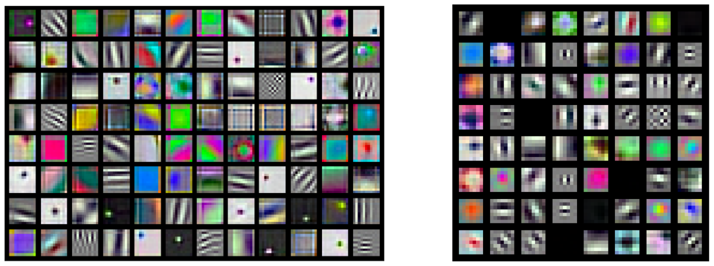


Figure A.2: Visualization of the first convolutional layers of our $[3k \times 10]$ AlexNet (left) and the $[1k \times 1]$ ResNet-50 (right). The filters are scaled to lie between (0,1) for visualization.

$0.07 \approx \ln(1.07)$ (see Fig. A.4 and A.5 for low entropy label visualizations) and the mean around $4.2 \approx \ln(66)$ (see Fig. A.6 and A.7 for randomly chosen labels' visualizations).

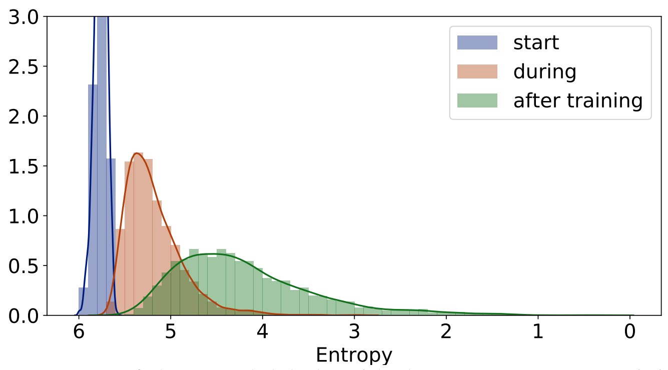


Figure A.3: Cross-entropy of the pseudolabels with the true ImageNet training set labels. This measure is not used for training but indicates how good a clustering is. This plot uses the $[10\text{k} \times 1]$ AlexNet to compare to the equivalent plot in (Caron et al., 2018).

Further AlexNet baselines In Table 5, we report additional linear probe evaluation details for fully-supervisedly trained and random AlexNet 2012 models. Averaging over 10-crops is consistently better.

Table 5: **2012 AlexNet performances.** A linear classifier is trained on the (downsampled) activations of each layer in the pretrained model. See text for technical details.

	ILSVRC-12				
Method	c1	c2	c3	c4	c5
ImageNet labels	19.3	36.3	44.2	48.3	50.5
Random	11.6	17.1	16.9	16.3	14.1
ImageNet labels (10-crop)	22.7	39.7	52.3	59.6	61.3
ImageNet labels (1-crop)	20.5	33.9	41.9	47.4	49.2
Random (10-crop)	17.6	20.3	20.6	17.8	11.0
Random (1-crop)	15.6	16.8	17.4	15.6	10.6

A.3 Complete tables

In the following, we report the unabridged tables with all related work.

Table 6: **Linear probing evaluation - AlexNet.** A linear classifier is trained on the (downsampled) activations of each layer in the pretrained model. We **bold** the best result in each layer and underline the second best. The best layer is highlighted in blue. * denotes a larger AlexNet variant. The numbers are taken from (Feng et al., 2019) except those with ‡, which are taken from their original publications.

