Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles

Mehdi Noroozi and Paolo Favaro

Institute for Informatiks
University of Bern
{noroozi,paolo.favaro}@inf.unibe.ch

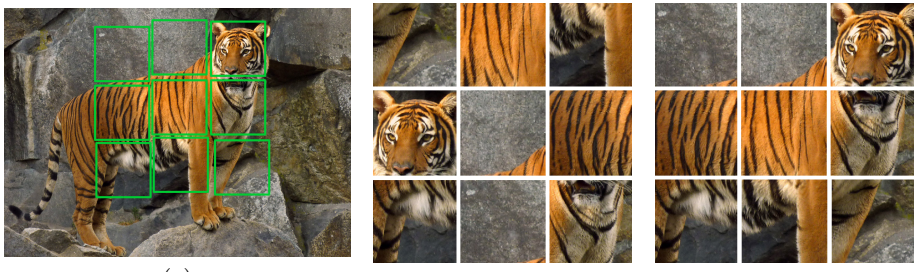
Abstract. In this paper we study the problem of image representation learning without human annotation. By following the principles of self-supervision, we build a convolutional neural network (CNN) that can be trained to solve Jigsaw puzzles as a *pretext* task, which requires no manual labeling, and then later repurposed to solve object classification and detection. To maintain the compatibility across tasks we introduce the *context-free network* (CFN), a siamese-ennead CNN. The CFN takes image tiles as input and explicitly limits the receptive field (or context) of its early processing units to one tile at a time. We show that the CFN includes fewer parameters than AlexNet while preserving the same semantic learning capabilities. By training the CFN to solve Jigsaw puzzles, we learn both a feature mapping of object parts as well as their correct spatial arrangement. Our experimental evaluations show that the learned features capture semantically relevant content. Our proposed method for learning visual representations outperforms state of the art methods in several transfer learning benchmarks.

1 Introduction

Visual tasks, such as object classification and detection, have been successfully approached through the supervised learning paradigm [1,11,25,36], where one uses labeled data to train a parametric model. However, as manually labeled data can be costly, unsupervised learning methods are gaining momentum.

Recently, Doersch *et al.* [10], Wang and Gupta [39] and Agrawal *et al.* [2] have explored a novel paradigm for unsupervised learning called *self-supervised learning*. The main idea is to exploit different labelings that are freely available besides or within visual data, and to use them as intrinsic reward signals to learn general-purpose features. [10] uses the relative spatial co-location of patches in images as a label. [39] uses object correspondence obtained through tracking in videos, and [2] uses ego-motion information obtained by a mobile agent such as the Google car [7]. The features obtained with these approaches have been successfully transferred to classification and detections tasks, and their performance is very encouraging when compared to features trained in a supervised manner.

A fundamental difference between [10] and [39,2] is that the former method uses single images as the training set and the other two methods exploit multiple images related either through a temporal or a viewpoint transformation.



(a) (b) (c)

Fig. 1: Learning image representations by solving Jigsaw puzzles. (a) The image from which the tiles (marked with green lines) are extracted. (b) A puzzle obtained by shuffling the tiles. Some tiles might be directly identifiable as object parts, but others are ambiguous (*e.g.*, have similar patterns) and their identification is much more reliable when all tiles are jointly evaluated. In contrast, with reference to (c), determining the relative position between the central tile and the top two tiles from the left can be very challenging [10].

While it is true that biological agents typically make use of multiple images and also integrate additional sensory information, such as ego-motion, it is also true that single snapshots may carry more information than we have been able to extract so far. This work shows that this is indeed the case. We introduce a novel self-supervised task, the *Jigsaw puzzle reassembly* problem (see Fig. 1), which builds features that yield high performance when transferred to detection and classification tasks.

We argue that solving Jigsaw puzzles can be used to teach a system that an object is made of parts and what these parts are. The association of each separate puzzle tile to a precise object part might be ambiguous. However, when all the tiles are observed, the ambiguities might be eliminated more easily because the tile placement is mutually exclusive. This argument is supported by our experimental validation. Training a Jigsaw puzzle solver takes about 2.5 days compared to 4 weeks of [10]. Also, there is no need to handle chromatic aberration or to build robustness to pixelation. Moreover, the features are highly transferrable to detection and classification and yield the highest performance to date for an unsupervised method.

2 Related work

This work falls in the area of *representation/feature learning*, which is an unsupervised learning problem [3]. Representation learning is concerned with building intermediate representations of data useful to solve machine learning tasks. It also involves *transfer learning* [41], as one applies and repurposes features that have been learned by solving the Jigsaw puzzle to other tasks such as object classification and detection. In our experiments we do so via the *pre-training +*

fine-tuning scheme, as in prior work [2]. Pre-training corresponds to the feature learning that we obtain with our Jigsaw puzzle solver. Fine-tuning is instead the process of updating the weights obtained during pre-training to solve another task (object classification or detection).

