Self-supervised Jigsaw Puzzles Tasks

1. CFN

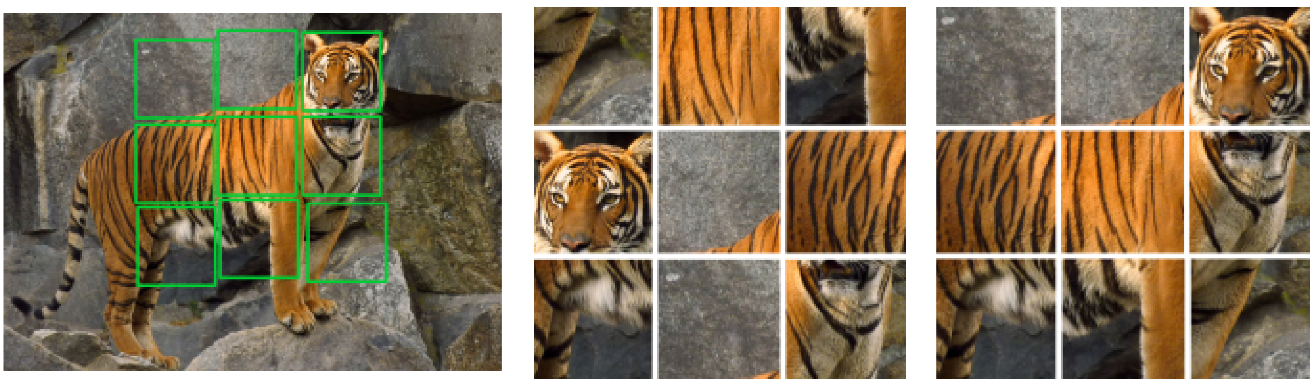
【Paper】Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles

【Date】ECCV2016

【Method】 context-free network (CFN)

【Introdunction】

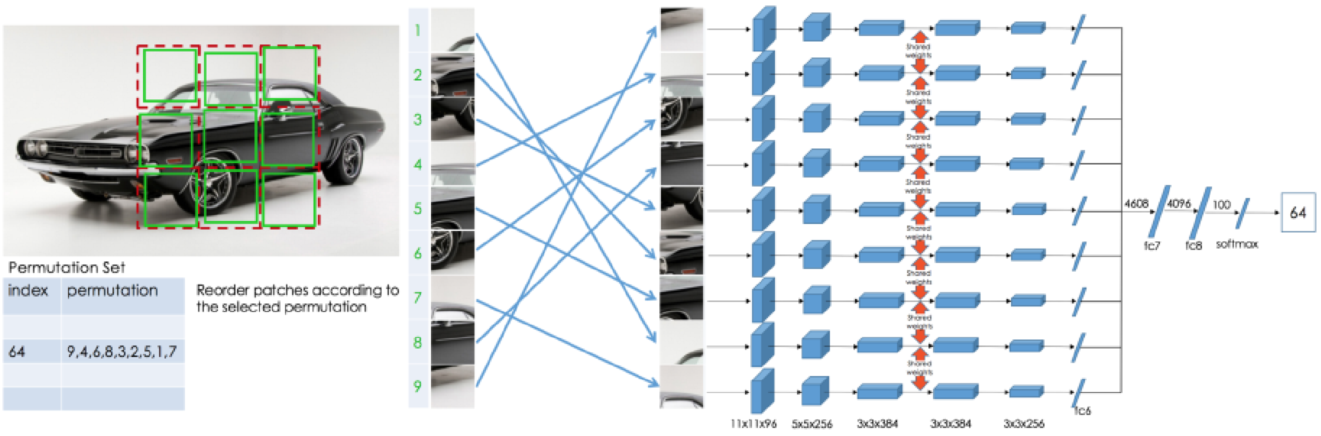
This paper proposed a novel self-supervised task called **Jigsaw puzzles**. The Jigsaw puzzles problem aims at arranging the shuffled image patches in the right order. Solving Jigsaw puzzles can help the network learn some high-level, global spatial correlations.



