Iterative Reorganization with Weak Spatial Constraints:
Solving Arbitrary Jigsaw Puzzles for Unsupervised Representation Learning

Chen Wei[1,*], Lingxi Xie[2,3,†], Xutong Ren[1], Yingda Xia[2], Chi Su[4], Jiaying Liu[1], Qi Tian[3], Alan L. Yuille[2]
[1]Peking University   [2]Johns Hopkins University   [3]Noah's Ark Lab, Huawei Inc.   [4]Kingsoft Cloud

weichen582@pku.edu.cn   198808xc@gmail.com   tonghelen@pku.edu.cn   yxia25@jhu.edu

suchi@kingsoft.com   liujiaying@pku.edu.cn   tian.qi1@huawei.com   alan.l.yuille@gmail.com

Abstract

Learning visual features from unlabeled image data is an important yet challenging task, which is often achieved by training a model on some annotation-free information. We consider spatial contexts, for which we solve so-called jigsaw puzzles, i.e., each image is cut into grids and then disordered, and the goal is to recover the correct configuration. Existing approaches formulated it as a classification task by defining a fixed mapping from a small subset of configurations to a class set, but these approaches ignore the underlying relationship between different configurations and also limit their applications to more complex scenarios.

This paper presents a novel approach which applies to jigsaw puzzles with an arbitrary grid size and dimensionality. We provide a fundamental and generalized principle, that weaker cues are easier to be learned in an unsupervised manner and also transfer better. In the context of puzzle recognition, we use an iterative manner which, instead of solving the puzzle all at once, adjusts the order of the patches in each step until convergence. In each step, we combine both unary and binary features of each patch into a cost function judging the correctness of the current configuration. Our approach, by taking similarity between puzzles into consideration, enjoys a more efficient way of learning visual knowledge. We verify the effectiveness of our approach from two aspects. First, it solves arbitrarily complex puzzles, including high-dimensional puzzles, that prior methods are difficult to handle. Second, it serves as a reliable way of network initialization, which leads to better transfer performance in visual recognition tasks including classification, detection and segmentation.

1. Introduction

Deep learning especially convolutional neural networks has been boosting the performance of a wide range of
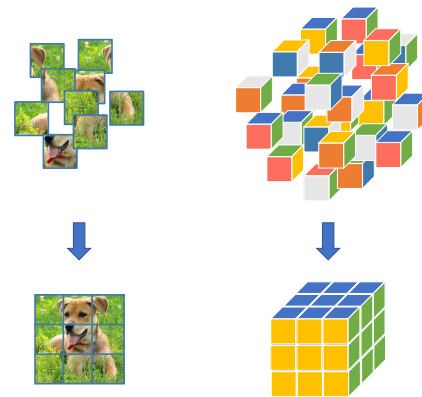

Figure 1. We study the problem of solving jigsaw puzzles for visual recognition. Compared to the previous work [31] working on 1,000 fixed configurations of 3×3 puzzles, our method can generalize to arbitrary configurations of 2D and 3D puzzles.

applications in computer vision [24]. These statistics-based approaches build hierarchical structures which contain a large number of neurons, so that visual knowledge is learned by fitting labeled training data [21]. However, annotating a large-scale dataset is often difficult and expensive. Therefore, weakly supervised or unsupervised learning has attracted a lot of research attentions [48, 23]. These approaches are often built on some naturally existing constraints such as temporal consistency [47], spatial relationship [8] and sum-up equations [32]. Such information, though being weak, constructs loss functions without requiring annotations, and networks pre-trained in this way can either be used for weak visual feature extraction [47] or fine-tuned in a standalone supervised learning process towards better recognition performance [10].

In this work, we focus on a specific way of exploiting spatial relationship, which is to solve so-called *jigsaw puzzles* in unlabeled image data [31, 33, 44, 3]. These approaches work by cutting an image into a grid, say, 3×3, of patches and then disordering them as training data, with

---

the goal to recover its correct spatial configuration. Examples are shown in Figure 1. Thus, in order to achieve this goal, the network should have the ability to capture some semantic information, *e.g.*, learning the concept of *car* and *ground*, though not labeled, knowing that *car* always appears above *ground*. Technically, these approaches simply assigned each configuration a unique ID, so that puzzle recognition turns into a *plain* classification problem.

We point out two major drawbacks of this strategy. First, a plain classifier takes an assumption that all configurations are equally similar to each other, which is not true even when inter-class distances are maximized using a greedy algorithm [31]. This brings a negative impact to representation learning. Second, the number of parameters required for plain classification increases linearly with the number of configurations, so that it is difficult to deal with all possible configurations due to the risk of over-fitting. For example, there are $9! = 362,880$ possible configurations for a 3×3 puzzle, but the original approach [31] reached the best performance at $1,000$ and observed over-fitting when this number continues growing. Both of these drawbacks limit us from generalizing this approach to more complex puzzles. On the other hand, solving complex puzzles is especially useful for some areas such as medical imaging analysis, in which it is difficult to pre-train 3D networks [5, 28] in 2D scenarios, yet a reasonable initialization helps a lot on training stability and testing performance. An empirical study of this topic can be found in Section 4.3.

In this paper, we extend the ability of such approaches by allowing to solve arbitrary jigsaw puzzles, *i.e.*, the puzzles are not constrained by a pre-defined set of configurations. Our improvement lies in two parts. First, we allow the jigsaw puzzles to have an arbitrary configuration rather than being limited in a fixed set. Second, we introduce an iterative solver built upon weak spatial constraints to replace the direct classifier used previously. As a result, our approach extends the ability of representation learning based on jigsaw puzzles and transfers well to 3D data. To this end, we formulate puzzle recognition into an optimization problem which involves a set of unary and binary terms, with each unary term indicating whether a specified patch is located at a specified position, and each binary term measuring whether two patches should have a specified relative position. These terms are determined by a deep network backbone so that the entire system can be trained in an end-to-end manner. In both training and testing, we allow the first trial not to find the correct configuration, in which case we iteratively adjust the configuration according to prediction until convergence.