Unsupervised Learning. There is a rich literature in unsupervised learning of visual representations [5]. Most techniques build representations by exploiting general-purpose priors such as smoothness, sharing of factors, factors organized hierarchically, belonging to a low-dimension manifold, temporal and spatial coherence, and sparsity. Unfortunately, a general criterion to design a visual representation is not available. Nonetheless, a natural choice is the goal of disentangling factors of variation. For example, several factors such as the object shapes, the object materials, and the light sources, combine to create complex effects such as shadows, shading, color patterns and reflections in images. Ideal features would separate each of these factors so that other learning tasks (*e.g.*, classification based on just shape or surface materials) can be handled more easily. In this work we design features to separate the appearance from the arrangement (geometry) of parts of objects.

Because of the relevance to contemporary research and to this work, we discuss mainly methods in deep learning. In general one can group unsupervised learning methods into: probabilistic, direct mapping (autoencoders), and manifold learning ones. Probabilistic methods divide variables of a network into observed and latent ones. Learning is then associated with determining model parameters that maximize the likelihood of the latent variables given the observations. A family of popular probabilistic models is the *Restricted Boltzmann Machine* (RBM) [37,18], which makes training tractable by imposing a bipartite graph between latent and observed variables. Unfortunately, these models become intractable when multiple layers are present and are not designed to produce features in an efficient manner. The direct mapping approach focuses on the latter aspect and is typically built via *autoencoders* [6,19,29]. Autoencoders specify explicitly the feature extraction function (encoder) in a parametric form as well as the mapping from the feature back to the input (decoder). These direct mappings are trained by minimizing the reconstruction error between the input and the output produced by the autoencoder (obtained by applying the encoder and decoder sequentially). A remarkable example of a very large scale autoencoder is the work of Le *et al.* [26]. Their results showed that robust human and cat faces as well as human body detectors could be built without human labeling.

If the data structure suggests that data points might concentrate around a manifold, then *manifold learning* techniques can be employed [34,4]. This representation allows to map directly smooth variations of the factors to smooth variations of the observations. Some of the issues with manifold learning techniques are that they might require computing nearest neighbors (which scales quadratically with the number of samples) and that they need a sufficiently high density of samples around the manifold (and this becomes more difficult to achieve with high-dimensional manifolds).

In the context of Computer Vision, it is worth mentioning some early work on unsupervised learning of models for classification. For instance [13,40] introduced methods to build a probabilistic representation of objects as constellations of parts. A limitation is the high computational complexity of these models. As we will see later, training the Jigsaw puzzle solver also amounts to building a model of both appearance and configuration of the parts.

Self-supervised Learning. This learning strategy is a recent variation on the unsupervised learning theme that exploits labeling that comes for "free" with the data [10,39,2]. We make a distinction between labels that are easily accessible and are associated with a non-visual signal (for example, ego-motion [2], but also one could consider audio, text and so on), and labels that are obtained from the structure of the data [10,39]. Our work relates to the latter case as we simply re-use the input images and exploit the pixel arrangement as a label.

Doersch *et al.* [10] train a convolutional network to classify the relative position between two image patches. One tile is kept in the middle of a 3×3 grid and the other tile can be placed in any of the other 8 available locations (up to some small random shift). In Fig. 1 (c) we show an example where the relative location between the central tile and the top-left and top-middle tiles is ambiguous. In contrast, the Jigsaw puzzle problem is solved by observing all the tiles at the same time. This allows the trained network to intersect all ambiguity sets and possibly reduce them to a singleton.

The method of Wang and Gupta [39] builds a metric to define similarity between patches. Three patches are used as input, where two patches are matched via tracking in a video and the third one is arbitrarily chosen. The main advantage of this method is that labeling requires just using a tracking method (they use SURF interest points to detect initial bounding boxes and then tracking via the KCF method [17]). The matched patches will have intraclass variability due to changes in illumination, occlusion, viewpoint, pose, occlusions, and clutter factors. However, because the underlying object is the same, the estimated features may not necessarily cluster patches with two different instances of the same object (*i.e.*, based on their semantic content). The method proposed by Agrawal *et al.* [2] exploits labeling (egomotion) provided by other sensors. The advantage is that this labeling is freely available in most cases or is quite easy to obtain. They show that egomotion is a useful supervisory signal when learning features. They train a siamese network to estimate egomotion from two image frames and compare it to the egomotion measured with odometry sensors. The trained features will build an invariance similar to that of [39]. However, because the object identity is the same in both images, the intraclass variability may be limited. With two images of the same instance, learned features focus on their similarities (such as color and texture) rather than their high-level structure. In contrast, the Jigsaw puzzle approach ignores similarities between tiles (such as color and texture), as they do not help their localization, and focuses instead on their differences. In Fig. 2 we illustrate this concept with two examples: Two cars that have different colors and two dogs with different fur patterns. The features learned to solve puzzles in one (car/dog) image will apply also to the

Fig. 2: Most of the shape of these 2 pairs of images is the same (two separate instances within the same categories). However, some low-level statistics are different (color and texture). The Jigsaw puzzle solver learns to ignore such statistics when they do not help the localization of parts.

other (car/dog) image as they will be invariant to shared patterns. The ability of the Jigsaw puzzle solver to cluster together object parts can be seen in the top 16 activations shown in Fig. 4 and in the image retrieval samples in Fig. 5.