In this paper, the network is trained to solve Jigsaw puzzles as a pretext task. The learned features are then transferred to target tasks such as classification, detection and segmentation by fine-tuning.

【Method】

The CFN network architecture used for Jigsaw puzzles tasks is



The network is required to recognize the index of the correct permutation order from a permutation set.

CFN uses AlexNet as a backbone. It is a siamese network with nine branches. Each image patch is fed into a branch independently. The parameters are shared until fc6 (including fc6). Then the output features of fc6 are concatenated and fed into fc7.

CFN has fewer parameters than AlexNet. Because the feature map size output from conv5 and the number of hidden units in fc6 is smaller.

A 225×225 pixel window is randomly cropped from an image. The cropped image is divided into a 3×3 grid. Then a 64×64 pixel tile is randomly picked from each 75×75 pixel cell.

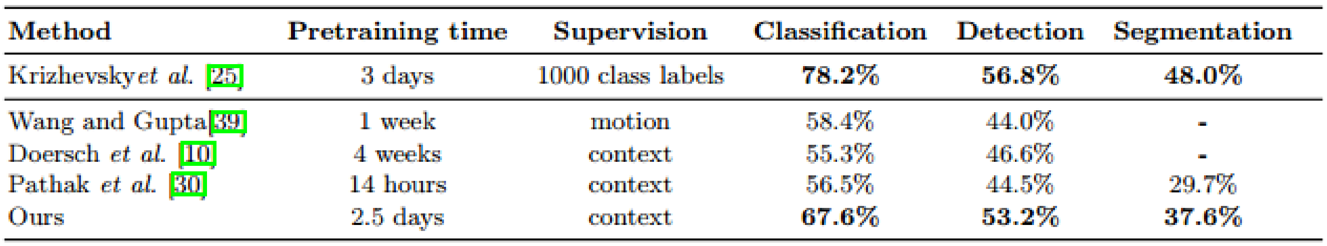
【Tricks】

CFN can take "shortcuts" to solve Jigsaw puzzles. In order to prevent the network from learning some superficial information, several tricks are used.

- Multiple permutations: Using different permutations for training can prevent the network overfits on specific locations. The premutation set is a hyperparameter that needs to be carefully chosen.
- Random gap: leaving random gap between the tiles can prevent the model exploiting information like edge continuity and pixel intensity.
- Independent normalization: Normalizing the mean and the standard deviation of each patch independently can prevent the model from finding similar low-level statistics between adjacent patches.
- Color jittering: Jittering the color channels and using grayscale images can prevent the model learning from chromatic aberration.

【Experiments】

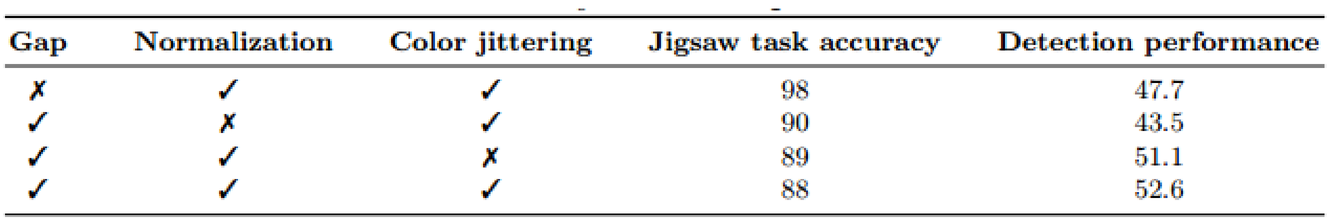
1) Transfer pre-trained Jigsaw features to target tasks



Method	**Pretraining time**	**Supervision**	**Classification**	**Detection**	**Segmentation**
Krizhevsky*et al.* [25]	3 days	1000 class labels	**78.2%**	**56.8%**	**48.0%**
Wang and Gupta[39]	1 week	motion	58.4%	44.0%	-
Doersch *et al.* [10]	4 weeks	context	55.3%	46.6%	-
Pathak *et al.* [30]	14 hours	context	56.5%	44.5%	29.7%
Ours	2.5 days	context	**67.6%**	**53.2%**	**37.6%**

Conv5 feature of CFN is transferred. CFN can close the gap with supervised AlexNet.