	ILSVRC-12					Places				
Method	c1	c2	c3	c4	c5	c1	c2	c3	c4	c5
ImageNet supervised	19.3	36.3	44.2	48.3	50.5	22.7	34.8	38.4	39.4	38.7
Places supervised	-	-	-	-	-	22.1	35.1	40.2	43.3	44.6
Random	11.6	17.1	16.9	16.3	14.1	15.7	20.3	19.8	19.1	17.5
Inpainting, (Pathak et al., 2016)	14.1	20.7	21.0	19.8	15.5	18.2	23.2	23.4	21.9	18.4
BiGAN, (Donahue et al., 2017)	17.7	24.5	31.0	29.9	28.0	22.0	28.7	31.8	31.3	29.7
Context*, (Doersch et al., 2015)	16.2	23.3	30.2	31.7	29.6	19.7	26.7	31.9	32.7	30.9
Colorization, (Zhang et al., 2016)	13.1	24.8	31.0	32.6	31.8	16.0	25.7	29.6	30.3	29.7
Jigsaw, (Noroozi & Favaro, 2016)	18.2	28.8	34.0	33.9	27.1	23.0	31.9	35.0	34.2	29.3
Counting, (Noroozi et al., 2017)	18.0	30.6	34.3	32.5	25.7	23.3	33.9	36.3	34.7	29.6
SplitBrain, (Zhang et al., 2017)	17.7	29.3	35.4	35.2	32.8	21.3	30.7	34.0	34.1	32.5
Instance retrieval, (Wu et al., 2018)	16.8	26.5	31.8	34.1	35.6	18.8	24.3	31.9	34.5	33.6
CC+VGG-, (Noroozi et al., 2018)	19.2	32.0	37.3	37.1	34.6	22.9	34.2	37.5	37.1	34.4
Context 2 (Mundhenk et al., 2018)	19.6	31.8	37.6	37.8	33.7	23.7	34.2	37.2	37.2	34.9
RotNet, (Gidaris et al., 2018)	18.8	31.7	38.7	38.2	36.5	21.5	31.0	35.1	34.6	33.7
Artifacts, (Jenni & Favaro, 2018)	19.5	33.3	37.9	38.9	34.9	23.3	34.3	36.9	37.3	34.4
DeepCluster (RGB)*‡, (Caron et al., 2018)	18.0	32.5	39.2	37.2	30.6	-	-	-	-	-
AND*‡,(Jiabo Huang & Zhu, 2019)	15.6	27.0	35.9	39.7	37.9	-	-	-	-	-
DeepCluster*, (Caron et al., 2018)	13.4	32.3	41.0	39.6	38.2	23.8	32.8	37.3	36.0	31.0
AET*‡,(Zhang et al., 2019)	19.3	35.4	44.0	43.6	42.4	22.1	32.9	37.1	36.2	34.7
RotNet+retrieval*, (Feng et al., 2019)	22.2	**38.2**	**45.7**	**48.7**	**48.3**	**25.5**	**36.0**	**40.1**	**42.2**	**41.3**
Self-Label* [3k × 10]	**22.4**	36.9	44.5	46.8	43.3	26.4	35.0	39.6	41.1	38.5

Table 7: **Linear probing evaluation - ResNet.** ResNet-50 and ResNet-101 perform comparably in this task. We have separated much larger architectures such as RevNet-50×4 and ResNet-170. Results from from (Donahue & Simonyan, 2019) and (Kolesnikov et al., 2019). We **bold** the best result in each layer and underline the second best.

Method	**Architecture**	**Evaluation details (epochs)**	**Top-1**	**Top-5**
Supervised, (Donahue & Simonyan, 2019)	ResNet-50	Adam, LR sweeps (135)	76.3	93.1
Supervised, (Donahue & Simonyan, 2019)	ResNet-101	Adam, LR sweeps (135)	77.8	93.8
Jigsaw, (Kolesnikov et al., 2019)	ResNet-50	SGD (500)	38.4	–
RelPathLoc, (Kolesnikov et al., 2019)	ResNet-50	SGD (500)	42.2	–
Exemplar, (Kolesnikov et al., 2019)	ResNet-50	SGD (500)	43.0	–
Rotation, (Kolesnikov et al., 2019)	ResNet-50	SGD (500)	43.8	–
Multi-task, (Doersch & Zisserman, 2017)	ResNet-101	*unclear*	–	69.3
CPC, (Oord et al., 2018)	ResNet-101	SGD (145)	48.7	73.6
BigBiGAN, (Donahue & Simonyan, 2019)	ResNet-50	Adam, LR sweeps (135)	**55.4**	**77.4**
Self-Label [3k × 1]	ResNet-50	SGD (145)	50.5	74.3
other architectures				
Rotation, (Kolesnikov et al., 2019)	RevNet-50×4	SGD (500)	55.4	-
BigBiGAN, (Donahue & Simonyan, 2019)	RevNet-50×4	Adam, LR sweeps (135)	60.8	81.4
Efficient CPC, (Hénaff et al., 2019)	ResNet-170	SGD (145)	61.0	83.0

