Boosting Self-Supervised Learning via Knowledge Transfer

Mehdi Noroozi[1] Ananth Vinjimoor[2] Paolo Favaro[1] Hamed Pirsiavash[2]
University of Bern[1] University of Maryland, Baltimore County[2]
{noroozi,favaro}@inf.unibe.ch {kan8,hpirsiav@umbc.edu}

Abstract

In self-supervised learning, one trains a model to solve a so-called pretext task on a dataset without the need for human annotation. The main objective, however, is to transfer this model to a target domain and task. Currently, the most effective transfer strategy is fine-tuning, which restricts one to use the same model or parts thereof for both pretext and target tasks. In this paper, we present a novel framework for self-supervised learning that overcomes limitations in designing and comparing different tasks, models, and data domains. In particular, our framework decouples the structure of the self-supervised model from the final task-specific fine-tuned model. This allows us to: 1) quantitatively assess previously incompatible models including handcrafted features; 2) show that deeper neural network models can learn better representations from the same pretext task; 3) transfer knowledge learned with a deep model to a shallower one and thus boost its learning. We use this framework to design a novel self-supervised task, which achieves state-of-the-art performance on the common benchmarks in PASCAL VOC 2007, ILSVRC12 and Places by a significant margin. Our learned features shrink the mAP gap between models trained via self-supervised learning and supervised learning from 5.9% *to* 2.6% *in object detection on PASCAL VOC 2007.*

1. Introduction

Self-supervised learning (SSL) has gained considerable popularity since it has been introduced in computer vision [7, 39, 23, 20]. Much of the popularity stems from the fact that SSL methods learn features without using manual annotation by introducing a so-called *pretext task*. Feature representations learned through SSL in computer vision are often transferred to a *target data domain* and a *target task*, such as object classification, detection and semantic segmentation in PASCAL VOC. These learned features implicitly define a metric on the data, *i.e*., which data samples are similar and which ones are dissimilar. Thus, the main objective of a pretext task is to learn a metric that makes images of the same object category similar and images of different categories dissimilar. A natural question is then: How do we design such a task? Some SSL approaches define pretext tasks through explicit desirable invariances of the metric [32, 33, 10, 23] or such that they implicitly require a good object representation [7, 21, 39].

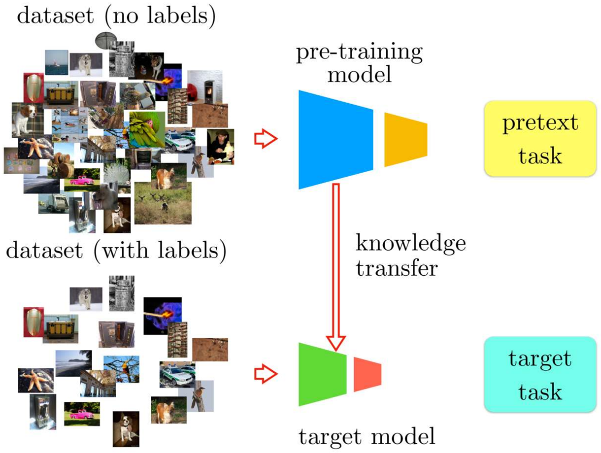

Figure 1: Most current self-supervised learning approaches use the same architecture both in pre-training and fine-tuning. We develop a knowledge transfer method to decouple these two architectures. This allows us to use a deeper model in pre-training.

Even if we had a clear strategy to relate pretext tasks to a target task, comparing and understanding which one is better presents challenges. Most of the recent approaches transfer their learned features to a common supervised target task. This step, however, is complicated by the need to use the same model (*e.g*., AlexNet [17]) to solve both tasks. This clearly poses a major limitation on the design choices. For example, some pretext tasks may exploit several data domains (*e.g*., sound, text, videos), or may exploit different datasets sizes and formats, or might require very deep neural networks to be solved.

There is thus the need to build better representations by exploring and comparing difficult, but learnable [29], pretext tasks and arbitrarily deep architectures. Towards this goal, one could use methods such as *distillation* [12, 4] to transfer the representation in a strong model (the one trained with the pretext task) to a smaller one (the one employed on the target task).

In this paper, we propose to transfer knowledge by reducing a learned representation to *pseudo-labels* on an unlabeled dataset. First, we compute the features learned through a pretext task which might use a complex model on a dataset of unlabeled images. Second, we cluster the features (*e.g.*, using k-means) and use the cluster ID as pseudo-labels for unlabeled images. Third, we learn our final representation by training a smaller deep network (*e.g.*, AlexNet) to classify the images based on the pseudo-labels. By reducing the feature representation of a model to data/pseudo-label pairs, it seems that we are discarding a lot of information. However, we know that given good labels that group semantically similar images, standard supervised classification methods work well. Also, we believe that in a good representation space, semantically related images should be close to each other. Hence, pseudo-labels obtained through a simple clustering algorithm should be a robust estimate of the learned representation.

Once we have obtained pseudo-labels on some dataset, we can *transfer knowledge by simply training a model to predict those pseudo-labels*. The simplicity and flexibility of our technique allows us to:

1. Transfer knowledge from any model (different network architecture and training settings) to any other final task model; we can thus capture representations of complex architectures as well as complicated pretext tasks (see Figure 1).