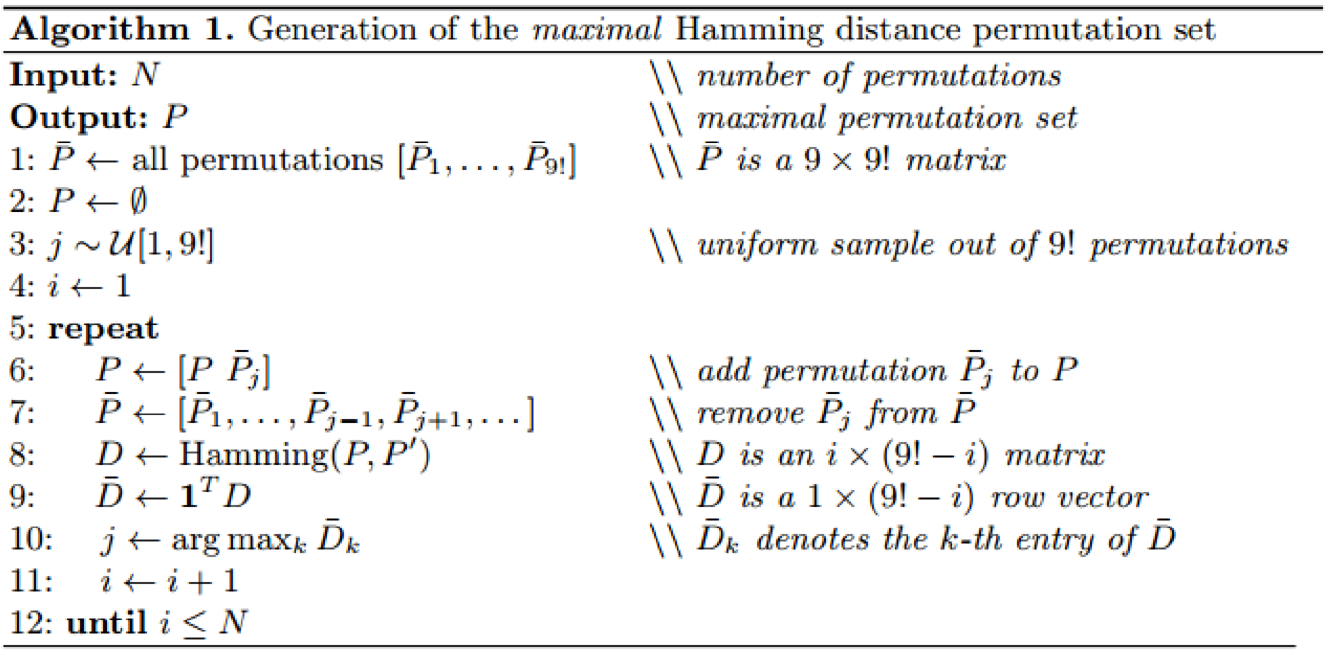
2) Ablation studies of tricks



Gap	**Normalization**	**Color jittering**	**Jigsaw task accuracy**	**Detection performance**
✗	✓	✓	98	47.7
✓	✗	✓	90	43.5
✓	✓	✗	89	51.1
✓	✓	✓	88	52.6