We evaluate our approach in both puzzle recognition and transfer learning. The puzzle solver is trained on ILSVRC2012 training set [43] and tested on the validation set, both of which do not contain class labels. Our approach solves arbitrary jigsaw puzzles with reasonable accuracy, while the prior approaches can only work on a limited set of puzzles. Then, we transfer the pre-trained model to be fine-tuned on the Pascal VOC 2007 dataset [11] for image classification and object detection. Either learning from more complex puzzles or achieving higher accuracy in puzzle recognition boosts transfer learning performance, which verifies our motivation. Finally, we apply our approach to initialize a 3D network with unlabeled medical data, and verify its effectiveness in segmenting an abdominal organ from CT scans.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work, and Section 3 describes the proposed approach. After experiments shown in Section 4, we draw our conclusions in Section 5.

2. Related Work

Deep neural networks have been playing an important role in modern computer vision systems. With the availability of large-scale datasets [7] and powerful computational device such as GPUs, researchers have designed network structures with tens [21, 45, 46] or hundreds [16, 17] of layers towards better recognition performance. Also, the networks pre-trained on ImageNet were transferred to other recognition tasks by either extracting visual features directly [9, 15, 37] or being fine-tuned on a new loss function [27, 38]. Despite their effectiveness, these networks still strongly rely on labeled image data. However in some areas such as medical imaging, data collection and annotation can be expensive, time-consuming, or requiring expertise. Thus, there has been efforts to design unsupervised [48, 23] or weakly supervised [18] approaches which learn visual knowledge from unlabeled data, or semi-supervised learning algorithms [34, 35] which aim at combining a limited amount of labeled data and a large corpus of unlabeled data towards better performance. It has been verified that unsupervised pre-training helps supervised learning especially deep learning [10].

The key factor to learning from unlabeled data is to establish some kind of *prior*, or some weak constraints that naturally exist, *i.e.*, no annotations are required. Such prior can be either (1) embedded into the network architecture or (2) encoded as a weak supervision to optimize the network. For the first type, researchers designed clustering-based approaches to optimize visual representation [50, 4], as well as generator-based approaches [36, 54].

This paper mainly considers the second type which, in comparison to the first type, is much easier in algorithmic design. Typical examples include temporal consistency which assumes that neighboring video frames contain similar visual contents [47, 12], the temporal order in the context of video [29, 2, 25], spatial relationship between some pairs of unlabeled patches [8], learning an additive function

on different regions as well as the entire image [32], *etc*. Among these priors, spatial contexts are widely believed to contain rich information which a vision system should be able to capture. Going one step beyond modeling patch relationship [8], researchers designed so-called jigsaw puzzles [31, 33, 44, 3] which are more complex so that the networks are better trained by learning to solve them.

Researchers believed that learning from these weakly-supervised cues can help visual recognition, because many problems are indeed built on understanding and integrating this type of information. Regarding spatial contexts, a wide range of recognition tasks can benefit from understanding the relative position of two or more patches, such as image classification [1], semantic segmentation [42] and parsing [52], *etc*.

3. Our Approach

3.1. Problem and Baseline Solution

The problem of puzzle recognition assumes that an image is partitioned into a grid (*e.g.*, 3×3) of patches and then disordered. The task is to recover the original configuration (*i.e.*, patches are ordered in the natural form). To accomplish this task, the network needs to understand what a patch contains as well as how two or more patches are related to each other (*e.g.*, in a *car* image, a *wheel* is often located to the top of the *ground*). Therefore, we expect this task to teach a network both intra-patch and inter-patch information, which we formulate as unary terms and binary terms, respectively.

We first define the terminologies used in this paper. Let $\mathbf{I}$ be an image, which is partitioned into $W \times H$ patches. Each patch, denoted as $\mathbf{i}_{x,y}$ ($0 \leqslant x < W$, $0 \leqslant y < H$), is assigned a unique ID $a_{x,y} \in \{0, 1, \ldots, WH - 1\}$ according to its original position, *e.g.*, the row-major policy gives $a_{x,y} = x + yW$. After that, all patches are randomly disordered, and we use $c^{\star}_{x,y}$ to denote the ID owned by the patch that currently occupies the (x, y) position. All $c^{\star}_{x,y}$ values compose a configuration, denoted as $\mathbf{c}^{\star} = \left(c^{\star}_{x,y}\right)^{W,H}_{x=0,y=0}$. There are in total $(WH)!$ different configurations, composing the configuration set $\mathcal{C}$ that $|\mathcal{C}| = (WH)!$.

Our goal is to predict the correct configuration $\mathbf{c}^{\star} \in \mathcal{C}$. For this purpose, a network structure with two parts was constructed [31]. The network *backbone* $\mathbb{M}^{\mathrm{B}}$: $\mathbf{f}_{x,y} = \mathbf{f}\left(\mathbf{i}_{x,y}; \boldsymbol{\theta}^{\mathrm{B}}\right)$ is built upon each individual patch, and outputs a set of features for the network *head* $\mathbb{M}^{\mathrm{H}}$: $\mathbf{c} = \mathbf{g}\left(\mathbf{F}; \boldsymbol{\theta}^{\mathrm{H}}\right)$ to produce the final output $\mathbf{c} = (c_{x,y})^{W,H}_{x=0,y=0}$, where $\mathbf{F} = (\mathbf{f}_{x,y})^{W,H}_{x=0,y=0}$ is the ordered concatenation of patch features. In practice, $\mathbf{f}\left(\cdot; \boldsymbol{\theta}^{\mathrm{B}}\right)$ is often borrowed from existing network architectures [21, 45, 16], while $\mathbf{g}\left(\cdot; \boldsymbol{\theta}^{\mathrm{H}}\right)$

is often more interesting to investigate.