Jigsaw Puzzles. Jigsaw puzzles have been associated with learning since their inception. They were introduced in 1760 by John Spilsbury as a pretext to help children learn geography. The first puzzle was a map attached to a wooden board, which was then sawed in pieces corresponding to countries [38]. Studies in Psychonomic show that Jigsaw puzzles can be used to assess visuospatial processing in humans [33]. Indeed, the *Hooper Visual Organization Test* [20] is routinely used to measures an individual's ability to organize visual stimuli. This test uses puzzles with line drawings of simple objects and requires the patient to recognize the object without moving the tiles. Instead of using Jigsaw puzzles to assess someone's visuospatial processing ability, in this paper we propose to use Jigsaw puzzles to develop a visuospatial representation of objects in the context of CNNs.

There is also a sizeable literature on solving Jigsaw puzzles computationally (see, for example, [32,14,31]). However, these methods rely on the shape of the tiles or on texture especially in the proximity of the borders of the tiles. These are cues that we avoid when training the Jigsaw puzzle solver, as they do not carry useful information when learning a part detector.

3 Solving Jigsaw Puzzles

At the present time, the design of convolutional neural networks (CNN) is still an art that relies on extensive experience. Here we provide a brief discussion of how we arrived at a convolutional architecture capable of solving Jigsaw puzzles while learning general-purpose features.

An immediate approach to solve Jigsaw puzzles is to stack the tiles of the puzzle along the channels (*i.e.*, the input data would have $9 \times 3 = 27$ channels) and then correspondingly increase the depth of the filters of the first convolutional layer in AlexNet [25]. The problem with this design is that the network prefers to identify correlations between low-level texture statistics across tiles

rather than between the high-level primitives. Low-level statistics, such as similar structural patterns and texture close to the boundaries of the tile, are simple cues that humans actually use to solve Jigsaw puzzles. However, solving a Jigsaw puzzle based on these cues does not require any understanding of the global object. Thus, here we present a network that delays the computation of statistics across different tiles (see Fig. 3). The network first computes features based only on the pixels within each tile (one row in Fig. 3). Then, it finds the parts arrangement just by using these features (last fully connected layers in Fig. 3). The objective is to force the network to learn features that are as representative and discriminative as possible of each object part for the purpose of determining their relative location.

3.1 The Context-Free Architecture

We build a siamese-ennead convolutional network (see[1] Fig. 3), where each row up to the first fully connected layer (*fc6*) uses the AlexNet architecture [25] with shared weights. Similar schemes were used in prior work [10,39,2]. The outputs of all *fc6* layers are concatenated and given as input to $fc7$. All the layers in the rows share the same weights up to and including *fc6*.

We call this architecture the *context-free network* (CFN) because the data flow of each patch is explicitly separated until the fully connected layer and context is handled only in the last fully connected layers. We verify that this architecture performs as well as AlexNet in the classification task on the ImageNet 2012 dataset [8]. In this test we resize the input images to 225×225 pixels, split them into a 3×3 grid and then feed the full 75×75 tiles to the network. We find that the CFN achieves 57.1% top-1 accuracy while AlexNet achieves 57.4% top-1 accuracy. However, the CFN architecture is more compact than AlexNet. It depends on only 27.5M parameters, while AlexNet uses 61M parameters. The *fc6* layer includes $4 \times 4 \times 256 \times 512 \sim 2$M parameters while the *fc6* layer of AlexNet includes $6 \times 6 \times 256 \times 4096 \sim 37.5$M parameters. However, the *fc7* layer in our architecture includes 2M parameters more than the same layer in AlexNet.

This network can thus be used interchangeably for different tasks including detection and classification. In the next section we show how to train the CFN for the Jigsaw puzzle reassembly.

3.2 The Jigsaw Puzzle Task

To train the CFN we define a set of Jigsaw puzzle permutations, *e.g.*, a tile configuration $S = (3, 1, 2, 9, 5, 4, 8, 7, 6)$, and assign an index to each entry. We randomly pick one such permutation, rearrange the 9 input patches according to

[1] In earlier versions of this publication we reported transfer learning results where AlexNet had a stride 2 in the first convolutional layer as used during the training for the puzzle task. This arXiv version introduces new updated results. See Fig. 3 caption for more information and the Experiments section.