3) Choice of permutation set

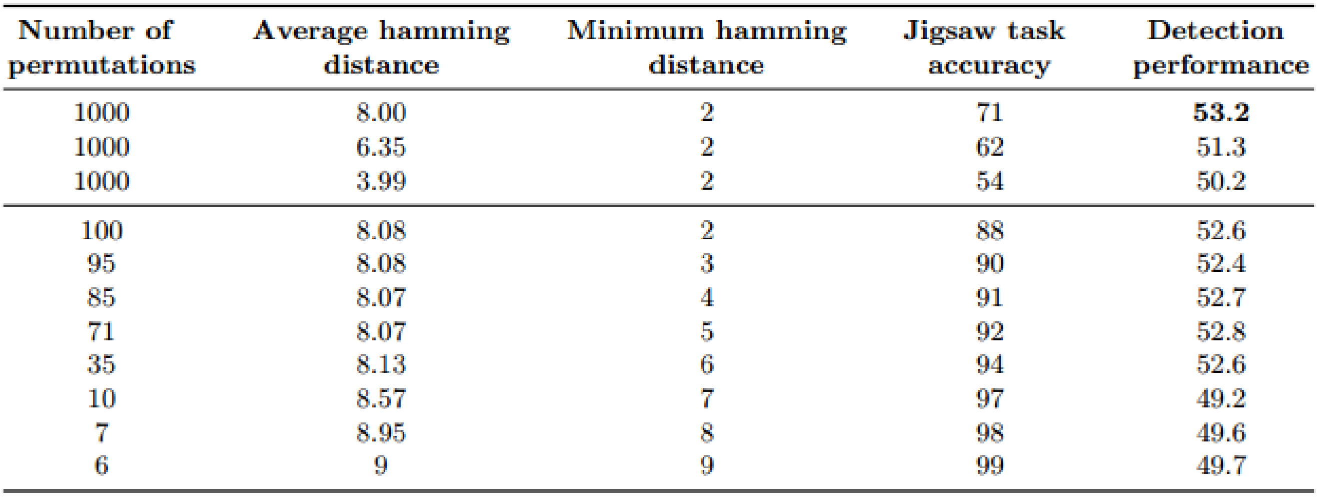
The authors use *maximal Hamming distance* to choose permutation set. The algorithm is



Algorithm 1. Generation of the *maximal* Hamming distance permutation set

```
Input: $N$                                          \\ number of permutations
Output: $P$                                         \\ maximal permutation set
1: $\bar{P} \leftarrow$ all permutations $[\bar{P}_1, \dots, \bar{P}_{9!}]$     \\ $\bar{P}$ is a $9 \times 9!$ matrix
2: $P \leftarrow \emptyset$
3: $j \sim \mathcal{U}[1, 9!]$                      \\ uniform sample out of $9!$ permutations
4: $i \leftarrow 1$
5: repeat
6:     $P \leftarrow [P\ \bar{P}_j]$                \\ add permutation $\bar{P}_j$ to $P$
7:     $\bar{P} \leftarrow [\bar{P}_1, \dots, \bar{P}_{j-1}, \bar{P}_{j+1}, \dots]$   \\ remove $\bar{P}_j$ from $\bar{P}$
8:     $D \leftarrow \text{Hamming}(P, P')$         \\ $D$ is an $i \times (9! - i)$ matrix
9:     $\bar{D} \leftarrow \mathbf{1}^T D$          \\ $\bar{D}$ is a $1 \times (9! - i)$ row vector
10:    $j \leftarrow \arg\max_k \bar{D}_k$          \\ $\bar{D}_k$ denotes the $k$-th entry of $\bar{D}$
11:    $i \leftarrow i + 1$
12: until $i \leq N$
```

The results of different settings of permutation set are



Number of permutations	Average hamming distance	Minimum hamming distance	Jigsaw task accuracy	Detection performance
1000	8.00	2	71	**53.2**
1000	6.35	2	62	51.3
1000	3.99	2	54	50.2
100	8.08	2	88	52.6
95	8.08	3	90	52.4
85	8.07	4	91	52.7
71	8.07	5	92	52.8
35	8.13	6	94	52.6
10	8.57	7	97	49.2
7	8.95	8	98	49.6
6	9	9	99	49.7

Two conclusions can be drawn:

- As the total number of permutations increases, the accuracy of Jigsaw task deceases because of the increasing difficulty, while the accuracy of detection increases.
- As the average Hamming distance increases, both the accuracy of Jigsaw task and detection increases. (A larger value of the average Hamming distance will induce a easier Jigsaw task because the permutations are very different.)