A.4 Low Entropy Pseudoclasses

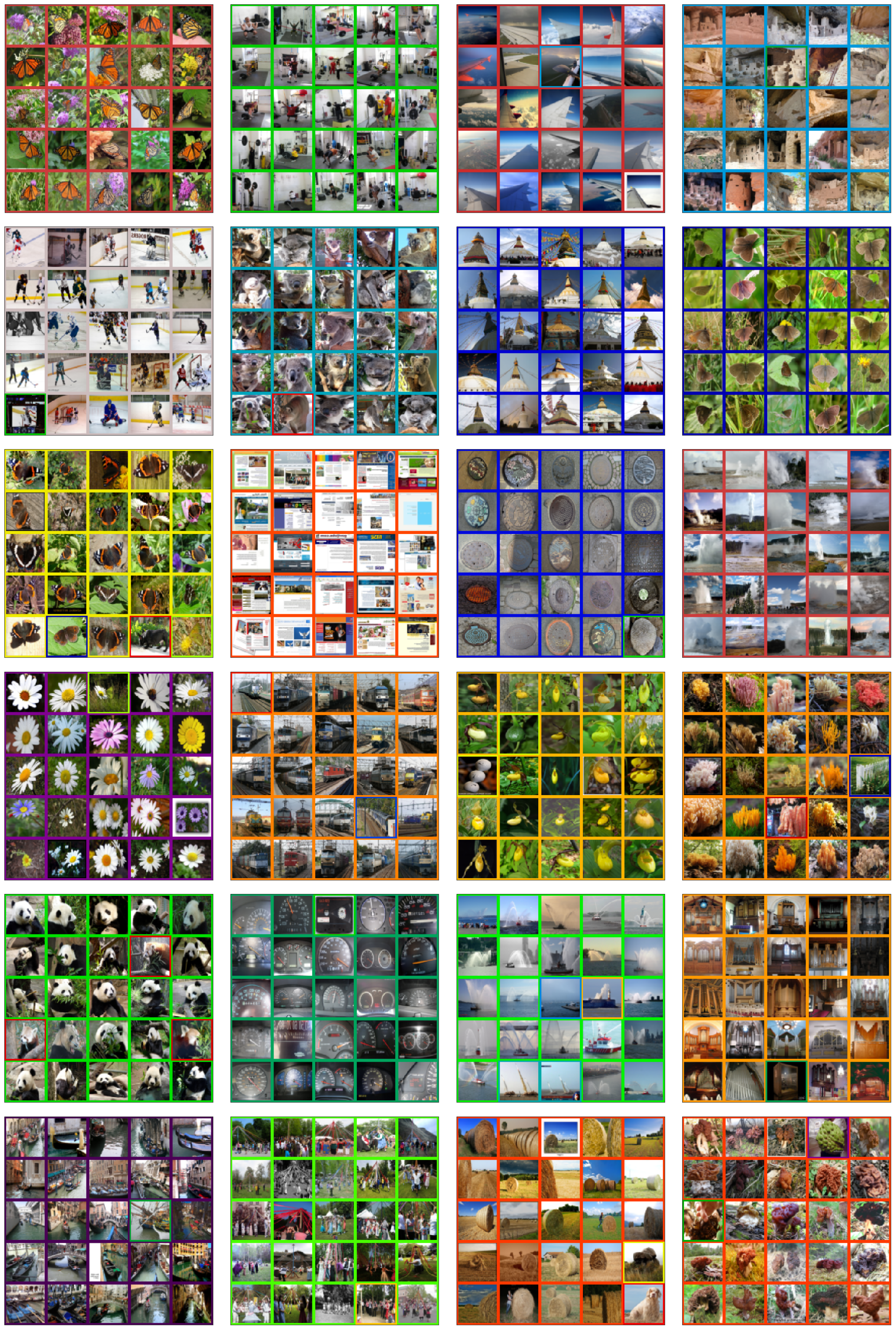


Figure A.4: Here we show a random sample of images associated to the lowest entropy pseudoclasses. The entropy is given by true image labels which are also shown as a frame around each picture with a random color. This visualization uses ResNet-50 $[3k \times 1]$. The entropy varies from $0.07 - -0.83$