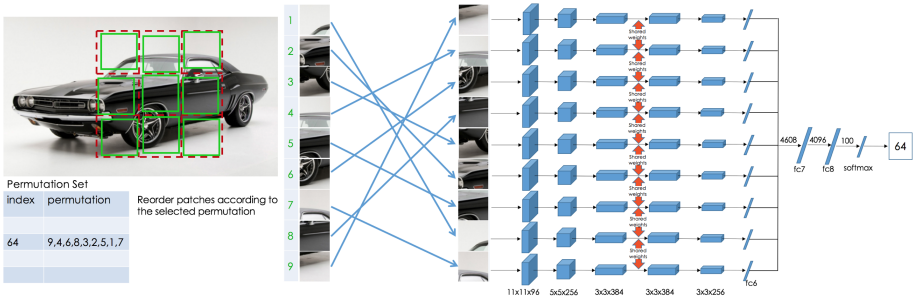


Fig. 3: **Context Free Network.** The figure illustrates how a puzzle is generated and solved. We randomly crop a 225×225 pixel window from an image (red dashed box), divide it into a 3×3 grid, and randomly pick a 64×64 pixel tiles from each 75×75 pixel cell. These 9 tiles are reordered via a randomly chosen permutation from a predefined permutation set and are then fed to the CFN. The task is to predict the index of the chosen permutation (technically, we define as output a probability vector with 1 at the 64-th location and 0 elsewhere). The CFN is a siamese-ennead CNN. For simplicity, we do not indicate the max-pooling and ReLU layers. These shared layers are implemented exactly as in AlexNet [25]. **In the transfer learning experiments we show results with the trained weights transferred on AlexNet (precisely, stride 4 on the first layer). The training in the transfer learning experiment is the same as in the other competing methods. Notice instead, that during the training on the puzzle task, we set the stride of the first layer of the CFN to 2 instead of 4.**

that permutation, and ask the CFN to return a vector with the probability value for each index. Given 9 tiles, there are $9! = 362{,}880$ possible permutations. From our experimental validation, we found that the permutation set is an important factor on the performance of the representation that the network learns. We perform an ablation study on the impact of the permutation set in subsection 4.2.

3.3 Training the CFN

The output of the CFN can be seen as the conditional probability density function (pdf) of the spatial arrangement of object parts (or scene parts) in a part-based model, *i.e.*,

$$p(S|A_1, A_2, \ldots, A_9) = p(S|F_1, F_2, \ldots, F_9) \prod_{i=1}^{9} p(F_i|A_i) \quad (1)$$

where S is the configuration of the tiles, A_i is the i-th part appearance of the object, and $\{F_i\}_{i=1,\ldots,9}$ form the intermediate feature representation. Our objective is to train the CFN so that the features F_i have semantic attributes that can identify the relative position between parts.

Given the limited amount of data that we can use to build an approximation of this very high-dimensional pdf, close attention must be paid to the training strategy. One problem is when the CFN learns to associate each appearance A_i to an absolute position. **In this case, the features F_i would carry no semantic meaning, but just information about an arbitrary 2D position**. This problem could happen if we generate just 1 Jigsaw puzzle per image. Then, the CFN could learn to cluster patches only based on their absolute position in the puzzle, and not on their textural/structural content. If we write the configuration S as a list of tile positions $S = (L_1, \dots, L_9)$ then in this case the conditional pdf $p(S|F_1, F_2, \dots, F_9)$ would factorize into independent terms

$$p(L_1, \dots, L_9|F_1, F_2, \dots, F_9) = \prod_{i=1}^{9} p(L_i|F_i) \tag{2}$$

where each tile location L_i is fully determined by the corresponding feature F_i.

More in general, a self-supervised learning system might lead to representations that are suitable to solve the pre-text task, but not the target tasks, *e.g.*, object classification, detection, and segmentation. In this regard, an important factor to learn better representations is to prevent our model from taking these undesirable solutions, such as the one just described above, to solve the pre-text task. We call these solutions *shortcuts*. Other shortcuts that the model can use to solve the Jigsaw puzzle task include exploiting low-level statistics, such as edge continuity, the pixel intensity/color distribution, and chromatic aberration.

To avoid shortcuts we employ multiple techniques. To prevent mapping the appearance to an absolute position we feed multiple Jigsaw puzzles of the same image to the CFN (an average of 69 out of 1000 possible puzzle configurations) and make sure that the tiles are shuffled as much as possible by choosing configurations with sufficiently large average Hamming distance. In this way the same tile would have to be assigned to multiple positions (possibly all 9) thus making the mapping of features F_i to any absolute position equally likely. To avoid shortcuts due to edge continuity and pixel intensity distribution we also leave a random gap between the tiles. This discourages the CFN from learning low-level statistics and was also done in [10]. During training we resize each input image until either the height or the width matches 256 pixels and preserve the original aspect ratio. Then, we crop a random region from the resized image of size 225×225 and split it into a 3×3 grid of 75×75 pixels tiles. We then extract a 64×64 region from each tile by introducing random shifts and feed them to the network. Thus, we have an average gap of 11 pixels between the tiles. However, the gaps may range from a minimum of 0 pixels to a maximum of 22 pixels. To avoid shortcuts due to chromatic aberration we jitter the color channels and use grayscale images (see more details in the Experiments section). In subsection 4.2 we perform ablation studies on the techniques we use to prevent the shortcuts.

We used Caffe [23] and modified the code to choose random image patches and permutations during the training time. This allowed us to keep the dataset small (1.3M images from ImageNet) and the training efficient, while the CFN

could see an average of 69 different puzzles per image (that is about 90M different Jigsaw puzzles).