In the prior work [31, 33], the network head worked by constraining the number of possible configurations, say $K = 1,000$ out of $9!$, which are randomly sampled from $\mathcal{C}$ using a greedy algorithm to guarantee the Hamming distance between any two configurations is sufficiently large. Then, $\mathbf{f}\left(\cdot; \boldsymbol{\theta}^{\mathrm{H}}\right)$ was designed to be a K-way classifier, implemented as a fully-connected layer. The purpose of this design was mainly to control the number of parameters of the classifier (proportional to K) so as to prevent overfitting[1], but we argue that it largely limits the model from being applied more complex scenarios like 3D puzzles, while it was believed that learning from a harder task can lead to a stronger ability [6]. This motivates us to propose a new approach in which the number of configurations can be arbitrarily large while the number of parameters remains unchanged. We will see later that the essence behind this motivation is to use weak cues with an iterative algorithm towards a more compact representation and a safer learning process.

3.2. Solving Jigsaw Puzzles with Weak Cues

We design a network head to learn *weak spatial constraints*. By "weak" we are comparing this strategy with the aforementioned K-way classifier that predicts the configuration of the entire puzzle all at once. Instead, we consider an indirect cost function $S(\mathbf{I}, \mathbf{c})$ which outputs a cost that patch $\mathbf{i}_{x,y}$ or equivalently feature $\mathbf{f}_{x,y}$ is located at position $c_{x,y}$, and thus the most probable configuration is determined by $\arg\max_{\mathbf{c}} \{S(\mathbf{I}, \mathbf{c})\}$. $S(\mathbf{I}, \mathbf{c})$ is composed of two parts, namely, unary terms and binary terms. Each *unary term* provides cues for the *absolute* position of a patch, and each *binary term* provides cues for the *relative* position of two patches. Mathematically,

$$
\begin{aligned}
S(\mathbf{I}, \mathbf{c}) \equiv S(\mathbf{F}, \mathbf{c}) = \sum_{(x,y)} p_1(\mathbf{f}_{x,y}, c_{x,y} \mid \mathbf{F}) + \\
\sum_{(x_1,y_1) \neq (x_2,y_2)} p_2(\mathbf{f}_{x_1,y_1}, \mathbf{f}_{x_2,y_2}, c_{x_1,y_1}, c_{x_2,y_2}).
\end{aligned} \quad (1)
$$

Here, $p_1(\mathbf{f}_{x,y}, c_{x,y} \mid \mathbf{F})$ is a unary term which measures how likely that patch $f_{x,y}$ is located at position $c_{x,y}$, and $p_2(\mathbf{f}_{x_1,y_1}, \mathbf{f}_{x_2,y_2}, c_{x_1,y_1}, c_{x_2,y_2})$ is a binary term measures how likely that patches $\mathbf{f}_{x_1,y_1}$ and $\mathbf{f}_{x_2,y_2}$ have the spatial relationship indicated by c_{x_1,y_1} and c_{x_2,y_2}. Each unary term is computed based on $\mathbf{F}$, the overall variable containing feature vectors of all patches, because the position of each

---

[1][31] observed that setting a larger K leads to performance drop in transfer experiments, and explained it as the network gets confused by very similar jigsaw puzzles. However, as shown in experiments (see Section 4.2), our approach works well in the entire puzzle set $\mathcal{C}$, *i.e.*, $K = 9! = 362,880$, which implies that the performance drop may due the large number of parameters.
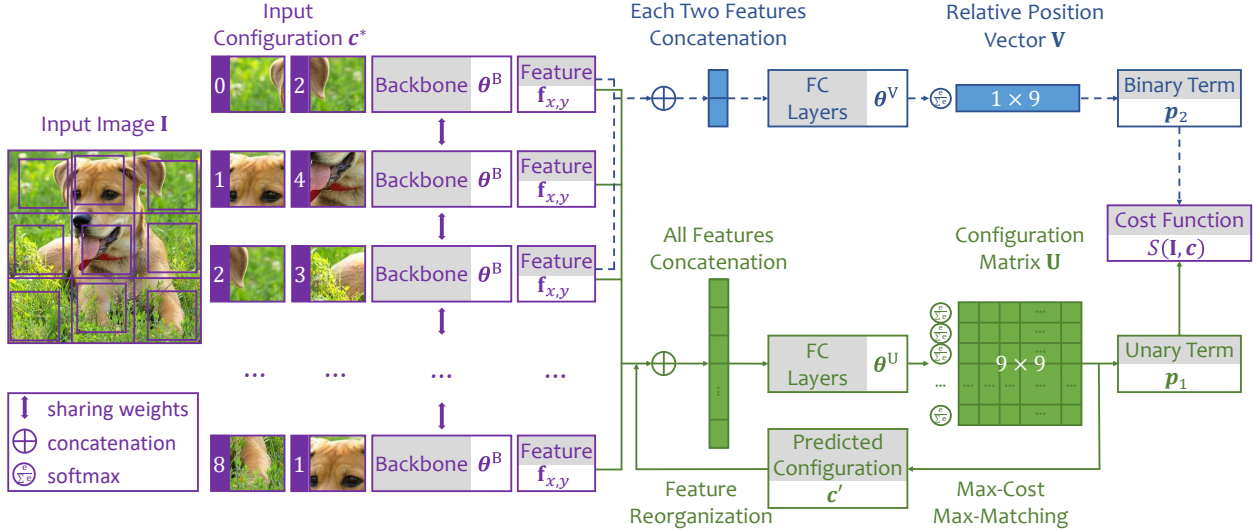
Figure 2. The overall structure (best viewed in color). Each training image (without semantic annotations) is randomly cropped, disordered and fed into puzzle recognition network. Two types of loss terms (unary and binary) are computed and summed into the final cost function $S(\mathbf{I}, \mathbf{c})$. The training process continues until the puzzle is completely correct or a maximal number of rounds is achieved.

patch $\mathbf{f}_{x,y}$ depends on the visual messages delivered by other patches. The binary terms, on the other hand, do not have such a dependency.