2. Compare different models (*e.g.*, learned features via neural network architectures versus handcrafted features), built on different data domains with different pretext tasks using a common reference model, data, and task (e.g., AlexNet, PASCAL VOC, and object detection)

Based on this analysis, to show the effectiveness of our algorithm, we design a novel self-supervised task that is more complicated and uses a deeper model. We start from an existing pretext task, the jigsaw problem [21], and make it more challenging by adding occlusions. We refer to this task as the *Jigsaw*++ problem. Furthermore, we boost its performance by training it on VGG16 [30] and then transferring to AlexNet [17] via our proposed transfer method. The resulting model achieves state-of-the-art-performance on several benchmarks shrinking the gap with supervised learning significantly. Particularly, on object detection with Fast R-CNN on PASCAL VOC 2007, Jigsaw++ achieves 56.5% mAP, while supervised pre-training on ImageNet achieves 59.1% mAP. Note that the final model in both cases uses the same AlexNet architecture. We believe our knowledge transfer method can potentially boost the performance of shallow representations with the help of more complicated pretext tasks and architectures.

2. Prior Work

Self-supervised learning. As mentioned in the introduction, SSL consists of learning features using a pretext task. Some methods define as task the reconstruction of data at the pixel level from partial observations and are thus related to denoising autoencoders [31]. Notable examples are the *colorization problem* [39, 18], where the task is to reconstruct a color image given its gray scale version. Another example is image inpainting [27], where the task is to predict a region of the image given the surrounding. Another category of methods uses temporal information in videos. Wang and Gupta [32] learn a similarity metric using the fact that tracked patches in a video should be semantically related. Another type of pretext task, is the reconstruction of more compact signals extracted from the data. For example, Misra *et al.* and Brattoli *et al.* [20, 3] train a model to discover the correct order of video frames. Doersch *et al.* [7] train a model that predicts the spatial relation between image patches of the image. Noroozi and Favaro [21] propose to solve jigsaw puzzles as a pretext task. Pathak *et al.* [26] obtain a supervisory signal by segmenting an image into foreground and background through the optical flow between neighboring frames of a video. Then, they train a model to predict this segmentation from a single image. Other methods also use external signals that may come freely with visual data. The key idea is to relate images to information in other data domains like ego-motion [15, 1] or sound [25]. Finally, recent work [23] has also exploited the link between relative transformations of images to define relative transformations of features.

Currently, the above pretext tasks have been assessed through transfer learning and benchmarked on common neural network architectures (*e.g.*, AlexNet) and datasets (*e.g.*, PASCAL). However, so far it is unclear how to design or how to further improve the performance of SSL methods. One natural direction is the combination of multiple tasks [8, 33]. However, this strategy does not seem to scale well as it becomes quickly demanding in terms of computational resources. Moreover, a possible impediment to progress is the requirement of using the same model both for training on the pretext task and to transfer to another task/domain. In fact, as we show in our experiments, one can learn better representations from challenging tasks by using deep models, than by using shallower ones. Through our knowledge transfer method we map the representation in the deep model to a reference model (*e.g.*, AlexNet) and show that it is better than the representation learned directly with the reference model. This allows to improve SSL methods by exploring: 1) designs of architectures that may be more suitable for learning a specific pretext task; 2) data formats and types different from the target domain; 3) more challenging pretext tasks. To illustrate these advantages, we make the method of Noroozi and Favaro [21] more challenging by incorporating occlusions in the tiles.

Knowledge distillation. Since we transfer knowledge from a model trained on a pretext task to a target model, our work is related to *model distillation*. [12, 2] perform knowledge distillation by training a target model that mimics the output probability distribution of the source model. [34, 37] extend that method to regressing neuron activations, an approach that is more suitable to our case. Our approach is fundamentally different. We are only interested in preserving the essential metric of the learned representation (the cluster associations), rather than regressing the exact activations. A clustering technique used in Dosovitskiy *et al*. [10] is very related to our method. They also use clustering to reduce the classification ambiguity in their task, when too many surrogate classes are used. However, they only train and re-train the same network and do not exploit it for knowledge transfer. Other related work uses clustering in the feature space of a supervised task to extract labels from unlabeled data for novel categories [36] or building hash functions [35]. Neither of these works uses clustering to transfer knowledge from a deep network to a shallow one. Our HOG experiment is related to [5] which shows the initial layers of VGG perform similarly to hand-crafted SIFT features.

3. Transferring Knowledge

The common practice in SSL is to learn a rich representation by training a model on a SSL pretext task with a large scale unlabeled dataset and then fine-tune it for a final supervised task (*e.g*., PASCAL object detection) by using a limited amount of labeled training data. This framework has an inherent limitation: The final task model and the SSL model must use the same architecture. This limitation becomes more important as the community moves on to more sophisticated SSL pretext tasks with larger scale datasets that need more complicated deeper model architectures. Since we do not want to change the architecture of the final supervised task, we need to develop a novel way of transferring the learned knowledge from the SSL task to the final supervised model. Moreover, when trained on the pretext task, the model learns some extra knowledge that should not be transferred to the final task. For instance, in standard fine-tuning, we usually copy the weights only up to some intermediate convolutional layers and ignore the final layers since they are very specific to the pretext task and are not useful for general visual recognition.