3.4 Implementation Details

We use stochastic gradient descent without batch normalization [21] on one Titan X GPU. The training uses 1.3M color images of 256×256 pixels and mini-batches with a batch size of 256 images. The images are resized by preserving the aspect ratio until either the height or the width matches 256 pixels. Then the other dimension is cropped to 256 pixels. The training converges after 350K iterations with a basic learning rate of 0.01 and takes 59.5 hours in total (~ 2.5 days). If we take $122\% = \frac{3072\text{cores@1000Mhz}}{2880\text{cores@875Mhz}} = \frac{6{,}144\text{GFLOPS}}{5{,}040\text{GFLOPS}}$ as the best possible performance ratio between the Titan X and the Tesla K40 (used for [10]) we can predict that the CFN would have taken ~ 72.5 hours (~ 3 days) on a Tesla K40. We compute that on average each image is used $350K \times 256/1.3M \simeq 69$ times. That is, we solve on average 69 Jigsaw puzzles per image.

4 Experiments

We first evaluate the performance of our learned representations on different transfer learning benchmarks. We then perform ablation studies on our proposed method. We also visualize the neurons of the intermediate layers of our network. Finally, we compare our features with those of [10,39] both qualitatively and quantitatively on image retrieval.

4.1 Transfer Learning

We evaluate our learned features as pre-trained weights for classification, detection, and semantic segmentation tasks on the PASCAL VOC dataset[12]. We also introduce a novel benchmark to evaluate methods for unsupervised/self-supervised representation learning. After training the CFN on the self-supervised learning task, we use the CFN weights to initialize all the `conv` layers of a standard AlexNet network (stride 4 on the first layer). Then, we retrain the rest of the network from scratch (Gaussian noise as initial weights) for object classification on ImageNet dataset. Notice that while during the training of the Jigsaw task we use stride 2 in the first layer of our CFN, we use a standard AlexNet (stride 4 on the first layer) to make the comparison with competing methods in all the experiments directly comparable.

Pascal VOC We fine-tune the Jigsaw task features on the classification task on PASCAL VOC 2007 by using the framework of Krähenbühl *et al.* [24] and on the object detection task by using the Fast R-CNN [16] framework. We also fine-tune our weights for the semantic segmentation task using the framework [27] on the PASCAL VOC 2012 dataset. Because our fully connected layers are

Table 1: Results on PASCAL VOC 2007 Detection and Classification. The results of the other methods are taken from Pathak *et al.* [30].

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%**

different from those of the standard AlexNet, we select one row of the CFN (up to `conv5`), copy only the weights of the convolutional layers, and fill the fully connected layers with Gaussian random weights with mean 0.1 and standard deviation 0.001. The results are summarized in Table 1.

Our features achieve 53.2% mAP using multi-scale training and testing, 67.6% in classification, and 37.6% in semantic segmentation thus outperforming all other methods and closing the gap with features obtained with supervision.

ImageNet Classification Yosinski *et al.* [41] have shown that the last layers of AlexNet are specific to the task and dataset used for training, while the first layers are general-purpose. In the context of transfer learning, this transition from general-purpose to task-specific determines where in the network one should extract the features. In this section we try to understand where this transition occurs in our learned representation. We repurpose our weights, [10], and [39] to the classification task on the ImageNet 2012 dataset [8]. Table 2 summarizes the results. The analysis consists of training each network with the labeled data from ImageNet 2012 by locking a subset of the layers and by initializing the unlocked layers with random values. If we train AlexNet, we obtain the reference maximum accuracy of 57.4%. Our method achieves 34.6% when only fully connected layers are trained. There is a significant improvement (from 34.6% to 45.3%) when the `conv5` layer is also trained. This shows that the `conv5` layer starts to be specialized on the Jigsaw puzzle reassembly task.

We also perform a novel experiment to understand whether semantic classification is useful to solve Jigsaw puzzles, and thus to see how much object classification and Jigsaw puzzle reassembly tasks are related. We take the pretrained AlexNet and transfer its features to solve Jigsaw puzzles. We also use the same locking scheme to see the transferability of features at different layers. The performance is shown in Table 3. Compared to the maximum accuracy of the Jigsaw task, 88%, we can see that semantic training is quite helpful towards recognizing object parts. Indeed, the performance is very high up to `conv4`.

4.2 Ablation Studies

We perform ablation studies on our proposed methods to show the impact of each component during the training of Jigsaw task. We train under different

Table 2: Comparison of classification results on ImageNet 2012 [9]. The numbers are obtained by averaging 10 random crops predictions.

	🔒 conv1	🔒 conv2	🔒 conv3	🔒 conv4	🔒 conv5
CFN	**54.7**	**52.8**	**49.7**	45.3	**34.6**
Doersch *et al.* [10]	53.1	47.6	48.7	**45.6**	30.4
Wang and Gupta [39]	51.8	46.9	42.8	38.8	29.8
Random	48.5	41.0	34.8	27.1	12.0

Table 3: Transfer learning of AlexNet from a classification task to the Jigsaw puzzle reassembly problem. The j-th column indicates that all layers from `conv1` to `conv-`j were locked and all subsequent layers were randomly initialized and retrained. Notice how the first 4 layers provide very good features for solving puzzles. This shows that object classification and the Jigsaw puzzle problems are related.

	🔒 conv1	🔒 conv2	🔒 conv3	🔒 conv4	🔒 conv5
AlexNet [25]	88	87	86	83	74

scenarios and evaluate the performance on detection task on PASCAL VOC 2007.