In practice, the unary terms are formulated in a matrix $\mathbf{U}$ with $WH \times WH$ elements, each of which, $[\![\mathbf{U}]\!]_{a,c}$, indicates the cost obtained by putting the specified patch with ID a at a specified position with ID c. This is implemented by a fully-connected layer between $\mathbf{F}$ and these $(WH)^2$ elements, parameterized by $\boldsymbol{\theta}^{\mathrm{U}}$. We perform the softmax function over all elements in each row, so that the scores corresponding to each patch sum to 1^2. Then, each unary term is the log-likelihood of the score at a specified position:

$$p_1(\mathbf{f}_{x,y}, c_{x,y}, \mathbf{F}) = -\ln \left[\!\!\left[\mathbf{U}\left(\mathbf{F}; \boldsymbol{\theta}^{\mathrm{U}}\right) \right]\!\!\right]_{a_{x,y}, c_{x,y}}. \quad (2)$$

For each binary term involving $\mathbf{f}_{x_1, y_1}$ and $\mathbf{f}_{x_2, y_2}$, we build another mapping from these two vectors to a 9-dimensional vector, with each index indicating the probability that the spatial relationship of $\mathbf{f}_{x_1, y_1}$ and $\mathbf{f}_{x_2, y_2}$ belongs to one of the 9 possibilities, namely, the first patch is located to the top, bottom, left, right, top-left, top-right, bottom-left, bottom-right of the second patch or none of the above happens. Similarly, this is implemented using another fully-connected layer between $\mathbf{f}_{x_2, y_2} \oplus \mathbf{f}_{x_2, y_2}$ ($\oplus$ denotes concatenation) and a 9-dimensional vector parameterized by $\boldsymbol{\theta}^{\mathrm{V}}$

---

[2] Ideally, the elements in each column should also sum to 1, but it is mathematically intractable if we hope to keep the ratio between all elements. There are two arguments. First, after normalizing scores in each row, we find that there often exists one major elements in each column, and the sum of each column is close to 1. Second, we add an additional ℓ_1 loss term between the sum of each column and 1, but only observe to minor changes in either puzzle recognition accuracy or transfer learning performance.

followed by a softmax activation over these 9 numbers. We denote $r_{x_1, y_1, x_2, y_2} \doteq r(c_{x_1, y_1}, c_{x_2, y_2}) \in \{0, 1, \ldots, 8\}$ as the relative position type between $\mathbf{f}_{x_1, y_1}$ and $\mathbf{f}_{x_2, y_2}$, so that we can write the binary term as:

$$p_2(\mathbf{f}_{x_1, y_1}, \mathbf{f}_{x_2, y_2}, c_{x_1, y_1}, c_{x_2, y_2}) = -\ln \left[\!\!\left[\mathbf{V}\left(\mathbf{f}_{x_1, y_1}, \mathbf{f}_{x_2, y_2}; \boldsymbol{\theta}^{\mathrm{V}}\right) \right]\!\!\right]_{r_{x_1, y_1, x_2, y_2}}. \quad (3)$$

Compared to a plain classifier assigning a class index to each puzzle, the amount of parameters required by our approach is reduced. Take a 3×3 puzzle as an example, and we assume that F contains D elements. On the one hand, the K-way classifier requires KD parameters (a typical setting [31] is $K = 1,000$) which grows linearly with K. On the other hand, our approach requires $(WH)^2 D$ parameters for the unary terms, and $9D$ parameters for the binary terms. The total number of parameters, $(W^2 H^2 + 9) D$ (e.g., $90D$ for a 3×3 puzzle), is largely reduced and does not increase with K. Consequently, our approach is easier to be applied to the scenario with a larger set of (e.g., all 9! possible) configurations. This advantage is verified in experiments.

Last but not least, there are many other ways of using weak spatial constraints to formulate $S(\mathbf{I}, \mathbf{c})$ – we just provide a practical example.

3.3. Optimization: Iterative Reorganization

We aim at optimizing $S(\mathbf{F}, \mathbf{c})$ with respect to network parameters $\boldsymbol{\theta}^{\mathrm{U}}$, $\boldsymbol{\theta}^{\mathrm{V}}$ and configuration $\mathbf{c}$. However, note that $\mathbf{c}$ is a discrete variable which cannot be optimized by gradient descent. So we apply different strategies in training and testing.

In the training stage, we know the ground-truth configuration $\mathbf{c}^\star$, so the optimization becomes:

$$\arg\min_{\boldsymbol{\theta}^{\mathrm{U}},\boldsymbol{\theta}^{\mathrm{V}}} S(\mathbf{F}, \mathbf{c}^\star). \qquad (4)$$

This is implemented by setting the supervision signal accordingly, *i.e.*, the correct cells are filled up with 1 while others with 0, and using stochastic gradient descent. Note that each unary term depends on the order of input patches. To sample more training data as well as adjust data distribution (explained later), we introduce iteration to the training stage. Denote the input configuration as $\mathbf{c}^{(0)} = \mathbf{c}^\star$, and the corresponding feature as $\mathbf{F}^{(0)}$. In each iteration, with fixed $\boldsymbol{\theta}^{\mathrm{U}}$ and $\boldsymbol{\theta}^{\mathrm{V}}$, we maximize $S(\mathbf{F}, \mathbf{c})$ with respect to $\mathbf{c}$:

$$\mathbf{c}' = \arg\min_{\mathbf{c}} S\left(\mathbf{F}, \mathbf{c}^{(0)}\right), \qquad (5)$$

and use $\mathbf{c}'$ to find the next input $\mathbf{c}^{(1)}$, so that applying $\mathbf{c}'$ to $\mathbf{c}^{(1)}$ obtains $\mathbf{c}^{(0)}$, *e.g.*, if $\mathbf{c}'$ is perfect, then $\mathbf{c}^{(1)}$ corresponds to the original configuration that every patch is placed at the correct position. This process continues until convergence or a maximal number of iterations is reached. The losses with respect to $\boldsymbol{\theta}^{\mathrm{U}}$ and $\boldsymbol{\theta}^{\mathrm{V}}$ are accumulated, averaged, and back-propagated to update these two parameters. The same strategy, iteration, is used at the testing stage to solve jigsaw puzzles, with the only difference that no gradient back-propagation is required.