2. Knowledge Transfer

【**Paper**】Boosting Self-Supervised Learning via Knowledge Transfer

【**Date**】CVPR 2018

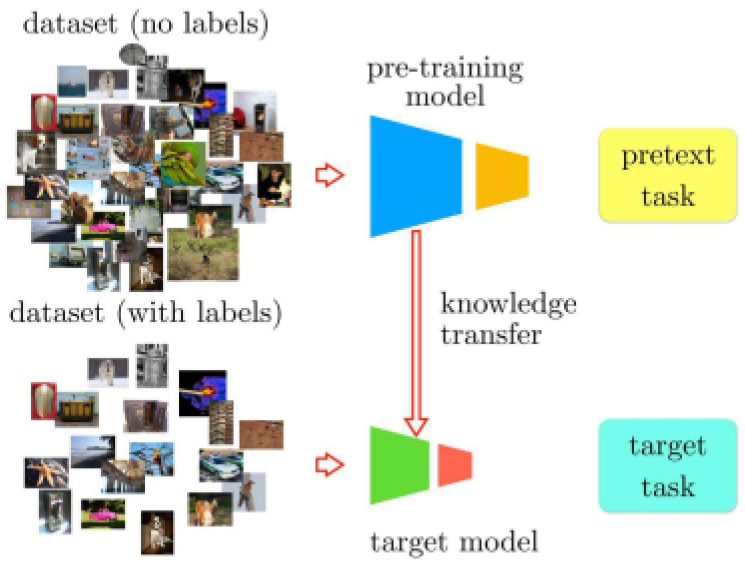
【**Method**】knowledge transfer

【**Introduction**】

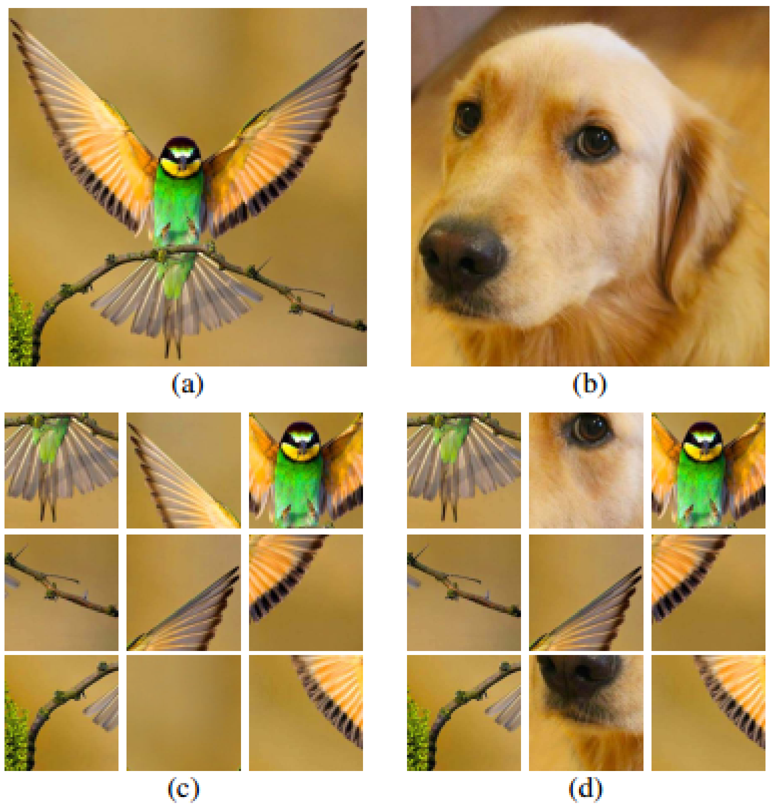
The definition of self-supervised learning:

> 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.