In this section, we propose an algorithm to transfer the part of knowledge learned in SSL that is useful for visual recognition to the target task. Our idea is based on the intuition that in the space of a good visual representation, semantically similar data points should be close to each other. The common practice to evaluate this is to search for nearest neighbors and make sure that all retrieved results are semantically related to the query image. This means a simple clustering algorithm based on the Euclidean distance should

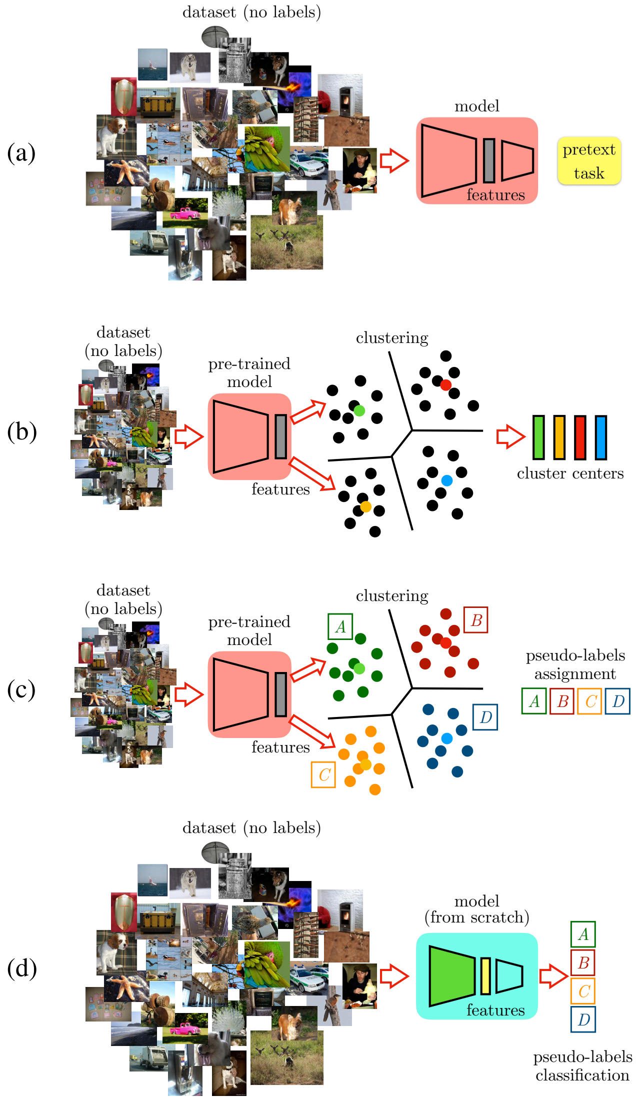

Figure 2: **Knowledge transfer pipeline.** We break down the four steps of our proposed method for knowledge transfer: (a) an arbitrary model is pre-trained on an SSL pretext task; (b) the features extracted from this model are clustered and cluster centers are extracted; (c) pseudo-labels are defined for each image in the dataset by finding the closest cluster center; (d) training of the target model on the classification of the pseudo-labels.

group semantically similar images in the same cluster. Our idea is to perform this clustering in the feature space and to obtain the cluster assignments of each image in the dataset as pseudo-labels. We then train a classifier network with the target task architecture on the pseudo-labels to learn a novel representation. We illustrate our pipeline in Figure 2 and describe it here below.

(a) Self-Supervised Learning Pre-Training. Suppose that we are given a pretext task, a model and a dataset. Our first step in SSL is to train our model on the pretext task with the given dataset (see Figure 2 (a)). Typically, the models of choice are convolutional neural networks, and one considers as feature the output of some intermediate layer (shown as a grey rectangle in Figure 2 (a)).

(b) Clustering. Our next step is to compute feature vectors for all the unlabeled images in our dataset. Then, we use the k-means algorithm with the Euclidean distance to cluster the features (see Figure 2 (b)). Ideally, when performing this clustering on ImageNet images, we want the cluster centers to be aligned with object categories. In the experiments, we typically use 2,000 clusters.

(c) Extracting Pseudo-Labels. The cluster centers computed in the previous section can be considered as *virtual categories*. Indeed, we can assign feature vectors to the closest cluster center to determine a *pseudo-label* associated to the chosen cluster. This operation is illustrated in Figure 2 (c). Notice that the dataset used in this operation might be different from that used in the clustering step or in the SSL pre-training.

(d) Cluster Classification. Finally, we train a simple classifier using the architecture of the target task so that, given an input image (from the dataset used to extract the pseudo-labels), predicts the corresponding pseudo-label (see Figure 2 (d)). This classifier learns a new representation in the target architecture that maps images that were originally close to each other in the pre-trained feature space to close points.

4. The Jigsaw++ Pretext Task

Recent work [8, 33] has shown that deeper architectures can help in SSL with PASCAL recognition tasks (*e.g.*, ResNet). However, those methods use the same deep architecture for both SSL and fine-tuning. Hence, they are not comparable with previous methods that use a simpler AlexNet architecture in fine-tuning. We are interested in knowing how far one can improve the SSL pre-training of AlexNet for PASCAL tasks. Since in our framework the SSL task is not restricted to use the same architecture as in the final supervised task, we can increase the difficulty of the SSL task along with the capacity of the architecture and still use AlexNet at the fine-tuning stage.