It remains a problem to solve Eqn (5). This is a combinatoric optimization problem, as $\mathbf{c}$ can only take $(WH)!$ discrete values which indicate the entries in $\mathbf{U}$ and $\mathbf{V}$ that are summed up. There is obviously no closed form solutions to maximize $S(\mathbf{F}, \mathbf{c})$, yet enumerating all $(WH)!$ possibilities is computationally intractable especially when the puzzle size becomes large. A possible solution lies in approximation, which first switches off all binary terms, so that the optimization becomes choosing WH entries from a $WH \times WH$ matrix with a maximal sum, but no two entries can appear in the same row or column (this is a max-cost-max-matching problem, and the best solution $\tilde{\mathbf{c}}$ can be found using the Hungarian algorithm); then enumerates all possibilities within a limited Hamming distance from $\tilde{\mathbf{c}}$ and chooses the one with the best overall cost $S(\mathbf{F}, \mathbf{c})$.

Finally, we discuss strategy of introducing iteration to solve this problem. Mathematically, Eqn (5) is a fixed-point model [26], *i.e.*, the output variable $\mathbf{c}$ also impacts $\mathbf{F}$ and thus $S(\mathbf{F}, \mathbf{c})$, so iteration is considered a regular way of optimizing it. However, the roles played by iteration are different in training and testing. In the **training stage**, after each iteration, we shall expect the configuration to be adjusted closer to the ground-truth. Therefore, if we take the input configuration fed into each round as an individual case, then the distribution of input data is changed by iteration, and the cases that are more similar to the ground-truth are more likely to be sampled. Therefore, in the

testing stage, we can expect the iteration to improve puzzle recognition accuracy, because as the iteration continues, the input puzzle gets closer to the ground-truth by statistics, and our model sees more training data in this scenario and is stronger. We show a typical example in Figure 3, in which we can observe how iteration gradually predicts the correct configuration.

4. Experiments

4.1. Jigsaw Puzzle Recognition

We follow [31] to train and evaluate puzzle recognition on the ILSVRC2012 dataset [43], a subset of the ImageNet database [7]. We train the model using all the 1.3M training images and test it on the validation set with 50K images, both of which do not contain class annotations.

In the training stage, we pre-process the images to prevent the model from being disturbed by pixel-level information. We first determine the size of puzzles, *e.g.*, $W \times H$, and then resize each input image into $85W \times 85H$ and partition it evenly into a $W \times H$ grid. In each 85×85 image, we randomly crop a 64×64 subimage as the patch fed into the puzzle recognition network. To maximally reduce the possibility that low-level information is used, we further horizontally flip each input patch with a probability of 50% and subtract mean value from each channel – we do not perform other data augmentation techniques because they are less likely to appear in real data. In practice, flip augmentation brings consistent accuracy gain to transfer learning tasks though we observe significant accuracy drop in puzzle recognition (see Table 1).

The backbone of our puzzle network is borrowed from two popular architectures, namely, an 8-layer AlexNet [21] and two deep ResNets [16] with 18 and 50 layers. We do not evaluate VGGNet [45] as in [22, 33] because it is more difficult to initialize and produces lower accuracy than ResNets. The outputs of the first layer with a spatial resolution of 1×1 (*i.e.*, *fc6* in AlexNet and *avg-pool* in ResNets) are fed into a $1{,}024$-way fully-connected layer and the output is taken as $\mathbf{f}_{x,y}$, followed by our designed layers for extracting unary and binary terms for puzzle recognition. All these networks are trained from scratch. We use the SGD optimizer and a total of 250K iterations (mini-batches) for AlexNet and 350K for ResNets. Each batch contains 256 puzzles. On four NVIDIA Titan-V100 GPUs, the training times on AlexNet, ResNet18 and ResNet50 are 10, 20 and 60 hours, respectively.

In the testing stage, to reduce randomization factors, we switch off randomization in patch cropping and data augmentation, with each 64×64 patch cropped at the center of the 85×85 fields and not flipped. Results are summarized in Table 1. We first evaluate 3×3 puzzle recognition accuracy. For each image, there are $9! = 362{,}880$ possible puzzles, so

	Setting		Pre-training Options				Puzzle Recognition		Pascal VOC 2007	
ID	Size	Backbone	Label	Unary	Binary	Mirror	Correct	$D \leqslant 2$	Classifi.	Detec.
(a)	3×3	AlexNet	✓				−	−	78.2	56.8
(b)	3×3	AlexNet					−	−	53.3	43.3
(c)	3×3	AlexNet		✓			32.2	48.2	66.6	51.8
(d)	3×3	AlexNet		✓		✓	3.4	15.4	68.1	51.9
(e)	3×3	AlexNet		✓	✓	✓	3.8	17.1	68.3	52.5
(f)	2×2	AlexNet		✓	✓	✓	74.5	90.5	64.2	49.1
(g)	3×3	ResNet18	✓				−	−	84.5	68.3
(h)	3×3	ResNet18					−	−	41.3	24.8
(i)	3×3	ResNet18		✓			44.7	61.5	72.5	58.7
(j)	3×3	ResNet18		✓		✓	5.2	20.4	72.9	58.7
(k)	3×3	ResNet18		✓	✓	✓	5.5	21.0	74.7	58.8
(l)	3×3	ResNet50	✓				−	−	86.4	70.2
(m)	3×3	ResNet50					−	−	46.8	23.5
(n)	3×3	ResNet50		✓			47.3	63.6	72.4	55.2
(o)	3×3	ResNet50		✓		✓	4.9	20.4	73.1	55.5
(p)	3×3	ResNet50		✓	✓	✓	5.2	20.8	75.3	56.2

