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

Chen Wei1, Lingxi Xie2,3†, Xutong Ren1, Yingda Xia2, Chi Su4, Jiaying Liu1, Qi Tian3, Alan L. Yuille2
1 Peking University  2 Johns Hopkins University  3 Noah’s Ark Lab, Huawei Inc.  4 Kingsoft Cloud

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 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 assign 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 $3 \times 3$ 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., $3 \times 3$) of patches and then reordered. 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 $I$ be an image, which is partitioned into $W \times H$ patches. Each patch, denoted as $i_{x,y}$ ($0 \leq x < W, 0 \leq y < H$), 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 reordered, and we use $c^*_{x,y}$ to denote the ID owned by the patch that currently occupies the $(x, y)$ position. All $c^*_{x,y}$ values compose a configuration, denoted as $c^* = (c^*_{x,y})_{x=0,y=0}^W$. There are in total $(WH)!$ different configurations, composing the configuration set $C$ that $|C| = (WH)!$.

Our goal is to predict the correct configuration $c^* \in C$. For this purpose, a network structure with two parts was constructed [31]. The network backbone $M^B : f_{x,y} = f(i_{x,y}; \theta^B)$ is built upon each individual patch, and outputs a set of features for the network head $M^H : c = g(F; \theta^H)$ to produce the final output $c = (c_{x,y})_{x=0,y=0}^W$, where $F = (f_{x,y})_{x=0,y=0}^W$ is the ordered concatenation of patch features. In practice, $f(\cdot; \theta^B)$ is often borrowed from existing network architectures [21, 45, 16], while $g(\cdot; \theta^H)$ 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 $C$ using a greedy algorithm to guarantee the Hamming distance between any two configurations is sufficiently large. Then, $f(\cdot; \theta^H)$ 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 over-fitting\(^1\), 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 $S(I, c)$ which outputs a cost that patch $i_{x,y}$ or equivalently feature $f_{x,y}$ is located at position $c_{x,y}$, and thus the most probable configuration is determined by $\arg \max_c \{S(I, c)\}$. $S(I, c)$ 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,\(^1\)

$$S(I, c) = S(F, c) = \sum_{(x,y)} p_1(f_{x,y}, c_{x,y} | F) + \sum_{(x_1,y_1),(x_2,y_2) \neq (x_1,y_1)} p_2(f_{x_1,y_1}, f_{x_2,y_2}, c_{x_1,y_1}, c_{x_2,y_2}). \quad (1)$$

Here, $p_1(f_{x,y}, c_{x,y} | F)$ is a unary term which measures how likely that patch $f_{x,y}$ is located at position $c_{x,y}$, and $p_2(f_{x_1,y_1}, f_{x_2,y_2}, c_{x_1,y_1}, c_{x_2,y_2})$ is a binary term measures how likely that patches $f_{x_1,y_1}$ and $f_{x_2,y_2}$ have the spatial relationship indicated by $c_{x_1,y_1}$ and $c_{x_2,y_2}$. Each unary term is computed based on $F$, 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 $C$, i.e., $K = 9! = 362,880$, which implies that the performance drop may due the large number of parameters.
Each Two Features
Concatenation
Relative Position
Vector $V$

1 × 9

c′

Predicted
Configuration $c'$

All Features
Concatenation
Configuration Matrix $U$

9 × 9

Feature
Reorganization

Input Image $I$

Input
Configuration $c^*$

Backbone $\theta^B$

Feature $f_{x,y}$

... ... ... ...

Sharing weights
concatenation
softmax

Max-Cost
Max-Matching

Binary Term $p_2$

Cost Function $S(I, c)$

Unary Term $p_1$

Feature $f_{x,y}$ 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 $U$ with $WH \times WH$ elements, each of which, $[U]_{a,c}$, 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 $F$ and these $(WH)^2$ elements, parameterized by $\theta^U$. 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:

$$p_1(f_{x,y}, c_{x,y}, F) = -\ln \left[ U \left( F; \theta^U \right) \right]_{a,x,y}^{c,x,y}. \quad (2)$$

For each binary term involving $f_{x_1,y_1}$ and $f_{x_2,y_2}$, we build another mapping from these two vectors to a 9-dimensional vector, with each index indicating the probability that the spatial relationship of $f_{x_1,y_1}$ and $f_{x_2,y_2}$ 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 $f_{x_2,y_2} \oplus f_{x_2,y_2}$ (denotes concatenation) and a 9-dimensional vector parameterized by $\theta^V$ followed by a softmax activation over these 9 numbers. We denote $r_{x_1,y_1,x_2,y_2} = r(c_{x_1,y_1}, c_{x_2,y_2}) \in \{0, 1, \ldots, 8\}$ as the relative position type between $f_{x_1,y_1}$ and $f_{x_2,y_2}$, so that we can write the binary term as:

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

Compared to a plain classifier assigning a class index to each puzzle, the amount of parameters required by our approach is reduced. Take a $3 \times 3$ 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, $(W^2H^2 + 9) D$ (e.g., $90D$ for a $3 \times 3$ 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 $S(I, c)$ – we just provide a practical example.