Towards this goal, we build on the method of Okanohara *et al.* [24] to learn representations in the text domain. They replace a word at random in a sentence and train a model to distinguish the original sentence from the corrupt one. We combine this idea with the jigsaw [21] task by replacing tiles in the image puzzle with a random tile from other images. We call this the *Jigsaw++ task*. The original pretext task [21] is to find a reordering of tiles from a 3 × 3 grid of a square region cropped from an image. In Jigsaw++, we replace a random number of tiles in the grid (up to 2) with (occluding) tiles from another random image (see Figure 3). The number of occluding tiles (0, 1 or 2 in our experiments) as well as their location are randomly selected. The occluding tiles make the task remarkably more complex. First, the model needs to detect the occluding tiles and second, it needs to solve the jigsaw problem by using only the remaining patches. To make sure we are not adding ambiguities to the task, we remove similar permutations so that the minimum Hamming distance between any two permutations is at least 3. In this way, there is a unique solution to the jigsaw task for any number of occlusions in our training setting. Our final training permutation set includes 701 permutations, in which the average and minimum Hamming distance is .86 and 3 respectively. In addition to applying the mean and std normalization independently at each image tile, we train the network 70% of the time on gray scale images. In this way, we prevent the network from using low level statistics to detect occlusions and solve the jig-

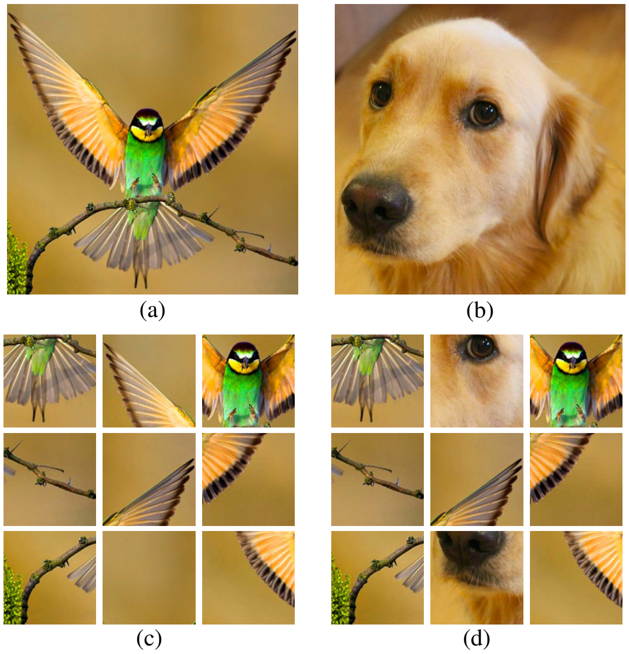
(a)
(b)
(c)
(d)

Figure 3: **The Jigsaw++ task.** (a) the main image. (b) a random image. (c) a puzzle from the original formulation of [21], where all tiles come from the same image. (d) a puzzle in the Jigsaw++ task, where at most 2 tiles can come from a random image.

Table 1: Impact of the number of cluster centers

#clusters	**500**	**1000**	**2000**	**5000**	**10000**
mAP on voc-classification	69.1	69.5	69.9	69.9	70.0

Table 2: Impact of the data domain used in clustering and pseudo-labels: We perform experiments where the training of clustering and extracting pseudo-labels (inference of clustering) are done on different datasets. We see just a little reduction in VOC2007 classification which means the clustering is not relying on the ImageNet bias.

clustering on	ImageNet	ImageNet	Places
pseudo-labels on	ImageNet	Places	ImageNet
mAP on voc-classification	69.9	68.4	68.3

saw task. We train the Jigsaw++ task on both VGG16 and AlexNet architectures. By having a larger capacity with VGG16, the network is better equipped to handle the increased complexity of the Jigsaw++ task and is capable of extracting better representations from the data. Following our pipeline in Figure 2, we train our models with this new SSL task, transfer the knowledge by: 1) clustering the features, 2) assigning pseudo-labels, and 3) training AlexNet to classify the pseudo-labels. We execute the whole pipeline by training VGG16 and AlexNet on the Jigsaw++ task. Our experiments show that when we train VGG16 with the Jigsaw++ task, there is a significantly better performance in fine-tuning. This confirms that training on a deeper network leads to a better representation and corresponding pseudo-labels.