Competitors with Different Backbones, Pre-training Cues and Settings

Ref.	Year	Backbone	Description of unsupervised training	Classifi.	Detec.
[8]	2015	AlexNet	*Determining the relative spatial position of two patches*	65.3	51.1
[47]	2015	AlexNet	*Unsupervised tracking in videos*	63.1	47.2
[31]	2016	AlexNet	*3×3 jigsaw puzzles with a 1,000-way plain classifier*	67.7	53.2
[22]	2017	ResNet152	*Predicting color from gray-scale intensity*	77.3	−
[32]	2017	AlexNet	*Counting visual primitives in subregions*	67.7	51.4
[4]	2018	AlexNet	*Classifying after clustering iteratively*	73.7	55.4
[13]	2018	AlexNet	*Predicting 2D image rotations*	73.0	54.4
[30]	2018	AlexNet	*[8] with enhancement techniques*	69.6	55.8
[33]	2018	VGGNet16	*[31] with knowledge distillation and noisy patches*	72.5	56.5
[39]	2018	AlexNet	*Predicting surface normal, depth, and instance contour*	68.0	52.6

Table 1. Puzzle recognition and transfer learning accuracy (%). In the pre-training options, "labeled" means to use the annotated ILSVRC2012 training set to pre-train a network. The instances without any ✓ imply that Pascal VOC 2007 tasks are trained from scratch. We also compare with prior approaches, some of which have different knowledge sources, network backbones and training strategies. We report the most powerful network backbone used in each paper. The works with puzzle recognition are highlighted in green.

random guess gives a 0.0003% accuracy. With only unary terms (Eqn 5 can be solved by the Hungarian algorithm), all network backbones achieve over 30% accuracy without mirror augmentation, which shows that weak visual cues can be combined to infer global patch contexts.

On top of this baseline, we investigate the impact of other four options. **First**, adding binary terms consistently improves puzzle recognition accuracy, arguably due to the additional contextual information, which is especially useful in determining the relative position of two neighboring patches. **Second**, mirror augmentation reduces puzzle recognition accuracy dramatically in both training and testing, but as we will see later, this strategy improves the generalization ability of our pre-trained models to other recognition tasks. **Third**, compared with 2×2 puzzles, 3×3 jigsaw puzzles are naturally more difficult to solve, but they also force the model to learn more visual knowledge and thus help transfer learning, as shown in our later discussions. **Fourth**, the above phenomena remain the same

as the network backbone becomes stronger, on which both puzzle recognition and transfer visual recognition becomes more accurate.

As a side comment, we point out that conventional puzzle recognition approaches with plain classification [31, 33] often achieved higher puzzle recognition accuracy in a limited class set. With models trained with our approach (Line (e) in Table 1) we enumerate the 1,000 classes generated with algorithm provided by [31] and find the maximal $S(\mathbf{F}, \mathbf{c})$, so as to mimic the behavior of plain classification. Our models with AlexNet reports a 60.2% puzzle recognition accuracy which is lower than 71% reported in [31]. However, our approach enjoys better transfer ability, as we will see in later experiments. In addition, the performance of [31] degenerates with increased puzzle size, as the fraction of explored puzzles becomes smaller, yet the weakness of ignoring underlying relationship between different configurations becomes more significant and harmful. From this perspective, the advantage of solving arbitrary puzzles

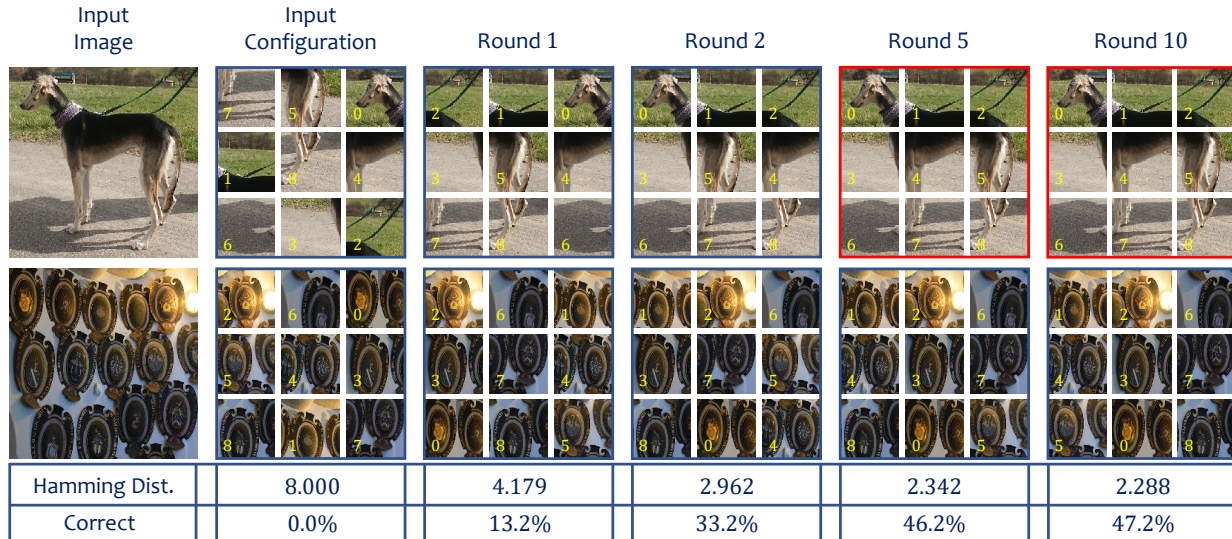|	Input Image	Input Configuration	Round 1	Round 2	Round 5	Round 10
Hamming Dist.		8.000	4.179	2.962	2.342	2.288
Correct		0.0%	13.2%	33.2%	46.2%	47.2%

Figure 3. Two examples of different difficulties in iterative puzzle recognition (best viewed in color). Each digit to the lower-left corner of each patch is the corresponding patch ID. For each round, we also report puzzle recognition statistics **over the entire testing set**.

