Boosting Self-Supervised Learning via Knowledge Transfer

(2018)

Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro and Hamed Pirsiavash

Notes

Contributions

In a self-supervised setting, if we'd like to use a small model in a downstream task, we also end up being forced to use the same model to learn the pretext task. And this may limit the representation learning capabilities of the model and how much complicated the pretext task can be. To solve this problem the authors proposed a way to transfer knowledge from a a bigger model used in self supervision, to a smaller model in the downstream task.

Knowledge Transfer. First, for all the unlabeled images, their features are computed using a model pretrained in a self supervised setting. Second, the images are clustered based on their features using K-means, this way we create pseudo labels for each images (the cluster each image belongs to), this is supposed to give good results, given that good labels group semantically similar images together. Finally, we pretrain a smaller network (e.g. AlexNet) using the pseudo labels and use it as a starting for fine tuning the network for a downstream task.

Given a deep model (e.g. VGG16), the authors proposes a more complicated version of Jigsaw puzzle pretext task by adding occlusions, called Jigsaw++ where one of the nine patches is replaced by random sampled patch.

Method

Transferring Knowledge

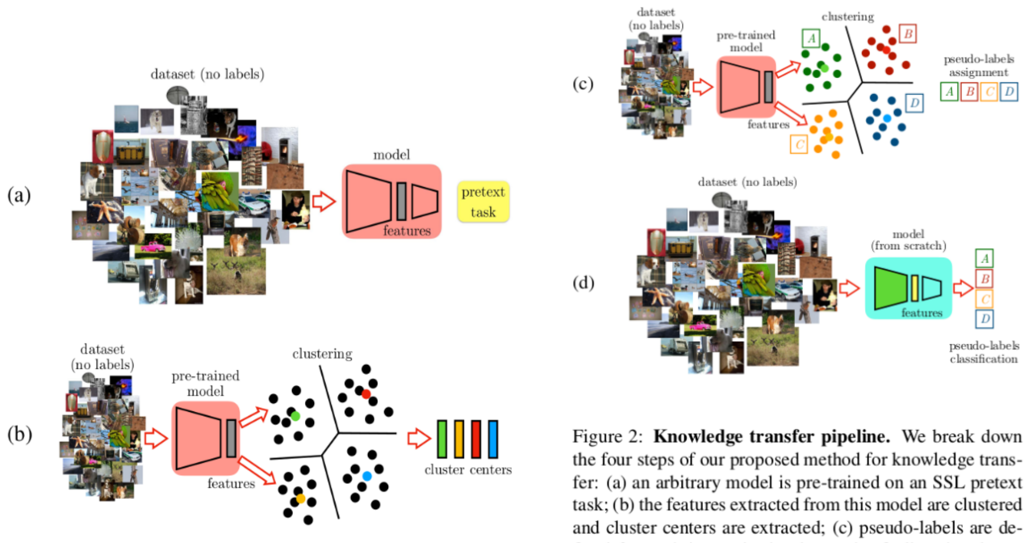

Figure 2: **Knowledge transfer pipeline.** We break down the four steps of our proposed method for knowledge transfer: (a) an arbitrary model is pre-trained on an SSL pretext task; (b) the features extracted from this model are clustered and cluster centers are extracted; (c) pseudo-labels are defined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

Based on the intuition that in the space of a good visual representation, semantically similar data points should be close to each other, the idea is to perform a simple clustering in the feature space and to obtain cluster assignments of each image in the dataset as pseudo labels. We can then train a smaller network with these pseudo labels as the targets. This method involves four steps: (1) Self-Supervised Learning Pre-Training using Jigsaw ++. (2) Clustering the images in the features spaces (3) Extracting Pseudo-Labels, which are the cluster assignments for each image. (d) Cluster Classification: to transfer knowledge from a bigger network that was used in self supervised setting, we train a smaller classification to predict the pseudo labels as the targets.

Jigsaw++ Pretext Task

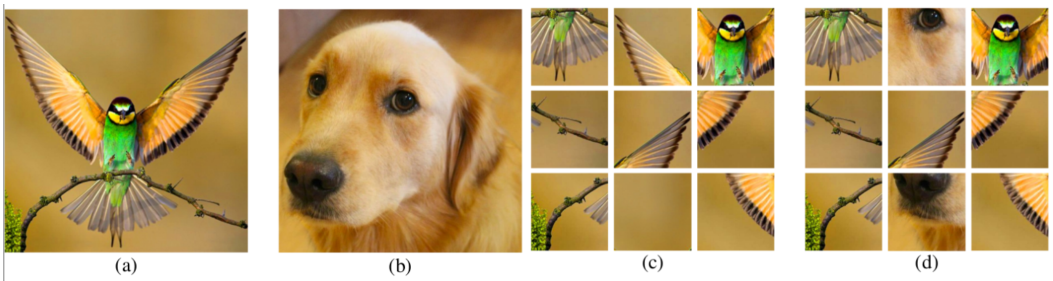

Figure 3: **The Jigsaw++ task.** (a) the main image. (b) a random image. (c) a puzzle from the original formulatio of [21], where all tiles come from the same image. (d) a puzzle in the Jigsaw++ task, where at most 2 tiles can com from a random image.

In the original pretext task, the goal is to find a reordering of tiles from a 3 x 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. 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 adding no ambiguities are added to the task, similar permutations are removed 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 the training setting.

Implementation details

All the features for all the methods are extracted from conv4 layer and max-pooled to 5 5 384 in the case of AlexNet and 4 4 512 in the case of VGG16, with k = 2K for k-means.

Results

Comparing Jigsaw++ trained with VGG16 and AlexNet, Context prediction, using HOG features to find the clusters, random clusters using a randomly initialized network and Knowledge distillation where we train a a student network to regress conv4-5 layers of the teacher network (both AlexNet).

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**

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**

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**