### 3.3. Optimization: Iterative Reorganization

We aim at optimizing $S(F, c)$ with respect to network parameters $\theta^U, \theta^V$ and configuration $c$. However, note that $c$ 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 $c^*$, so the optimization becomes:

$$\arg\min_{\theta^U, \theta^V} S(F, c^*). \quad (4)$$

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 $c^{(0)} = c^*$, and the corresponding feature as $F^{(0)}$. In each iteration, with fixed $\theta^U$ and $\theta^V$, we maximize $S(F, c)$ with respect to $c$:

$$c' = \arg\min_{c} S(F, c^{(0)}), \quad (5)$$

and use $c'$ to find the next input $c^{(1)}$, so that applying $c'$ to $c^{(1)}$ obtains $c^{(0)}$, e.g., if $c'$ is perfect, then $c^{(1)}$ 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 $\theta^U$ and $\theta^V$ 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 $c$ can only take $(WH)!$ discrete values which indicate the entries in $U$ and $V$ that are summed up. There is obviously no closed form solutions to maximize $S(F, c)$, 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 $\tilde{c}$ can be found using the Hungarian algorithm); then enumerates all possibilities within a limited Hamming distance from $\tilde{c}$ and chooses the one with the best overall cost $S(F, c)$.

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 $c$ also impacts $F$ and thus $S(F, c)$, 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 $85 \times 85$ image, we randomly crop a $64 \times 64$ 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 $1 \times 1$ (i.e., $fc6$ in AlexNet and $avg\text{-}pool$ in ResNets) are fed into a 1,024-way fully-connected layer and the output is taken as $f_{x,y}$, 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 $64 \times 64$ patch cropped at the center of the $85 \times 85$ fields and not flipped. Results are summarized in Table 1. We first evaluate $3 \times 3$ puzzle recognition accuracy. For each image, there are $9! = 362,880$ possible puzzles, so
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 $2 \times 2$ puzzles, $3 \times 3$ 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 $S(F, c)$, 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

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Setting</th>
<th>Pre-training Options</th>
<th>Puzzle Recognition</th>
<th>Pascal VOC 2007</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>$3 \times 3$ AlexNet</td>
<td>✓</td>
<td>32.2</td>
<td>84.5</td>
</tr>
<tr>
<td>(b)</td>
<td>$3 \times 3$ AlexNet</td>
<td>✓</td>
<td>44.7</td>
<td>41.3</td>
</tr>
<tr>
<td>(c)</td>
<td>$3 \times 3$ AlexNet</td>
<td>✓</td>
<td>5.2</td>
<td>72.5</td>
</tr>
<tr>
<td>(d)</td>
<td>$3 \times 3$ AlexNet</td>
<td>✓</td>
<td>5.5</td>
<td>72.9</td>
</tr>
<tr>
<td>(e)</td>
<td>$3 \times 3$ AlexNet</td>
<td>✓</td>
<td>47.3</td>
<td>46.8</td>
</tr>
<tr>
<td>(f)</td>
<td>$3 \times 3$ AlexNet</td>
<td>✓</td>
<td>1</td>
<td>72.4</td>
</tr>
<tr>
<td>(g)</td>
<td>$3 \times 3$ ResNet18</td>
<td>✓</td>
<td>5.2</td>
<td>57.3</td>
</tr>
<tr>
<td>(h)</td>
<td>$3 \times 3$ ResNet18</td>
<td>✓</td>
<td>6.4</td>
<td>70.2</td>
</tr>
<tr>
<td>(i)</td>
<td>$3 \times 3$ ResNet50</td>
<td>✓</td>
<td>71%</td>
<td>71%</td>
</tr>
<tr>
<td>(j)</td>
<td>$3 \times 3$ ResNet50</td>
<td>✓</td>
<td>71%</td>
<td>71%</td>
</tr>
<tr>
<td>(k)</td>
<td>$3 \times 3$ ResNet50</td>
<td>✓</td>
<td>71%</td>
<td>71%</td>
</tr>
<tr>
<td>(l)</td>
<td>$3 \times 3$ ResNet50</td>
<td>✓</td>
<td>71%</td>
<td>71%</td>
</tr>
<tr>
<td>(m)</td>
<td>$3 \times 3$ ResNet50</td>
<td>✓</td>
<td>71%</td>
<td>71%</td>
</tr>
<tr>
<td>(n)</td>
<td>$3 \times 3$ ResNet50</td>
<td>✓</td>
<td>71%</td>
<td>71%</td>
</tr>
<tr>
<td>(o)</td>
<td>$3 \times 3$ ResNet50</td>
<td>✓</td>
<td>71%</td>
<td>71%</td>
</tr>
<tr>
<td>(p)</td>
<td>$3 \times 3$ ResNet50</td>
<td>✓</td>
<td>71%</td>
<td>71%</td>
</tr>
</tbody>
</table>

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 plain classification with knowledge distillation and noisy patches.

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 $S(F, c)$, 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
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 standard 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 $3 \times 3$ 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.
convolutional layers of AlexNet. Results are summarized in Table 2. Our approach, particularly with binary terms and mirror augmentation, shows higher numbers with \textit{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 × 120 × 120 volumes within each case, and cut it evenly into two puzzle sizes, namely, 2 × 2 × 2 pieces with a 48 × 48 × 48 subvolume cropped within each cell, or 3 × 3 × 3 pieces with a 32 × 32 × 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. \textbf{First}, pre-training on jigsaw puzzles helps segmentation especially in the scenarios of fewer training data. \textbf{Second}, visual knowledge learned in this manner can transfer across different datasets regardless of the different distributions in intensity caused by the scanning device. \textbf{Third}, the model pre-trained on NIH with 2 × 2 × 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 × 3 × 3 puzzles which agrees with our motivation. \textbf{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 × 3 × 3 puzzles, which has 27! ≈ 1.1 × 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 \textit{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.

### Acknowledgments

This paper was supported by ONR N00014-15-1-2356. We thank Weichao Qiu, Chenxi Liu, Zhuotun Zhu, Siyuan Qiao, Yutong Bai and Angtian Wang for instructive discussions.

\[^3\]http://medicaldecathlon.com/