Unary	Binary	Mirror	conv1	conv2	conv3	conv4	conv5
✓			18.4	29.0	32.6	31.4	27.5
✓		✓	18.0	29.0	33.5	32.8	29.7
✓	✓	✓	18.3	28.8	34.4	34.1	30.4
Noroozi *et al.* [31]			18.2	28.8	34.0	33.9	27.1
✓			21.3	30.9	33.5	33.1	29.8
✓		✓	20.9	31.0	34.0	33.6	30.6
✓	✓	✓	21.6	30.8	34.7	34.2	31.8
Noroozi *et al.* [31]			23.0	31.9	35.0	34.2	29.3

Table 2. ILSVRC2012 (top) and Places205 (bottom): classification accuracy with linear classifiers on top of frozen convolutional layers of AlexNet.

becomes clearer. The same phenomenon also happens in 3D puzzles (Section 4.3).

Some statistics for our model with ResNet50 (Line (p) in Table 1) as well as two typical examples are shown in Figure 3 (one is difficult and not solved). We can observe how the disordered patches are reorganized with weak spatial cues throughout an iterative process. As an ablation study, we experiment with fewer numbers of maximal iterations, namely 1, 5 and 10 instead of 20, but achieve lower accuracies in both puzzle recognition and transfer learning tasks. This justifies our hypothesis that iteration, together with weak spatial cues, provides a mild way of unsupervised learning, which better fits state-of-the-art deep networks.

4.2. Transfer Learning Performance

Next, we investigate how well our models pre-trained on puzzle recognition transfer to other visual recognition tasks. Following the conventions [33, 4], we evaluate classification and detection tasks on the Pascal VOC 2007 dataset [11]. All pre-trained networks undergo a stan-dard fine-tuning flowchart, with a plain classifier and Fast-RCNN [14] being used as network heads, respectively. We do not lock any layers in our network, because this often leads to worse transfer performance as shown in prior approaches [31, 4, 13].

Results are summarized in Table 1. We can observe some interesting phenomena. **First**, transfer recognition performance goes up with the power of network backbones, which shows the ability of our approach to tap the potential of deep networks. **Second**, both unary and binary terms contribute to transfer accuracy and they are complementary. **Third**, mirror augmentation harms puzzle recognition but improves transfer learning, because it alleviates the chance that deep networks borrow low-level pixel continuity in solving the jigsaw puzzles which falls into the category of over-fitting and helps transfer recognition very little.

Here is a side note. It was suggested in [31] that forcing the network to discriminate very similar puzzles (*e.g.*, only a pair of patches are reversed) often leads to accuracy drop because the model can focus too much on local patterns. In the context of using AlexNet to solve 3×3 puzzles, we study different numbers of configurations, *i.e.*, 1% (3,629), 10% (36,288) and all (9! = 362,880) possible puzzles. We find that our approach reports the best transfer accuracy at the last option, while using smaller numbers of configurations leads to slightly worse performance. Hence, we make the following conjecture: it is indeed the larger number of parameters in a plain classifier, rather than solving very similar puzzles, that causes transfer performance drop.

In the last part, we evaluate the quality of features extracted from the pre-trained models directly. We implement the standard experiments on the ILSVRC2012 dataset [43] and Places205 [53] with a linear classifier on top of frozen

convolutional layers of AlexNet. Results are summarized in Table 2. Our approach, particularly with binary terms and mirror augmentation, shows higher numbers with *conv5* features. This is in line with our motivation, namely, solving arbitrary puzzles with weak spatial constraints indeed improves mid-level representation learning.

4.3. Generalization to 3D Networks

Finally, we apply our model to a 3D visual recognition task. We study medical imaging analysis, an important prerequisite for computer-assisted diagnosis (CAD). Most medical data are volumetric, and researchers have proposed some 3D network architectures [5, 28]. Compared to 2D networks [40, 51], 3D networks enjoy the benefit of seeing more contextual information, but still suffer the drawback of missing a pre-trained model. Due to the common situation that the amount of training data is limited, these 3D networks often have a relatively unstable training process and sometimes this downgrades their testing accuracy [49].

Our approach provides a solution for initializing 3D networks with jigsaw puzzles. We investigate the NIH pancreas segmentation dataset [41], which contains 82 cases. We partition it into 4 folds (around 20 cases in each fold), use three of them to train a segmentation model and test it on the remaining one. To construct jigsaw puzzles, we either directly use the training samples in the NIH dataset, or refer to another public dataset named Medical Segmentation Decathlon (MSD)[3] – the *pancreas tumour* subset with 282 training cases. For all the data used for jigsaw puzzles, we do not use any pixel-level annotations though they are provided. We randomly crop $120 \times 120 \times 120$ volumes within each case, and cut it evenly into two puzzle sizes, namely, $2 \times 2 \times 2$ pieces with a $48 \times 48 \times 48$ subvolume cropped within each cell, or $3 \times 3 \times 3$ pieces with a $32 \times 32 \times 32$ subvolume cropped within each cell. A typical example is shown in Figure 1. We randomly disorder these patches using all 8! or 27! possible configurations, and the task is to recover the original configuration. We use VNet [28] as the baseline (only the down-sampling layers are used in this stage), and compute the unary terms in an 8×8 or 27×27 matrix. We switch off the binary terms based on the consideration that one patch has 26 neighbors in the 3D space which makes prediction over-complicated.