Permutation Set. The permutation set controls the ambiguity of the task. If the permutations are close to each other, the Jigsaw puzzle task is more challenging and ambiguous. For example, if the difference between two different permutations lies only in the position of two tiles and there are two similar tiles in the image, the prediction of the right solution will be impossible. The challenge here is a weaker version of what happens in the method of Doersch *et al.* [10]. To show this issue quantitatively, we compare the performance of the learned representation on the PASCAL VOC 2007 detection task by generating several permutation sets based on the following three criteria:

I) Cardinality. We train the network with a different number of permutations and see what impact this has on the learned features. We find that as the total number of permutations increases, the training on the Jigsaw task becomes more and more difficult. Also, we find that the performance of the detection task increases with a growing number of permutations.

II) Average Hamming distance. We use a subset of 1000 permutations and select them based on their Hamming distance (*i.e.*, the number of different tile locations between 2 permutations S_1 and S_2). One can see that the average Hamming distance between permutations controls the difficulty of the Jigsaw puzzle

Table 4: Ablation study on the impact of the permutation set.

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

Table 5: Ablation study on the impact of the shortcuts.

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

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

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

reassembly task, and it also correlates with the object detection performance. We find that as the average Hamming distance increases, the CFN yields lower Jigsaw puzzle solving errors and lower object detection errors with fine-tuning. In the Experiments section we compare the performance on object detection of CFNs trained with 3 choices for the Hamming distance: minimal, average and maximal (see Table 4). From those tests we can see that large Hamming distances are desirable. We generate this permutation set iteratively via a greedy algorithm. We begin with an empty permutation set and at each iteration select the one that has the desired Hamming distance to the current permutation set.

Algorithm 1 provides more details about the algorithm. For the minimal and middle case, the $\arg\max_k$ function at line 10 is replaced by $\arg\min_k$ and uniform sampling respectively. Note that the permutation set is generated before training.

III) Minimum hamming distance. To increase the minimum possible distance between permutations, we remove similar permutations in a maximal set with 100 initial entries. As argued before, the minimum distance helps to make the task less ambiguous. The performance results showing the impact of each component are summarized in Table 4. The best performing permutation set is a trade off between the number of permutations and how dissimilar they are from each other.

The outcome of this ablation study seems to point to the following final consideration:

A good self-supervised task is neither simple nor ambiguous.

Preventing Shortcuts In a self-supervised learning method, *shortcuts* exploit information useful for solving the pre-text task, but not for a target task, such as detection. Similar to [10], we experimentally show that the CFN can take the following shortcuts to solve the Jigsaw Puzzle task:

Low level statistics. Adjacent patches include similar low-level statistics like the mean and standard deviation of the pixel intensities. This allows the model to find the arrangement of the patches. To avoid this shortcut, we normalize the mean and the standard deviation of each patch independently.

Edge continuity. A strong cue to solve Jigsaw puzzles is the continuity of edges. We select the 64×64 pixel tiles randomly from the 85×85 pixel cells. This allows us to have a 21 pixel gap between tiles.

Chromatic Aberration. Chromatic aberration is a relative spatial shift between color channels that increases from the images center to the borders. This type of distortion helps the network to estimate the tile positions. To avoid this shortcut, we use three techniques: i) We crop the central square of the original image and resize it to 255×255; ii) We train the network with both color and grayscale images. Our training set is a composition of grayscale and color images with a ratio of 30% to 70%; iii) We (spatially) jitter the color channels of the color images of each tile randomly by $\pm 0, \pm 1, \pm 2$ pixels.

Table 5 shows the performance of transfer learning our CFN, trained under different combinations of the above techniques to avoid shortcuts, to the detection task on Pascal VOC.

4.3 CFN filter activations

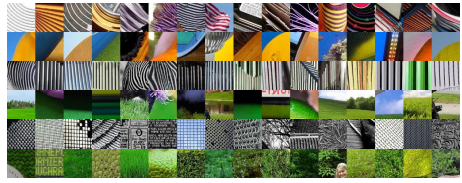
Some recent work has devoted efforts towards the visualization of CNNs to better understand how they work and how we can exploit them [42,35,28,22]. Some of these works aim at obtaining the input image that best represents a category according to a given neural network. This has shown that CNNs retain important information about the categories. Here instead we analyze the CFN by considering the units at each layer as object part detectors as in [15]. We extract 1M patches from the ImageNet validation set (20 randomly sampled 64×64 patches) and feed them as input to the CFN. At each layer (`conv1`, `conv2`, `conv3`, `conv4`, `conv5`) we consider the outputs of one channel and compute their ℓ_1 norm. We then rank the patches based on the ℓ_1 norm and select the top 16 ones that belong to different images. Since each layer has several channels, we hand-pick the 6 most significant ones. In Fig. 4 we show the top-16 activation patches for only 6 channels per layer. These activations show that the CFN features correspond to patterns sharing similar shapes and that there is a good correspondence based on object parts (in particular see the `conv4` activations for dog parts). Some channels seem to be good face detectors (see `conv3`, but the same detectors can be seen in other channels, not shown, in `conv4` and `conv5`) and others seem to be good texture detectors (*e.g.*, grass, water, fur). In Fig. 4(f) we also show the filters of the `conv1` layer of the CFN. We can see that these filters are quite strong and our transfer learning experiments in the next sections show that they are as effective as those trained in a supervised manner.