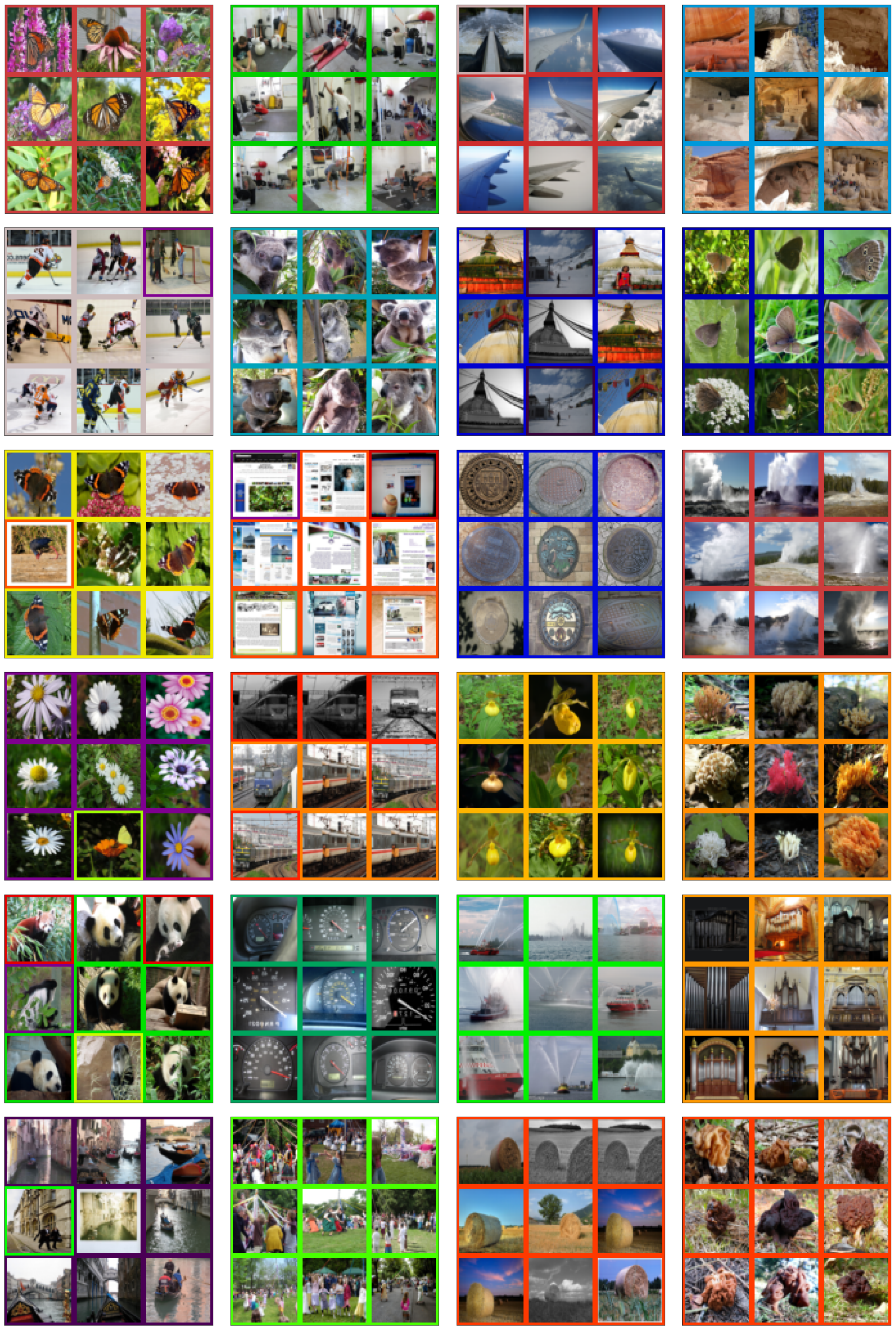


Figure A.5: Visualization of pseudoclasses on the validation set. Here we show random samples of validation set images associated to the lowest entropy pseudoclasses of training set. For further details, see Figure A.4. Classes with less than 9 images are sampled with repetition.