Now we recover the complete VNet structure with randomly-initialized up-sampling layers and start training on the NIH training set (62 cases) and its subsets. Results are shown in Table 3 revealing some useful knowledge. **First**, pre-training on jigsaw puzzles helps segmentation especially in the scenarios of fewer training data. **Second**, visual knowledge learned in this manner can transfer across different datasets regardless of the different distributions in intensity caused by the scanning device. **Third**, the

---

[3]http://medicaldecathlon.com/

Data	Scratch	Pre-trained on NIH		Pre-trained on MSD	
		$2 \times 2 \times 2$	$3 \times 3 \times 3$	$2 \times 2 \times 2$	$3 \times 3 \times 3$
10%	65.52	69.36	70.80	68.44	72.24
20%	74.78	76.30	76.50	76.58	77.80
100%	80.96	79.88	81.68	81.48	82.33

Table 3. Pancreas segmentation accuracy (DSC, %) with different amounts of training data and different initialization techniques. In each group, the accuracy is averaged over 20 testing cases.

model pre-trained on NIH with $2 \times 2 \times 2$ puzzles performs worse than the from-scratch one when tested on 100% data. We conjecture that solving such puzzles provides a better initialization, but the domain gap between solving puzzles and semantic segmentation cancels out the positive effect of initialization. This raises the necessity of solving more complex $3 \times 3 \times 3$ puzzles which agrees with our motivation. **Fourth**, constructing larger and thus more difficult puzzles improves the basic ability of networks. This shows the value of our research – it is unlikely for the baseline approach to sufficiently explore the space of $3 \times 3 \times 3$ puzzles, which has $27! \approx 1.1 \times 10^{28}$ different configurations. Sampling 1,000 configurations with the greedy algorithm described in [31] downgrades segmentation accuracy to 73.63% and 80.38% with 20% and 100% data, respectively.

5. Conclusions

This work generalizes the framework of jigsaw puzzle recognition which was previously studied in a constrained case. To this end, we change the network head from a plain K-way classifier to a combinatoric optimization problem which uses both unary and binary weak spatial cues. This strategy reduces the number of learnable parameters in the model, and thus alleviates the risk of over-fitting. The increased flexibility of pre-training allows us to apply our approach to a wide range of transfer learning tasks, including directly using it for feature extraction, and generalizing it to the 3D scenarios to provide an initialization for other tasks, *e.g.*, medical imaging segmentation.

Our study reveals the ease and benefits of learning to recognize weak visual cues in unsupervised learning, in which the key problem often lies in finding a compact way of representing knowledge, *e.g.*, decomposing the entire puzzle into unary and binary terms. We point out that the exploration of unsupervised learning is still far from the end. In the future, we will also apply our method to less structured data such as graphs [20] and more structured data such as videos [19], and explore its ability of learning visual knowledge in an unsupervised manner.

References

[1] J. Aghajanian, J. Warrell, S. J. Prince, P. Li, J. L. Rohn, and B. Baum. Patch-based within-object classification. In *International Conference on Computer Vision*, 2009.

[2] B. Brattoli, U. Buchler, A.-S. Wahl, M. E. Schwab, and B. Ommer. Lstm self-supervision for detailed behavior analysis. In *Computer Vision and Pattern Recognition*, 2017.

[3] U. Buchler, B. Brattoli, and B. Ommer. Improving spatiotemporal self-supervision by deep reinforcement learning. In *European Conference on Computer Vision*, 2018.

[4] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018.

[5] O. Cicek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2016.

[6] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European Conference on Computer Vision*, 2010.

[7] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 2009.

[8] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision*, 2015.

[9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, 2014.

[10] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[12] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *Computer Vision and Pattern Recognition*, 2017.

[13] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

[14] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision*, 2015.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, 2016.

[17] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition*, 2017.

[18] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *European Conference on Computer Vision*, 2016.

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition*, 2014.

[20] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[22] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *Computer Vision and Pattern Recognition*, 2017.

[23] Q. V. Le. Building high-level features using large scale unsupervised learning. In *International Conference on Speech and Signal Processing*, 2013.

[24] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[25] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang. Unsupervised representation learning by sorting sequences. In *International Conference on Computer Vision*, 2017.

[26] Q. Li, J. Wang, D. Wipf, and Z. Tu. Fixed-point model for structured labeling. In *International Conference on Machine Learning*, 2013.

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, 2015.

[28] F. Milletari, N. Navab, and S. A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *International Conference on 3D Vision*, 2016.

[29] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, 2016.

[30] T. N. Mundhenk, D. Ho, and B. Y. Chen. Improvements to context based self-supervised learning. In *Computer Vision and Pattern Recognition*, 2018.

[31] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, 2016.

[32] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. In *International Conference on Computer Vision*, 2017.

[33] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *Computer Vision and Pattern Recognition*, 2018.

[34] G. Papandreou, L. C. Chen, K. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a dcnn for semantic image segmentation. In *International Conference on Computer Vision*, 2015.

[35] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. L. Yuille. Deep co-training for semi-supervised image recognition. In *European Conference on Computer Vision*, 2018.

[36] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

[37] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition*, 2014.

[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[39] Z. Ren and Y. J. Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *Computer Vision and Pattern Recognition*, 2018.

[40] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[41] H. R. Roth, L. Lu, A. Farag, H. Shin, J. Liu, E. B. Turkbey, and R. M. Summers. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[42] F. Rousseau, P. A. Habas, and C. Studholme. A supervised patch-based approach for human brain labeling. *IEEE transactions on medical imaging*, 30(10):1852–1862, 2011.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[44] R. Santa Cruz, B. Fernando, A. Cherian, and S. Gould. Deeppermnet: Visual permutation learning. In *Computer Vision and Pattern Recognition*, 2017.

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, 2015.

[47] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *International Conference on Computer Vision*, 2015.

[48] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

[49] Y. Xia, L. Xie, F. Liu, Z. Zhu, E. K. Fishman, and A. L. Yuille. Bridging the gap between 2d and 3d organ segmentation with volumetric fusion net. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2018.

[50] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *Computer Vision and Pattern Recognition*, 2016.

[51] Q. Yu, L. Xie, Y. Wang, Y. Zhou, E. K. Fishman, and A. L. Yuille. Recurrent saliency transformation network: Incorporating multi-stage visual cues for small organ segmentation. In *Computer Vision and Pattern Recognition*, 2018.

[52] Z. Zhang, C. Xie, J. Wang, L. Xie, and A. L. Yuille. Deepvoting: An explainable framework for semantic part detection under partial occlusion. In *Computer Vision and Pattern Recognition*, 2018.

[53] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, 2014.

[54] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, 2017.