The general pipeline of self-supervised learning

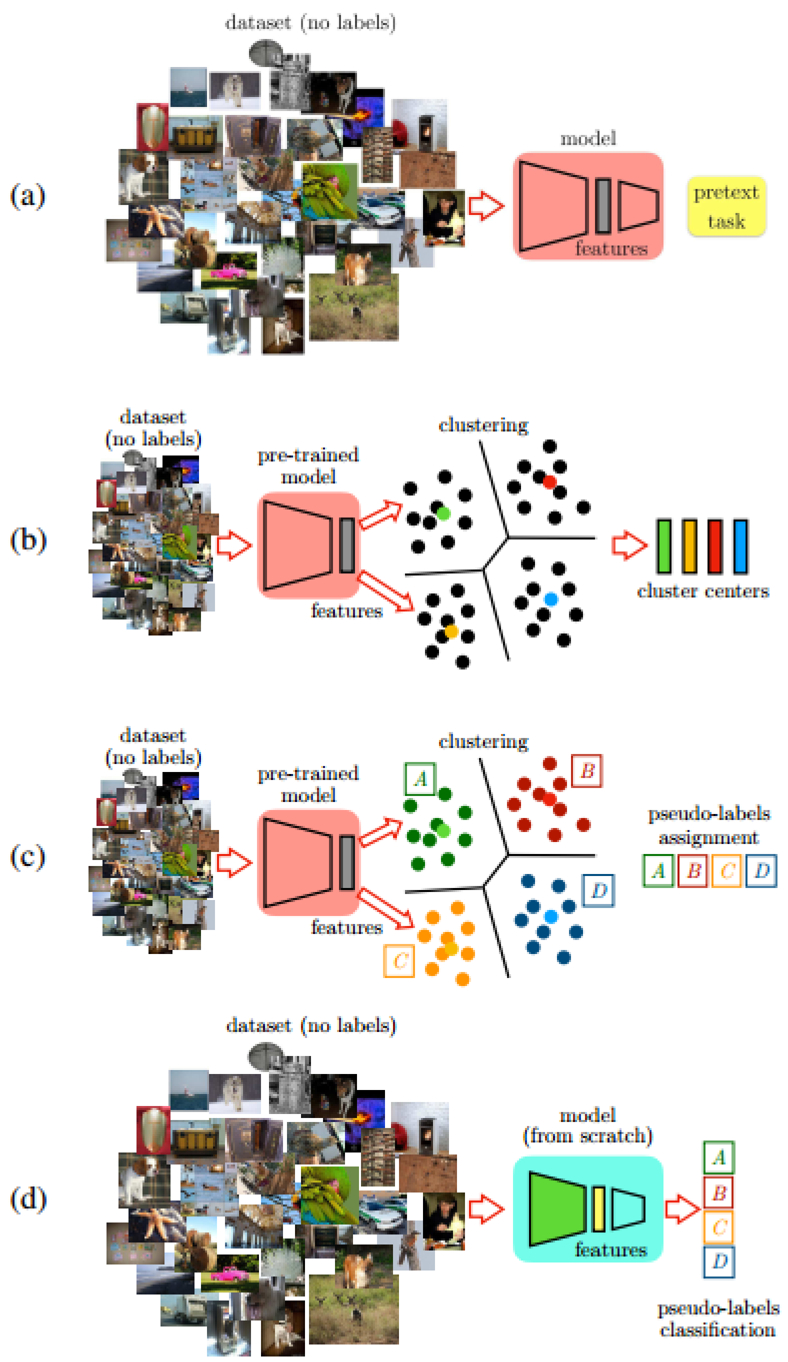


The authors re-define the Jigsaw puzzle tasks with a more difficult setting. The new task is called **Jigsaw++**. In Jigsaw++, a random number of tiles in the grid (up to 2) are replaced with tiles from another random image. The network needs to detect the occluding tiles and then reorders the remaining tiles.



【Method】

The method proposed in this paper includes four steps:



Step 1: Train the model to solve the Jigsaw++ task on a large-scale unlabeled dataset;

Step 2: Extract conv features from the pre-trained model and then do clustering on the extracted feature;

Step 3: Assign pseudo labels to the instances based on each cluster;

Step 4: Train the target model on the dataset with pseudo labels.