A.5 Random Pseudoclasses

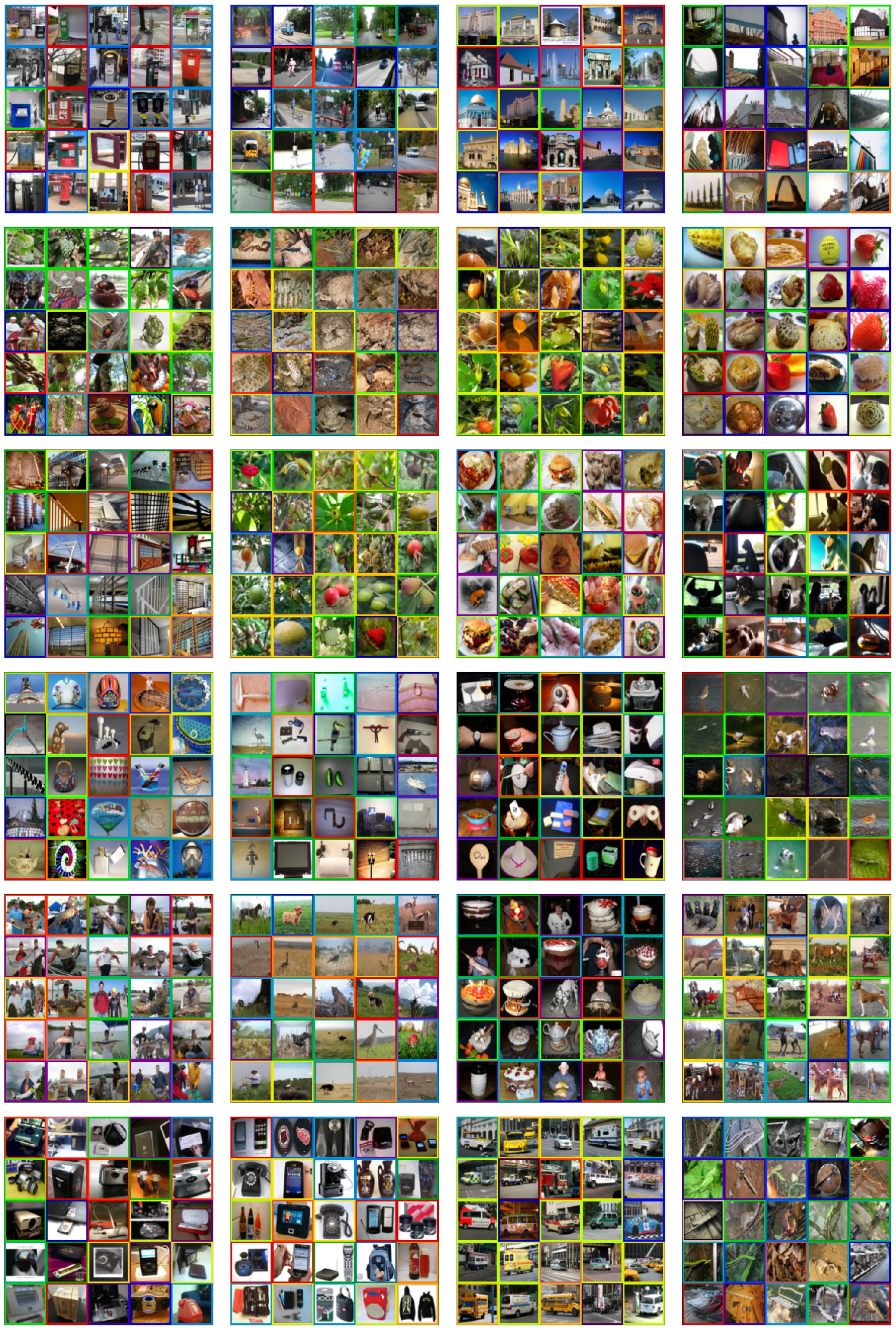


Figure A.6: Here we show a random sample of Imagenet training set images associated to the random pseudoclasses. The entropy is given by true image labels which are also shown as a frame around each picture with a random color. This visualization uses ResNet-50 $[3k \times 1]$.

