Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles

Mehdi Noroozi(✉) and Paolo Favaro

Institute of Informatics, University of Bern, Bern, Switzerland
{noroozi,paolo.favaro}@inf.unibe.ch

Abstract. We propose a novel unsupervised learning approach to build features suitable for object detection and classification. The features are pre-trained on a large dataset without human annotation and later transferred via fine-tuning on a different, smaller and labeled dataset. The pre-training consists of solving jigsaw puzzles of natural images. To facilitate the transfer of features to other tasks, we introduce the *context-free network* (CFN), a siamese-ennead convolutional neural network. The features correspond to the columns of the CFN and they process image tiles independently (i.e., free of context). The later layers of the CFN then use the features to identify their geometric arrangement. Our experimental evaluations show that the learned features capture semantically relevant content. We pre-train the CFN on the training set of the ILSVRC2012 dataset and transfer the features on the combined training and validation set of Pascal VOC 2007 for object detection (via fast RCNN) and classification. These features outperform all current unsupervised features with 51.8 % for detection and 68.6 % for classification, and reduce the gap with supervised learning (56.5 % and 78.2 % respectively).

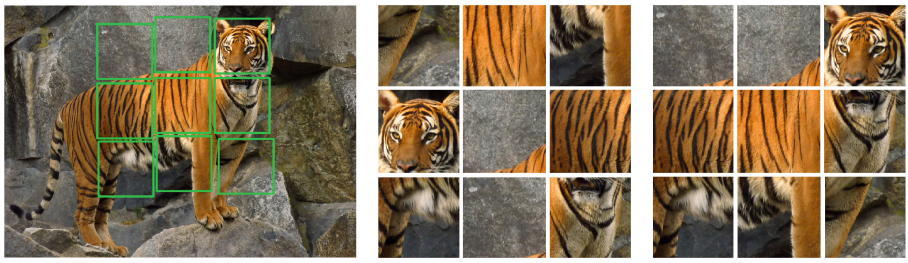
Keywords: Unsupervised learning · Image representation learning · Self-supervised learning · Feature transfer

1 Introduction

Visual tasks, such as object classification and detection, have been successfully approached through the supervised learning paradigm [1,10,23,33], 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.* [9], Wang and Gupta [36] 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. [9] uses the relative spatial co-location of patches in images as a label. [36] 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.

B. Leibe et al. (Eds.): ECCV 2016, Part VI, LNCS 9910, pp. 69–84, 2016.
DOI: 10.1007/978-3-319-46466-4_5



(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 or belong to the background) and their localization 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 [9]. (Color figure online)

A fundamental difference between [9] and [2,36] 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. 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 puzze 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 [9]. 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. In object classification these features lead to the best accuracy (38.1 %) when compared to other existing features trained via self-supervised learning on the ILSVRC2012 dataset [8]. Moreover, these features used as pre-training in the Fast R-CNN pipeline [14] achieve 51.8 % mAP for detection and 68.6 % for classification on PASCAL VOC 2007. This performance is close to that obtained in the supervised case by AlexNet [23] (56.5 % mAP for detection and 78.2 % in classification).

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* [37], as one 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. In this work we represent objects as a collection of parts (tiles) and design features to separate two factors: appearance and arrangement (geometry) of the parts.

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) [16,34], 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* [6,17,26]. 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.* [24]. 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 [4,31]. 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).

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 [2,9,36]. 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 [9,36]. 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.* [9] 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 [36] 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 [15]). The matched patches will have some intraclass variability due to changes in illumination, occlusion, viewpoint, pose, occlusions, and clutter factors. However, the choice of the third frame is crucial to learn non trivial features. If the third frame is too easily distinguishable (e.g., the color histogram is different) then the features might learn a simple shortcut.

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 Wang and Gupta [36]. However, because the task is to determine egomotion, the learned features may need to exploit correspondence across the two input frames and thus may focus on their similarities (such as color and texture), which might not generalize well to a category. 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 other (car/dog) image as they will be invariant to patterns shared across parts. 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.

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. (Color figure online)

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 [35]. Studies in Psychonomic show that jigsaw puzzles can be used to assess visuospatial processing in humans [30]. Indeed, the *Hooper Visual Organization Test* [18] is routinely used to measure 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.