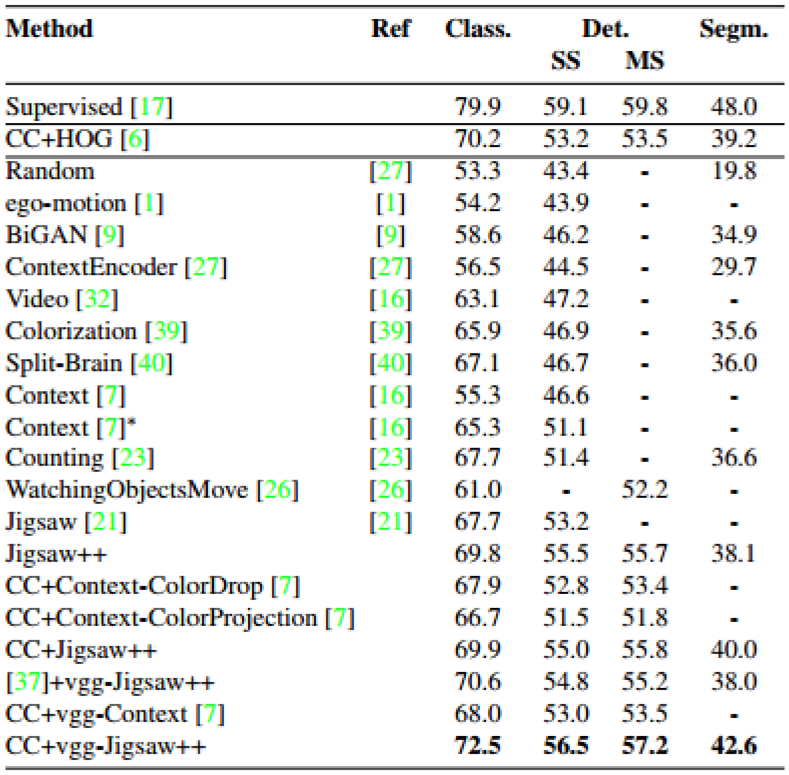
Note that the datasets used in Step 1, 2, 3 can be different.

In this method, the pre-training and fine-tuning process are separated. It means that one can use a deeper model to solve a tougher pre-text task and then transfer the knowledge of the pre-trained model to a shallower target model to solve the target task. Unlike previous methods, the pre-trained model and the target model can be different.

The idea of knowledge transfer is based on the intuition that semantically similar instances should be close to each other in a good feature space. In other words, knowledge transfer aims at mining the similarity between instances in the feature-level.

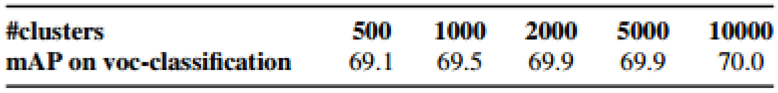
【Experiment】

The authors are interested in how far the performance of AlexNet on PASCAL can be improved by self-supervised pre-training. The conclusion is that training VGG16 with the Jigsaw++ task and then transferring to a target AlexNet can lead to a significant improvement.



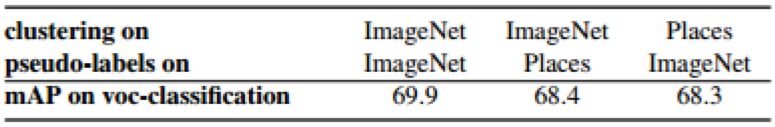
Method	Ref	Class.	Det.		Segm.
			SS	MS	
Supervised [17]		79.9	59.1	59.8	48.0
CC+HOG [6]		70.2	53.2	53.5	39.2
Random	[27]	53.3	43.4	-	19.8
ego-motion [1]	[1]	54.2	43.9	-	-
BiGAN [9]	[9]	58.6	46.2	-	34.9
ContextEncoder [27]	[27]	56.5	44.5	-	29.7
Video [32]	[16]	63.1	47.2	-	-
Colorization [39]	[39]	65.9	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**

Moreover, the method is not sensitive to the number of clusters



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

or using different datasets in different steps (2&3).



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