4.4 Image Retrieval

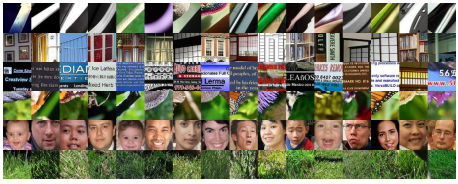
We also evaluate the features qualitatively (see Fig. 5) and quantitatively (see Fig. 6) for image retrieval with a simple image ranking.

We find the nearest neighbors (NN) of `pool5` features using the bounding boxes of the PASCAL VOC 2007 *test* set as query and bounding boxes of the *trainval* set as the retrieval entries. We discard bounding boxes with fewer than 10K pixels inside. In Fig. 5 we show some examples of image retrievals (top-4) obtained by ranking the images based on the inner product between normalized features of a query image and normalized features of the retrieval set. We can see that the features of the CFN are very sensitive to objects with similar shape and often these are within the same category. In Fig. 6 we compare CFN with the pre-trained AlexNet, [10], [39], and AlexNet with random weights. The precision-recall plots show that [10] and CFN features perform equally well. However, the real potential of CFN features is demonstrated when the feature metric is learned. In Table 2 we can see how CFN features surpass other features trained in an unsupervised way by a good margin. In that test the dataset (ImageNet) is more challenging because there are more categories and the bounding box is not used.

(a) conv1 activations

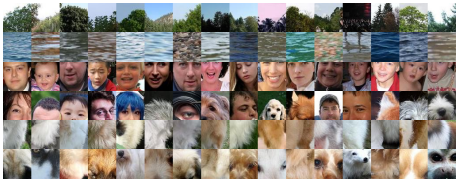


(b) conv2 activations



(c) conv3 activations

(d) conv4 activations

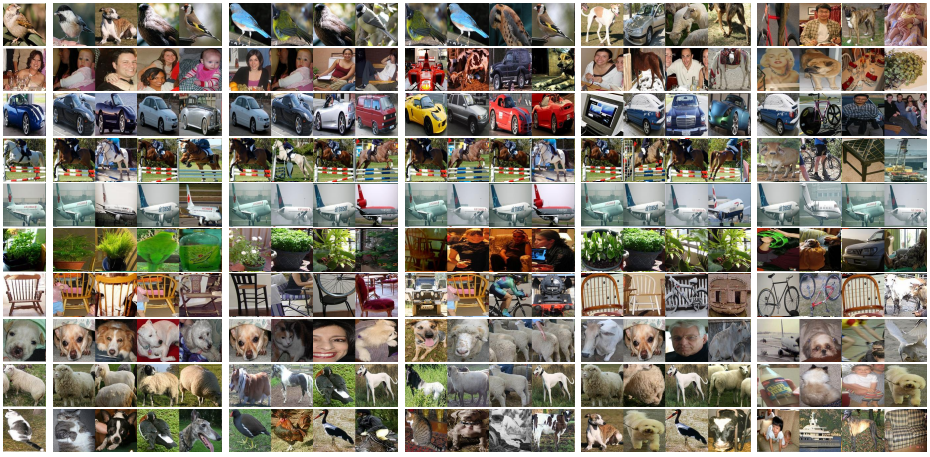


(e) conv5 activations

(f) conv1 filters without color jittering

(g) conv1 filters with color jittering

Fig. 4: Visualization of the top 16 activations for 6 units of the `conv1`, `conv2`, `conv3`, `conv4`, `conv5` layers in our CFN trained without blocking chromatic aberration. (f),(g) we show the filters of `conv1` trained without and with blocking chromatic aberration. The selection of the top activations is identical to the visualization method of Girshick *et al.* [15], except that we compute the average response rather than the maximum. We show some of the most significant units. We can see that in the first (a) and second (b) layers the filters specialize on different types of textures. On the third layer (c) the filters become more specialized and we have a first face detector (later layers will also have face detectors in some units) and some part detectors (*e.g.*, the bottom corner of the butterflies wing). On the fourth layer (d) we have already quite a number of part detectors. We purposefully choose all the dog part detectors: head top, head center, neck, back legs, and front legs. Notice the intraclass variation of the parts. Lastly, the fifth convolutional layer (e) has some other part detectors and some scene part detectors.



(a) (b) (c) (d) (e) (f)

Fig. 5: Image retrieval (qualitative evaluation). (a) query images; (b) top-4 matches with AlexNet; (c) top-4 matches with the CFN trained without blocking chromatic aberration; (d) top-4 matches with Doersch *et al.* [10]; (e) top-4 matches with Wang and Gupta [39]; (f) top-4 matches with AlexNet with random weights.