There is also a sizeable literature on solving jigsaw puzzles computationally (see, for example, [12,28,29]). 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 information that can generalize well to a part detector.

Recently, a new data initialization technique for training convolutional neural networks by Krähenbühl *et al.* [22] has been applied to [2,9,36] with a remarkable increase in performance for object detection on PASCAL VOC 2007. The performance of the method of Doersch *et al.* [9] gains considerably with this data initialization method, and goes from 46.6 % mAP to 51.1 % mAP, which however is still below our 51.8 % mAP performance. Moreover, our training time is quite low: [9] takes 4 weeks while our method takes only 2.5 days (for completeness, [36] takes 1 week and [2] takes 10 h).

3 Solving Jigsaw Puzzles

In the following sections we describe how we cast the jigsaw puzzle task so that features suitable for object detection and classification are implicitly learned.

3.1 The Jigsaw Puzzle Task

In the jigsaw task, the input data is a collection of identical square tiles from an image and the labels are permutations of these tiles taken from a predefined

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

Output: P

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: $\quad P \leftarrow [P\ \bar{P}_j]$ \\ *add permutation* $\bar{P}_j$ *to* P
7: $\quad \bar{P} \leftarrow [\bar{P}_1, \dots, \bar{P}_{j-1}, \bar{P}_{j+1}, \dots]$ \\ *remove* $\bar{P}_j$ *from* $\bar{P}$
8: $\quad D \leftarrow \text{Hamming}(P, \bar{P})$ \\ D *is an* $i \times (9! - i)$ *matrix*
9: $\quad \bar{D} \leftarrow \mathbf{1}^T D$ \\ $\bar{D}$ *is a* $1 \times (9! - i)$ *row vector*
10: $\quad j \leftarrow \arg\max_k \bar{D}_k$ \\ $\bar{D}_k$ *denotes the k-th entry of* $\bar{D}$
11: $\quad i \leftarrow i + 1$
12: **until** $i \leq 100$

set $\mathcal{S}$. For example, an image is divided in a 3×3 grid, where each tile corresponds to a number from 1 to 9 (see Fig. 3). A permutation is an ordering of the tiles, *e.g.*, $(9, 4, 6, 8, 3, 2, 5, 1, 7)$. We generate the set $\mathcal{S}$ via the greedy algorithm shown in Algorithm 1 and never change it during or after training. A factor that was found important to learn transferrable features is the Hamming distance between the permutations (*i.e.*, the number of different tile locations between 2 permutations $S_1, S_2 \in \mathcal{S}$ divided by 9). From our experimental validation, the average Hamming distance seems to control the difficulty of the jigsaw puzzle reassembly task, and to also correlate with the object detection performance. In Table 4 we compare 3 choices for the Hamming distance: minimal, middle and maximal. For the minimal and middle case, the $\arg\max_k$ function at line 10 is replaced by $\arg\min_k$ and uniform sampling respectively. From those tests we can see that large Hamming distances are desirable.

Although there are $9! = 362{,}880$ possible permutations with 9 tiles, it turns out that it is not necessary to consider the whole space of permutations. We have found out experimentally that increasing the cardinality of $\mathcal{S}$ beyond 100 does not improve any performance. Indeed, by setting $|\mathcal{S}| = 1000$ the `CFN-9(middle)` (3×3 grid with middle case for the Hamming distance) can solve puzzles with an accuracy of 85 %, achieves the classification performance of 38 % on ILSRVC 2012 [8] with all layers locked (see Table 2), and takes 2.5 days to train (notice that only the last layer changes with the size of $\mathcal{S}$ and the training dataset size does not change). Thus, in all the experiments below we fix $|\mathcal{S}| = 100$.