5. Experiments

We extensively evaluate the knowledge transfer method and the Jigsaw++ task on several transfer learning benchmarks including: fine-tuning on PASCAL VOC, nonlinear classification on ImageNet, and linear classification on Places and ImageNet. We also perform ablation studies to show the effect of the number of clusters and the datasets used for clustering and pseudo-label assignment in the transferred knowledge. Our experiments show that pre-training on the Jigsaw++ task yields features that outperform current self-supervised learning methods. In all the evaluations, the weights of the convolutional layers of the target AlexNet model are copied from the corresponding layers of the AlexNet model trained on the pretext task, and the fully connected layers are randomly initialized. We evaluate our knowledge transfer method in several cases while the cluster classification network is always AlexNet:
1) Jigsaw++ trained with VGG16: This procedure achieves the best performance on all the benchmarks.
2) Jigsaw++ trained with AlexNet: Our experiments show that the performance does not drop in this case. This implies that a pretext task can be reduced to a classification task, if it has learned a proper similarity metric.
3) The method of Doersch *et al.* [7] trained on AlexNet: Due to the difficulty of their task, they use batch normalization [14] during training. This makes the fine-tuning challenging, because of the difference in the settings of the target framework. To address this issue, Krähenbühl *et al.* [16] rescale the weights and thus boost the performance in fine-tuning. Our experiments show that our pipeline is doing significantly better than [16] in this case.
4) HOG: We cluster the HOG features to obtain the pseudo-labels. Surprisingly, HOG pseudo-labels yield a high performance in fine-tuning on classification in Pascal VOC.
5) Pseudo-labels of a random network: Zhang *et al.* [38] showed that AlexNet can be trained to predict random labels. We perform an experiment in the same spirit and obtain random pseudo-labels by clustering the features of a randomly initialized network. To make the cluster classification network converge in this case, we decreased the initial learning rate to 0.001. By using this settings, the network is able to predict random pseudo-labels on the training set with a high accuracy. However, its prediction accuracy on the validation set is close to random chance as expected. We found that the classifier network trained on random pseudo-labels yields the same performance on transfer learning on PASCAL VOC as random initialization.
6) Knowledge distillation: The standard knowledge distillation [12] is not directly applicable in many SSL tasks where the loss function does not involve any probability distribution. Similar to [2], we train a student network to regress conv4-5 layers of the teacher network (both AlexNet). The student gets 58.5% on VOC classification while the teacher gets 69.8%. We compare our method to [37] that minimizes the original loss function as well in distillation. We use the Jigsaw++ network trained on VGG as the teacher network and minimize eq. (5) of [37] on AlexNet (the student), where $\mathcal{L}$ has been replaced by the Jigsaw++ loss. As it is shown in Table 3, this method does not boost sensibly the performance on Pascal-VOC dataset.
Implementation Details. We extract the features for all the methods from `conv4` layer and max-pool them to $5 \times 5 \times 384$ in the case of AlexNet and $4 \times 4 \times 512$ in the case of VGG16. We implement the standard k-means algorithm on a GPU with $k = 2K$. It takes around 4 hours to cluster the $1.3M$ images of ImageNet on a single Titan X. We use the standard AlexNet and ImageNet classification settings to train the pseudo-label classifier network.

5.1. Ablation Studies

All the ablation studies are carried out with AlexNet pre-trained on the Jigsaw++ task with ImageNet as dataset (trainset without the labels). The pseudo-labels are also

Table 3: **PASCAL VOC fine-tuning**. Evaluation of SSL methods on classification, detection and semantic segmentation. All rows use the AlexNet architecture at fine-tuning. "CC+" stands for "cluster classification", our knowledge transfer method. "vgg-" means the VGG16 architecture is used before the "CC+" step. In "CC+vgg-Jigsaw++", we train Jigsaw++ using VGG16, cluster, and train AlexNet to predict the clusters. We also transfer using [37] in "[37]+vgg-Jigsaw++". In "CC+HOG", we cluster bag of words of HOG and predict them with AlexNet; this is not fully unsupervised since HOG is hand-crafted. Surprisingly, it outperforms many SSL algorithms on classification.

Method	**Ref**	**Class.**	**Det.**		**Segm.**
			SS	**MS**	
Supervised [17]		79.9	59.1	59.8	48.0
CC+HOG [6]		70.2	53.2	53.5	39.2
Random	[27]	53.3	43.4	-	19.8
ego-motion [1]	[1]	54.2	43.9	-	-
BiGAN [9]	[9]	58.6	46.2	-	34.9
ContextEncoder [27]	[27]	56.5	44.5	-	29.7
Video [32]	[16]	63.1	47.2	-	-
Colorization [39]	[39]	65.9	46.9	-	35.6
Split-Brain [40]	[40]	67.1	46.7	-	36.0
Context [7]	[16]	55.3	46.6	-	-
Context [7]*	[16]	65.3	51.1	-	-
Counting [23]	[23]	67.7	51.4	-	36.6
WatchingObjectsMove [26]	[26]	61.0	-	52.2	-
Jigsaw [21]	[21]	67.7	53.2	-	-
Jigsaw++		69.8	55.5	55.7	38.1
CC+Context-ColorDrop [7]		67.9	52.8	53.4	-
CC+Context-ColorProjection [7]		66.7	51.5	51.8	-
CC+Jigsaw++		69.9	55.0	55.8	40.0
[37]+vgg-Jigsaw++		70.6	54.8	55.2	38.0
CC+vgg-Context [7]		68.0	53.0	53.5	-
CC+vgg-Jigsaw++		**72.5**	**56.5**	**57.2**	**42.6**

assigned to ImageNet data unless specified otherwise. The knowledge transfer is then completed by training AlexNet on the pseudo-labels. Finally, this model is fine-tuned on PASCAL VOC 2007 for object classification.

What is the impact of the number of clusters? The k-means clustering algorithm needs the user to choose the number of clusters. In principle, too few clusters will not lead to discriminative features and too many clusters will not generalize. Thus, we explore different choices to measure the sensitivity of our knowledge transfer algorithm. Since each cluster corresponds to a pseudo-label, we can loosely say that the number of clusters determines the number of object categories that the final network will be able to discern. Therefore, one might wonder if a network trained with very few pseudo-labels develops a worse learning than a network with a very large number of pseudo-labels. This analysis is analogous to work done on the ImageNet labels [13]. Indeed, as shown in Table 1, we find that the network is not too sensitive to the number of clusters. Perhaps, one aspect to further investigate is that, as the number of clusters increases, the number of data samples per cluster decreases. This decrease might cause the network to overfit and thus reduce its performance despite its finer categorization capabilities.

Table 4: **ImageNet classification with a linear classifier.** We use the publicly available code and configuration of [39]. Every column shows the top-1 accuracy of AlexNet on the classification task. The learned weights from `conv1` up to the displayed layer are frozen. The features of each layer are spatially resized until there are fewer than 9K dimensions left. A fully connected layer followed by softmax is trained on a 1000-way object classification task.