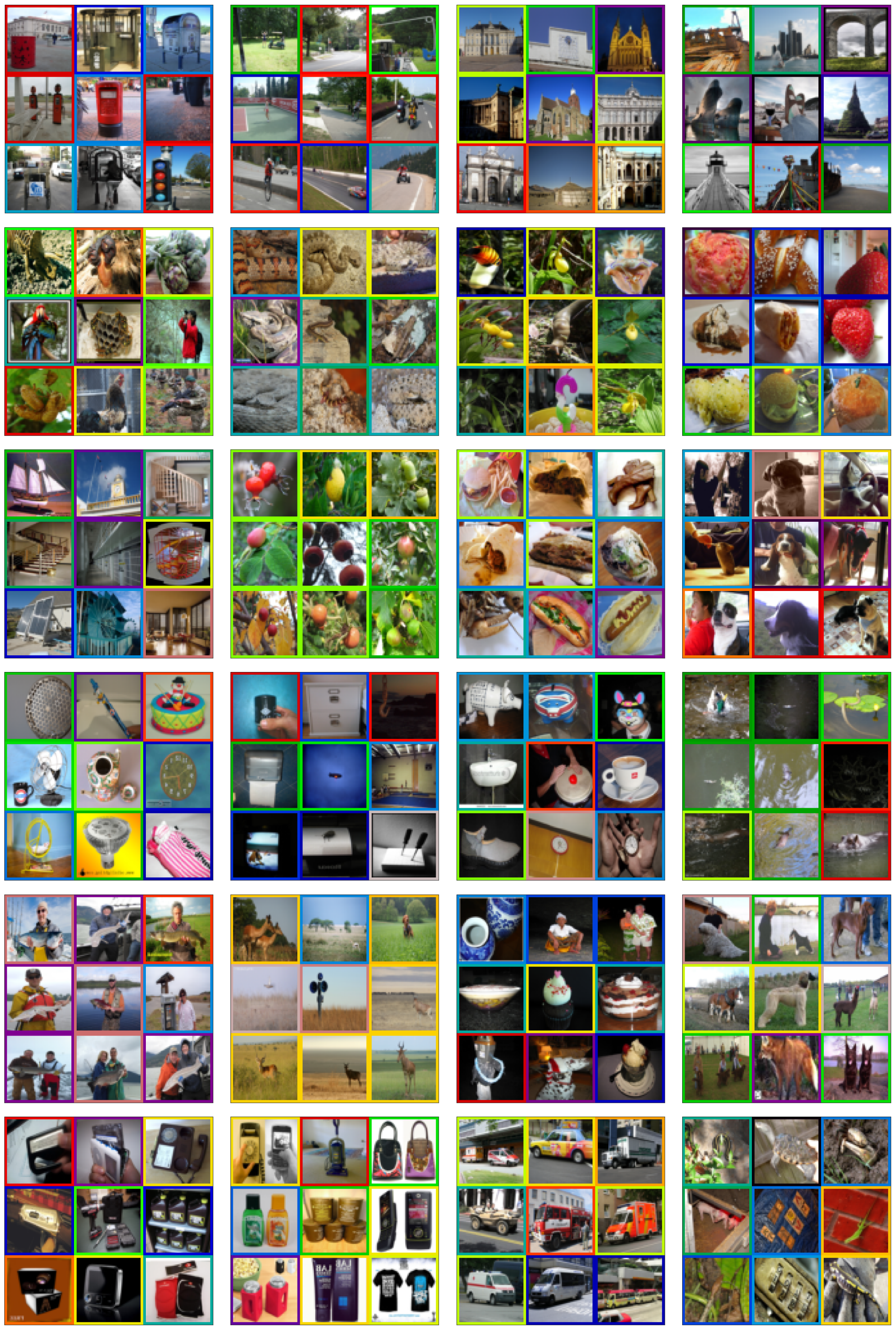


Figure A.7: Here we show a random sample of valdation set images associated to random pseudo-classes. The entropy is given by true image labels which are also shown as a frame around each picture with a random color. This visualization uses ResNet-50 $[3k \times 1]$. Classes with less than 9 images are sampled with repetition.