3.2 The Context-Free Architecture

Jigsaw puzzles can be solved by matching low-level statistics (*e.g.*, structural patterns or texture) close to the boundaries of adjacent tiles, that is, by ignoring the inner contents of the tiles. While these are cues that humans often use to solve jigsaw puzzles, they do not lead to a complete representation of the global object, which is necessary for object classification. 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 on the

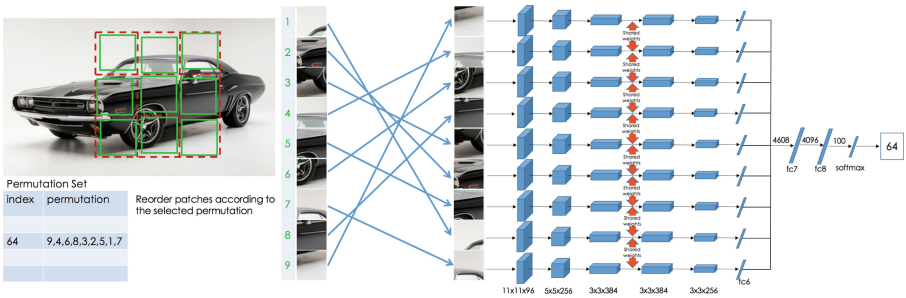


Fig. 3. Context Free Network. The figure illustrates how a puzzle is generated and solved. We randomly crop a 225×225 pixel window from image (red dashed box), divide it into a 3×3 grid, and randomly pick a 64×64 pixel tiles from each 75×75 pixel cell. These 9 tiles are reordered via a randomly chosen 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). 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 [23]. The only difference is that we set the stride of the first layer to 2 instead of 4. (Color figure online)

right-hand side of Fig. 3). Then, it finds the parts arrangement just by using these features (*fc7* and *fc8* 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.

Towards this goal we build a siamese-ennead convolutional network (see Fig. 3) where each row up to the first fully connected layer (*fc6*) uses the AlexNet architecture [23] with shared weights. Similar schemes were used in prior work [2,9,36]. 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 ILSVRC2012 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 (see Table 2). Thus, limiting the receptive field of *fc6* to a tile does not significantly affect the performance in classification.

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 $S \in \mathcal{S}$ is the configuration of the tiles, $\{A_i\}_{i=1,\ldots,9}$ form the appearance of an object, and $\{F_i\}_{i=1,\ldots,9}$ form the intermediate feature representation. In our CFN (see Fig. 3), A_i corresponds to the RGB tile fed to the i-th row, F_i corresponds to the output of *fc6* in the i-th row, and $p(S|F_1, F_2, \ldots, F_9)$ is the output of the softmax layer. 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 $S = (L_1, \ldots, L_9)$ then in this case the conditional pdf $p(S|F_1, F_2, \ldots, F_9)$ would factorize into independent terms

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

where each tile location L_i is fully determined by the corresponding feature F_i. The CFN would not have learned the correlation between different tiles.

To avoid these issues we feed multiple jigsaw puzzles of the same image to the CFN (an average of 69 out of 100 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.

As mentioned earlier on, we also leave a random gap between the tiles to discourage the CFN from learning low-level statistics. This was also done in [9]. 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.

No color dropping or filling image channels with noise was needed. We used Caffe [21] and modified the code to choose random image patches and permutations during the training time. This allowed us to keep the dataset small (1.3 M images from ImageNet) and the training efficient, while the CFN could see an average of 69 different puzzles per image (that is about 90 M different jigsaw puzzles).