Method	**Ref**	**conv1**	**conv2**	**conv3**	**conv4**	**conv5**
Supervised [17]	[40]	19.3	36.3	44.2	48.3	50.5
CC+HOG [6]		16.8	27.4	20.7	32.0	29.1
Random	[40]	11.6	17.1	16.9	16.3	14.1
Context [7]	[40]	16.2	23.3	30.2	31.7	29.6
ContextEncoder [27]	[40]	14.1	20.7	21.0	19.8	15.5
BiGAN [9]	[40]	17.7	24.5	31.0	29.9	28.0
Colorization [39]	[40]	12.5	24.5	30.4	31.5	30.3
Split-Brain [40]	[40]	17.7	29.3	35.4	35.2	32.8
Counting [23]	[23]	18.0	30.6	34.3	32.5	25.7
Jigsaw++		18.2	28.7	34.1	33.2	28.0
CC+Jigsaw++		18.9	30.5	35.7	35.4	32.2
CC+vgg-Jigsaw++		**19.2**	**32.0**	**37.3**	**37.1**	**34.6**

What is the impact of the cluster data domain? Our knowledge transfer method is quite flexible. It allows us to pre-train on a dataset, cluster on another, and then define pseudo-labels on a third one. In this study, we explore some of these options to illustrate the different biases of the datasets. The results are shown in Table 2. In all these experiments, we pre-train AlexNet on the Jigsaw++ task with ImageNet. Then, we decouple the training and inference of the clustering algorithm. For instance, in the right column of Table 2, we learn cluster centers on `conv4` features of Jigsaw++ extracted on Places and then run the assignment of clustering on ImageNet to get pseudo-labels to be used in the training of the final AlexNet. We see only a small reduction in performance, which implies that our clustering method is not relying on particular biases inherent in the ImageNet dataset.

5.2. Transfer Learning Evaluation

We evaluate the features learned with different SSL methods on PASCAL VOC for object classification, detection, and semantic segmentation. Also, we apply our knowledge transfer method to some of these SSL methods under relevant settings. In particular, we apply our knowledge transfer to: Context [7], Context-ColorDropping [7], Context-ColorProjection [7], Jigsaw++, and HOG [6].

Table 5: **Places classification with a linear classifier.** We use the same setting as in Table 4 except that to evaluate generalization across datasets, the model is pre-trained on ImageNet (with no labels) and then tested with frozen layers on Places (with labels).

Method	**conv1**	**conv2**	**conv3**	**conv4**	**conv5**
Places labels [41]	22.1	35.1	40.2	43.3	44.6
ImageNet labels [17]	22.7	34.8	38.4	39.4	38.7
CC+HOG [6]	20.3	30.0	31.8	32.5	29.8
Random	15.7	20.3	19.8	19.1	17.5
Context [7]	19.7	26.7	31.9	32.7	30.9
Jigsaw [22]	23.0	31.9	35.0	34.2	29.3
Context encoder [27]	18.2	23.2	23.4	21.9	18.4
Sound [25]	19.9	29.3	32.1	28.8	29.8
BiGAN [9]	22.0	28.7	31.8	31.3	29.7
Colorization [39]	16.0	25.7	29.6	30.3	29.7
Split-Brain [40]	21.3	30.7	34.0	34.1	32.5
Counting [23]	**23.3**	33.9	36.3	34.7	29.6
Jigsaw++	22.0	31.2	34.3	33.9	22.9
CC+Jigsaw++	22.5	33.0	36.2	36.1	34.0
CC+vgg-Jigsaw++	22.9	**34.2**	**37.5**	**37.1**	**34.4**

Table 6: **ImageNet classification with a nonlinear classifier** as in [21]. Every column shows top-1 accuracy of AlexNet on the classification task. The learned weights from `conv1` up to the displayed layer are frozen. The rest of the network is randomly initialized and retrained. Notice that the reported results of [32] are based on the original paper. All evaluations are done with 10 croppings per image.

Method	**Ref**	**conv1**	**conv2**	**conv3**	**conv4**	**conv5**	**fc6**	**fc7**
Supervised [17]	[17]	57.3	57.3	57.3	57.3	57.3		
Random	[21]	48.5	41.0	34.8	27.1	12.0	-	-
Video [32]	[21]	51.8	46.9	42.8	38.8	29.8		
BiGAN [9]	[9]	55.3	53.2	49.3	44.4	34.9	-	-
Counting [23]	[23]	54.7	52.7	48.2	43.3	32.9	-	-
Context [7]	[21]	53.1	47.6	48.7	45.6	30.4	-	-
Jigsaw [21]	[21]	54.7	52.8	49.7	45.3	34.6	-	-
Jigsaw++		54.7	52.9	50.3	46.1	35.4	-	-
CC+-vgg-Context		55.0	52.0	48.2	44.3	37.9	29.1	20.3
CC+Jigsaw++		55.3	52.2	51.4	47.6	41.1	33.9	25.9
CC+vgg-Jigsaw++		**55.9**	**55.1**	**52.4**	**49.5**	**43.9**	**37.3**	**27.9**