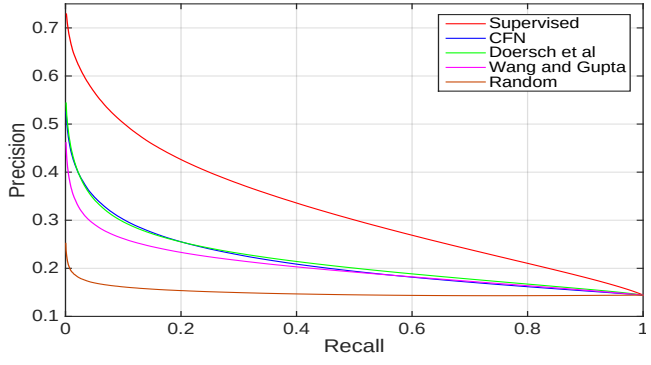


Fig. 6: Image retrieval (quantitative evaluation). We compare the precision-recall for image retrieval on the PASCAL VOC 2007. The ranking of the retrieved images is based on the inner products between normalized features extracted from a pre-trained AlexNet, the CFN, Doersch *et al.* [10], Wang and Gupta [39] and from AlexNet with random weights. The performance of CFN and [10] are very similar when using this simple ranking metric. When the metric is instead learned with two fully connected layers, then we see that CFN features yield a clearly higher performance than all other features from self-supervised learning methods (see Table 2).

5 Conclusions

We have introduced the *context-free* network (CFN), a CNN whose features can be easily transferred between detection/classification and Jigsaw puzzle reassembly tasks. The network is trained in an unsupervised manner by using the Jigsaw puzzle as a *pretext* task. We have built a training scheme that generates, on average, 69 puzzles for 1.3M images and converges in only 2.5 days. The key idea is that by solving Jigsaw puzzles the CFN learns to identify each tile as an object part and how parts are assembled in an object. The learned features are evaluated on both classification and detection and the experiments show that we outperform the previous state of the art. More importantly, the performance of these features is closing the gap with those learned in a supervised manner. We believe that there is a lot of untapped potential in self-supervised learning and in the future it will provide a valid alternative to costly human annotation.

References

1. Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. ECCV (2014)
2. Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. ICCV (2015)
3. Barlow, H.B.: Unsupervised learning. Neural Computation (1989)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation (2003)
5. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. PAMI (2013)
6. Boulard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biological Cybernetics (1988)
7. Chen, D.M., Baatz, G., Koser, K., Tsai, S.S., Vedantham, R., Pylvanainen, T., Roimela, K., Chen, X., Bach, J., Pollefeys, M., Girod, B., Grzeszczuk, R.: City-scale landmark identification on mobile devices. CVPR (2011)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. CVPR (2009)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., , Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. CVPR (2009)
10. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. ICCV (2015)
11. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. ICML (2014)
12. Everingham, M., Eslami, S.M.A., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. IJCV (2014)
13. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. CVPR (2003)
14. Freeman, H., Garder, L.: Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. IEEE Transactions on Electronic Computers (1964)
15. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR (2014)
16. Girshick, R.: Fast r-cnn. ICCV (2015)

17. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. PAMI (2015)
18. Hinton, G.E., Sejnowski, T.J.: Learning and relearning in boltzmann machines. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1 (1986)
19. Hinton, G.E., Zemel, R.S.: Autoencoders, minimum description length and helmholtz free energy. NIPS (1993)
20. Hooper, H.: The Hooper Visual Organization Test. Western Psychological Services, Los Angeles, CA (1983)
21. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML (2015)
22. Jason, Y., Jeff, C., Anh, N., Thomas, F., Hod, L.: Understanding neural networks through deep visualization. Deep Learning Workshop, ICML (2015)
23. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. ACM-MM (2014)
24. Krähenbühl, P., Doersch, C., Donahue, J., Darrell, T.: Data-dependent initializations of convolutional neural networks. ICLR (2016)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. NIPS (2012)
26. Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., Ng, A.: Building high-level features using large scale unsupervised learning. ICML (2012)
27. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
28. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. CVPR (2015)
29. Olshausen, B.A., Field, D.J.: "sparse coding with an overcomplete basis set: A strategy employed by v1? ". Vision Research (1997)
30. Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. CVPR (2016)
31. Pomeranz, D., Shemesh, M., Ben-Shahar, O.: A fully automated greedy square jigsaw puzzle solver. CVPR (2011)
32. Pomeranz, D.: Solving the square jigsaw problem. Ph.D. thesis, Ben-Gurion University of the Negev (2012)
33. Richardson, J., Vecchi, T.: A jigsaw-puzzle imagery task for assessing active visuospatial processes in old and young people. Behavior Research Methods, Instruments, & Computers (2002)
34. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science (2000)
35. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. ICLR (2014)
36. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. NIPS (2014)
37. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. Parallel Distributed Processing (1986)
38. Tybon, R.: Generating Solutions to the Jigsaw Puzzle Problem. Ph.D. thesis, Griffith University (2004)
39. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. ICCV (2015)

40. Weber, M., Weilling, M., Perona, P.: Unsupervised learning of models for recognition. ECCV (2000)
41. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? NIPS (2014)
42. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. ECCV (2014)