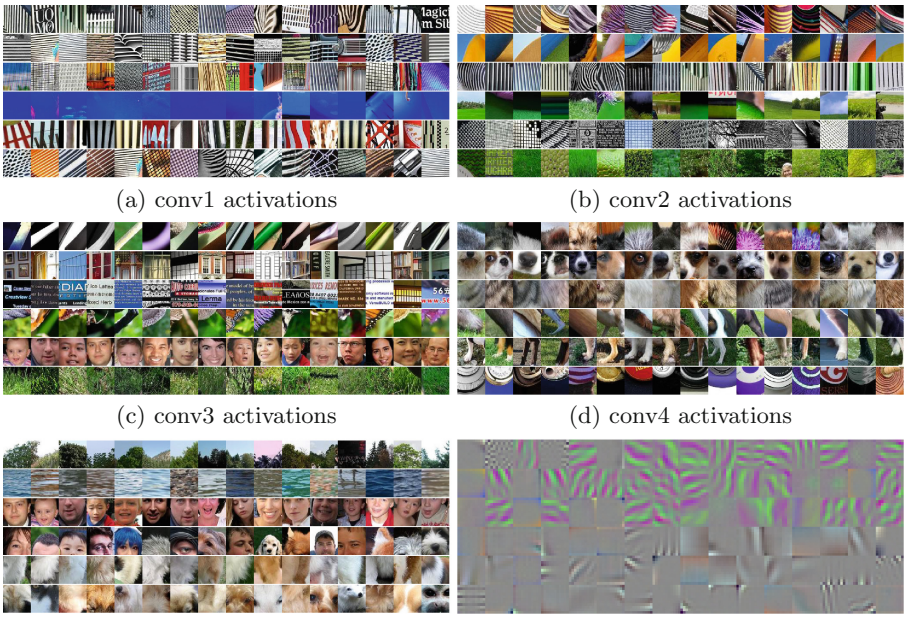
3.4 Implementation Details

We use stochastic gradient descent without batch normalization [19] 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. We preserve the aspect ratio of the original images by resizing them until the smallest between their height and 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 h in total ($\sim$2.5 days). If we take $122\,\% = \frac{3072\text{cores@1000 Mhz}}{2880\text{cores@875 Mhz}} = \frac{6{,}144\text{GFLOPS}}{5{,}040\text{GFLOPS}}$ as the best possible performance ratio between the Titan X and the Tesla K40 (used for [9]) we can predict that the CFN would have taken $\sim$72.5 h ($\sim$3 days) on a Tesla K40. We compute that on average each image is used $350\,\text{K} \times 256/1.3\,\text{M} \simeq 69$ times.

3.5 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 [20,25,32,38]. Some of these works and also Google Inceptionism[1] 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 [13]. 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. Because of the green-magenta bias, our `conv1` filters seem to be also sensitive to chromatic aberration [9]. However, it appears as though this bias does not affect the performance when features are transferred to detection and classification tasks (see next section).

[1] See http://googleresearch.blogspot.co.uk/2015/06/inceptionism-going-deeper-into-neural.html.



(a) conv1 activations (b) conv2 activations

(c) conv3 activations (d) conv4 activations

(e) conv5 activations (f) conv1 filters

Fig. 4. Visualization of the top 16 activations for 6 units of the `conv1`, `conv2`, `conv3`, `conv4`, `conv5` layers in our CFN. (f) we show the filters of `conv1`, which show a green-magenta bias. The selection of the top activations is identical to the visualization method of Girshick *et al.* [13], 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. (Color figure online)

4 Experiments

We evaluate our learned features by using transfer learning [37] on the object classification task on ImageNet and as pre-trained weights for classification and detection tasks on PASCAL VOC 2007 [11]. 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 pre-trained AlexNet and transfer its features to solve 3×3 jigsaw puzzles with permutations generated with the middle case Hamming distance (which is used by the `CFN-9(middle)` in later sections). We also use the same locking scheme as in [37] to see the transferability of features at

Table 1. 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 [23]	88	87	86	83	74

Table 2. Comparison of classification results on validation set of ImageNet 2012.

	🔒 conv1	🔒 conv2	🔒 conv3	🔒 conv4	🔒 conv5
CFN	**57.1**	**56.0**	**52.4**	**48.3**	**38.1**
Doersch *et al.* [9]	53.1	47.6	48.7	45.6	30.4
Wang and Gupta [36]	51.8	46.9	42.8	38.8	29.8
Random	48.5	41.0	34.8	27.1	12.0

different layers. The performance is shown in Table 1. Compared to the maximum accuracy of the `CFN-9(middle)` (88 %) we can see that semantic training is quite helpful towards recognizing object parts. Indeed, the performance is very high up to `conv4`. Finally, we also compare the CFN features with those of [9,36] both qualitatively and quantitatively on image retrieval.

4.1 ImageNet Classification

Yosinski *et al.* [37] 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 the CFN. We repurpose the CFN, [9,36] to the classification task on the ImageNet 2012 dataset [8] and Table 2 summarizes the results on the validation set. 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 38.1 % when only fully connected layers are trained, which is 7.0 % higher than the next best performing algorithm [9]. There is a significant improvement (from 38.1 % to 48.3 %) when the `conv5` layer is also trained. This shows that the `conv5` layer starts to be specialized in the jigsaw puzzle reassembly task.