Fine-Tuning on PASCAL VOC. In this set of experiments, we use fine-tuning on PASCAL VOC as a common benchmark for all the SSL methods. The comparisons are based on object classification and detection on VOC2007 using the framework of [16] and Fast-RCNN [11] respectively. We also report semantic segmentation result on VOC2012 dataset using the framework of [19]. We found that in most recent SSL papers, the settings of the detection task used for the evaluation of SSL methods are not the same as the ones used for supervised learning. More specifically, most SSL methods are using multi-scale fine-tuning for 150K iterations, with the basic learning rate of 0.001 and dividing the learning rate by 10 every 50K iterations. Moreover, some of the methods have been evaluated using the multiscale test. We found that fine-tuning supervised weights with these settings achieves 59.1% and 59.9% with multi scale and single scale test respectively. We believe that it would be useful to use these as the baseline. We follow the same settings in all of our evaluations and report the results for both cases. We have locked the first layer in all cases including supervised weights as it is the default settings of Fast-RCNN. In Table 3, we use "CC+" when our knowledge transfer method is used and "vgg-" when VGG16 is used in pre-training. All methods in Table 3 use AlexNet for cluster prediction and fine-tuning. In the case of HOG features [6] we only apply the cluster classification on pseudolabels obtained from HOG on ImageNet. Surprisingly, these handcrafted features yield a very high performance in all three tasks. Our knowledge transfer method does not have a significant impact on the performance when the source and destination architectures are the same. However, when pretraining on VGG16, there is a significant boost of 2.6% in classification, 1.6% in detection, and 2.6% in semantic segmentation. These results show state-of-the-art performance on all tasks. More importantly, the gap between SSL methods and supervised learning methods is further shrinking by a significant margin. We believe that our method allows to use larger scale datasets and deeper models in pre-training, while still using AlexNet in fine-tuning.

Linear Classification. We also evaluate the SSL methods by using a linear classifier on the features extracted from AlexNet at different convolutional layers [40]. We apply this on both ImageNet [28] and Places [41] and evaluate the classification performance on the respective datasets. We illustrate the performance on ImageNet in Table 4 and on Places in Table 5. As can be observed, the performance of Jigsaw++ is comparable to prior state-of-the-art methods. Surprisingly, our knowledge transfer method seems to be beneficial to the transferred model CC+Jigsaw++. Consistently with other experiments, we also observe that pre-training with VGG16 in CC+vgg-Jigsaw++ gives a further substantial boost (an average of almost 2% improvement). We also notice that HOG features do not demonstrate a performance in line with the performance observed on PASCAL VOC. A similar scenario is observed on the Places dataset. Notice that the performance obtained with CC+vgg-Jigsaw++ is quite close to the performance achieved with supervised pre-training on ImageNet labels.

Nonlinear Classification. We freeze several layers initialized with evaluating weights and retrain the remaining layers from scratch. For completeness, we also evaluate SSL features as done in [21], by freezing a few initial layers and training the remaining layers from random initialization. In comparison to the previous experiments, the main difference is that here we use a nonlinear classifier that consists of the final layers of AlexNet. This experi-

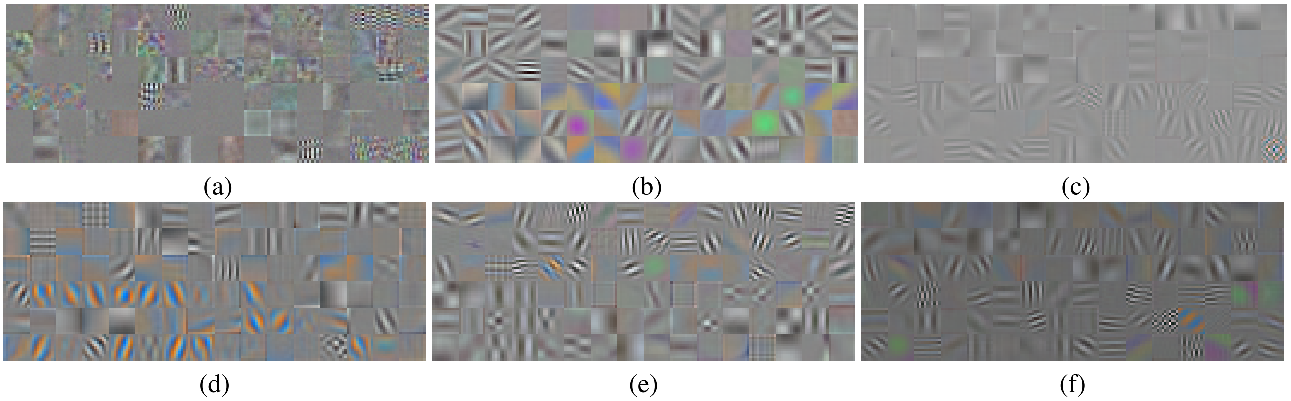

Figure 4: conv1 filters of the cluster classification network using the AlexNet architecture trained with different pseudo-labels obtained from: (a) a randomly initialized AlexNet network, (b) CC+HOG, (c) Doersch *et al.* [7] trained with Color-Dropping, (d) Doersch *et al.* [7] trained with ColorProjection, (e) CC+Jigsaw++ task trained on AlexNet, (f) the CC+vgg-Jigsaw++ task trained on VGG16.

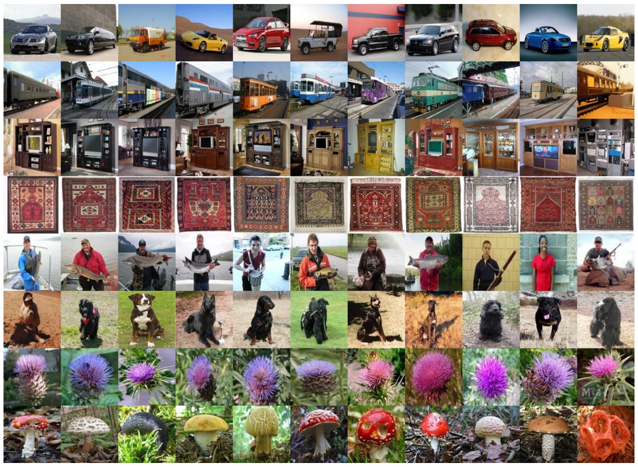

Figure 5: Some cluster samples used to train CC+vgg-Jigsaw++. Each row shows the 11 closest images to their corresponding cluster center.

ments is another way to illustrate the alignment between the pseudo-labels obtained from cluster classification and the ground truth labels. We show the comparisons in Table 6. The performance obtained by most SSL methods seems to confirm the pattern observed in the previous experiments. However, the difference between the previous state-of-the-art and CC+vgg-Jigsaw++ is quite remarkable. Also, the boost in performance of other prior work such as [7] through pre-training with VGG16 is up to 9% at conv5.

5.3. Visualizations

We show some filters of the cluster classifier network in Figure 4. We can see the impact of the pre-trained network on the cluster classifier. Interestingly, there is no color in the filters of the "color dropping" method of [7] after knowledge transfer. This is consistent with the fact that this method does not see any color image in the pre-training stage. We also show some sample clusters used in training CC+vgg-Jigsaw++ in Figure 5. Each row corresponds to images closest to the center of a single cluster. Ideally, for high-quality representations, we expect images from the same category on each row.

6. Conclusions

Self-supervised learning is an attractive research area in computer vision since unlabeled data is abundantly available and supervised learning has serious issues with scaling to large datasets. Most recent SSL algorithms are restricted to using the same network architecture in both the pre-training task and the final fine-tuning task. This limits our ability in using large scale datasets due to limited capacity of the final task model. We have relaxed this constraint by decoupling the pre-training model and the final task model by developing a simple but efficient knowledge transfer method based on clustering the learned features. Moreover, to truly show the benefit, we increase the complexity of a known SSL algorithm, the jigsaw task, and use a VGG network to solve it. We show that after applying our ideas to transfer the knowledge back to AlexNet, it outperforms all state-of-the-art SSL models with a good margin shrinking the gap between supervised and SSL models from %5.9 to %2.6 on PASCAL VOC 2007 object detection task.

Acknowledgements. PF has been supported by the Swiss National Science Foundation (SNSF) grant number 200021_169622. HP has been supported by GE Research and Verisk Analytics.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015. 2, 6

[2] L. J. Ba and R. Caruana. Do deep nets really need to be deep? In *NIPS*, 2014. 3, 5

[3] B. Brattoli, U. Büchler, A. S. Wahl, M. E. Schwab, and B. Ommer. Lstm self-supervision for detailed behavior analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[4] C. Bucila, R. Caruana, and A. Niculescu-Mizil. Model compression. In *KDD*, 2006. 2

[5] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi. Deep filter banks for texture recognition, description, and segmentation. *ICCV*, 2016. 3

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 6, 7

[7] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 1, 2, 5, 6, 7, 8

[8] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. *ICCV*, 2017. 2, 4

[9] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2017. 6, 7

[10] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *PAMI*, 2014. 1, 3

[11] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 7

[12] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS*, 2014. 2, 3, 5

[13] M. Huh, P. Agrawal, and A. A. Efros. What makes imagenet good for transfer learning? In *NIPS LSCVS Workshop*, 2016. 6

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[15] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015. 2

[16] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. In *ICLR*, 2016. 5, 6, 7

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012. 1, 2, 6, 7

[18] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016. 2

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 7

[20] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016. 1, 2

[21] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 1, 2, 3, 4, 6, 7

[22] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *arXiv preprint arXiv:1603.09246*, 2016. 7

[23] M. Noroozi, H. Pirsiavash, and P. Favaro. Represenation learning by learning to count. In *ICCV*, 2017. 1, 2, 6, 7

[24] D. Okanohara and J. Tsuji. A discriminative language model with pseudo-negative samples. In *ACL*, 2007. 4

[25] A. Owens, J. Wu, J. H. M. annd William T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016. 2, 7

[26] D. Pathak, R. Girshick, P. Dollr, T. Darrell, and B. Hariharan. Learning features by watching objects move. *arXiv preprint arXiv:1612.06370*, 2016. 2, 6

[27] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2, 6, 7

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 7

[29] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation. *PAMI*, 2014. 1

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognitoin. In *ICLR*, 2015. 2

[31] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2006. 2

[32] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 1, 2, 6, 7

[33] X. Wang, K. He, and A. Gupta. Transitive invariance for self-supervised visual representation learning. In *ICCV*, 2017. 1, 2, 4

[34] Y. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016. 3

[35] Y.-X. Wang, L. Gui, and M. Hebert. Few-shot hash learning for image retrieval. In *ICCVW*, 2017. 3

[36] Y.-X. Wang and M. Hebert. Learning from small sample sets by combining unsupervised meta-training with cnns. In *NIPS*, 2016. 3

[37] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017. 3, 5, 6

[38] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. 5

[39] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016. 1, 2, 6, 7

[40] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2016. 6, 7

[41] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 7