4.2 Pascal VOC 2007 Classification and Detection

We use the CFN features for object detection, with Fast R-CNN [14], and classification tasks on PASCAL VOC 2007. For both tasks we use the same baseline

Table 3. Results on PASCAL VOC 2007 Detection and Classification. The results of the other methods are taken from Pathak *et al.* [27].

Method	Pretraining time	Supervision	Classification	Detection
Krizhevsky*et al.* [23]	3 days	1000 class labels	**78.2 %**	**56.8 %**
Wang and Gupta [36]	1 week	motion	58.4 %	44.0 %
Doersch *et al.* [9]	4 weeks	context	55.3 %	46.6 %
Pathak *et al.* [27]	14 h	context	56.5 %	44.5 %
CFN-9(max)	2.5 days	context	**68.6 %**	**51.8 %**

Table 4. Object detection results on Pascal VOC 2007 for different configurations of the CFN. We compare the 2×2 grid (trained on all 24 permutations) and 3×3 grid with different average Hamming distances. We can see that the 3×3 grid achieves better results. Moreover, the average Hamming distance between permutations has a direct impact on the performance of the learned features.

CFN-4	CFN-9(min)	CFN-9(middle)	CFN-9(max)	CFN-sup
49.8 %	51.0 %	51.2 %	**51.8 %**	**56.3 %**

used by Krähenbühl *et al.* [22]. Because our fully connected layers are different from those used in Fast R-CNN, 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 3.

We evaluate the impact of different configurations on the detection task in Table 4. We consider two cases for the CFN pre-trained with the jigsaw puzzle task: one based on a 2×2 grid (denoted `CFN-4`) and one based on a 3×3 grid (denoted `CFN-9`). The average performance shows that `CFN-9` learns better features. We also analyze the impact of the average Hamming distance between permutations on the detection task. We generate 3 cases for the average Hamming distance: .45 (`CFN-9(min)`), .67 (`CFN-9(middle)`), and .88 (`CFN-9(max)`). As shown in Table 4 the feature map extracted from `CFN-9(max)` is the best performing one. We also use as a reference the CFN pre-trained on ImageNet with labels and call it `CFN-Sup` in Table 4. This test is done only to show the full potential of the network when trained in a supervised manner for classification. Indeed, we get 56.3 % mAP for detection which is almost the same as that of AlexNet (56.5 % mAP). `CFN-9(max)` features achieve 51.8 % mAP in detection, and 68.6 % in classification, thus outperforming all other methods and reducing the gap with features obtained with supervision.

4.3 Image Retrieval

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

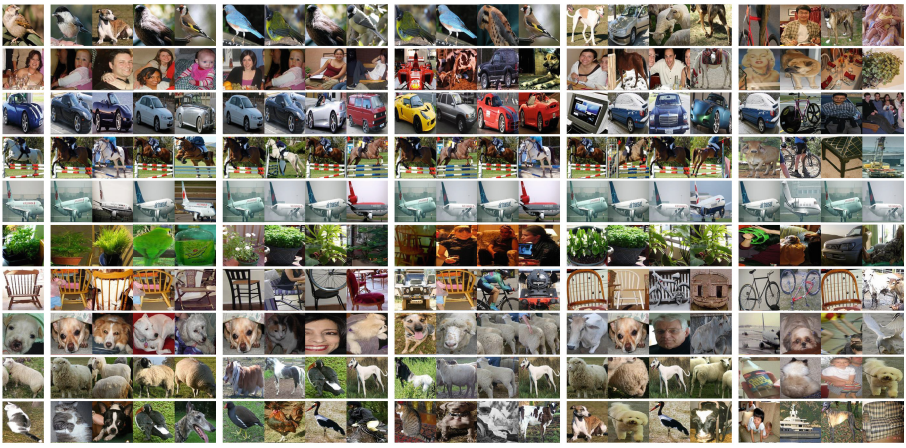


Fig. 5. Image retrieval (qualitative evaluation). (a) query images; (b) top-4 matches with AlexNet; (c) top-4 matches with the CFN; (d) top-4 matches with Doersch *et al.* [9]; (e) top-4 matches with Wang and Gupta [36]; (f) top-4 matches with AlexNet with random weights.

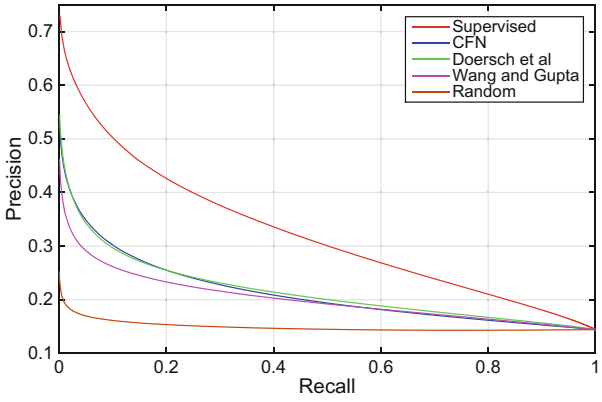


Fig. 6. Image retrieval (quantitative evaluation). We compare the precision-recall for image retrieval on the PASCAL VOC 2007. The ranking of the retrieved images is based on the inner products between normalized features extracted from a pre-trained AlexNet, the CFN, Doersch *et al.* [9], Wang and Gupta [36] and from AlexNet with random weights. The performance of CFN and [9] 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).

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, [9,36] and AlexNet with random weights. The precision-recall plots show that [9] 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.

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 1.3 M 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 help to costly human annotation. One possible extension to this work is the use of a hierarchical-framework where each tile can also provide a puzzle and thus represent not only parts within objects, but also subparts within parts and placement of objects within a scene. The puzzle solver could learn to solve all these cases at once.

Acknowledgements. We thank Philipp Krähenbühl for his assistance with the experiments on Pascal VOC 2007 and for kindly evaluating our CFN weights on his configuration for classification with Pascal VOC 2007.

References

1. Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 329–344. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10584-0_22

2. Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. In: ICCV (2015)
3. Barlow, H.B.: Unsupervised learning. Neural Comput. **1**, 295–311 (1989)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**, 1373–1396 (2003)
5. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. PAMI **35**(8), 1798–1828 (2013)
6. Boulard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biol. Cybern. **59**, 291–294 (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. In: CVPR (2011)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
9. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV (2015)
10. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: a deep convolutional activation feature for generic visual recognition. In: ICML (2014)
11. 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)
12. Freeman, H., Garder, L.: Apictorial jigsaw puzzles: the computer solution of a problem in pattern recognition. IEEE Trans. Electron. Comput. **EC–13**, 118–127 (1964)
13. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
14. Girshick, R.: Fast r-cnn. In: ICCV (2015)
15. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. PAMI (2015)
16. Hinton, G.E., Sejnowski, T.J.: Learning and relearning in boltzmann machines. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1 (1986)
17. Hinton, G.E., Zemel, R.S.: Autoencoders, minimum description length and helmholtz free energy. NIPS (1993)
18. Hooper, H.: The Hooper Visual Organization Test. Western Psychological Services, Los Angeles (1983)
19. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
20. Jason, Y., Jeff, C., Anh, N., Thomas, F., Hod, L.: Understanding neural networks through deep visualization. In: Deep Learning Workshop, ICML (2015)
21. 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)
22. Krähenbühl, P., Doersch, C., Donahue, J., Darrell, T.: Data-dependent initializations of convolutional neural networks. In: ICLR (2016)
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. NIPS (2012)
24. 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. In: ICML (2012)

25. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: CVPR (2015)
26. Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: a strategy employed by v1? Vision Research (1997)
27. Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: feature learning by inpainting. In: CVPR (2016)
28. Pomeranz, D., Shemesh, M., Ben-Shahar, O.: A fully automated greedy square jigsaw puzzle solver. In: CVPR (2011)
29. Pomeranz, D.: Solving the square jigsaw problem. Ph.D. thesis, Ben-Gurion University of the Negev (2012)
30. 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)
31. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science (2000)
32. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: ICLR (2014)
33. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. NIPS (2014)
34. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. Parallel Distributed Processing (1986)
35. Tybon, R.: Generating Solutions to the Jigsaw Puzzle Problem. Ph.D. thesis, Griffith University (2004)
36. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: ICCV (2015)
37. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? NIPS (2014)
38. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10590-1_53