JigsawGAN: Self-supervised Learning for Solving Jigsaw Puzzles with Generative Adversarial Networks

Ru Li, *Student Member, IEEE* Shuaicheng Liu, *Member, IEEE,* Guangfu Wang, Guanghui Liu, *Senior Member, IEEE,* Bing Zeng *Fellow, IEEE,*

拼图游戏

***Abstract*—The paper proposes a solution based on Generative Adversarial Network (GAN) for solving jigsaw puzzles. The problem assumes that an image is cut into equal square pieces, and asks to recover the image according to pieces information. Conventional jigsaw solvers often determine piece relationships based on the piece boundaries, which ignore the important semantic information. In this paper, we propose JigsawGAN, a GAN-based self-supervised method for solving jigsaw puzzles with unpaired images (with no prior knowledge of the initial images). We design a multi-task pipeline that includes, (1) a classification branch to classify jigsaw permutations, and (2) a GAN branch to recover features to images with correct orders. The classification branch is constrained by the pseudo-labels generated according to the shuffled pieces. The GAN branch concentrates on the image semantic information, among which the generator produces the natural images to fool the discriminator with reassembled pieces, while the discriminator distinguishes whether a given image belongs to the synthesized or the real target manifold. These two branches are connected by a flow-based warp that is applied to warp features to correct order according to the classification results. The proposed method can solve jigsaw puzzles more efficiently by utilizing both semantic information and edge information simultaneously. Qualitative and quantitative comparisons against several leading prior methods demonstrate the superiority of our method.**

***Index Terms*—Solving jigsaw puzzles, generative adversarial network, self-supervised learning**

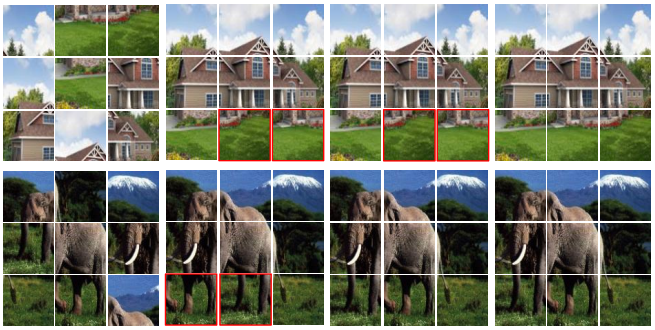
判断真假

I. Introduction

Solving jigsaw puzzle is a challenging mathematical and engineering problem that involves researches in computer science, mathematics and engineering. It abstracts a range of computational problems in which a set of unordered fragments should be organized into their original combination. Numerous applications are subsequently raised, such as reassembling archaeological artifacts [1]–[5], recovering shredded documents or photographs [6]–[9] and genome biology [10]. The automatic solution of puzzles, without having any information on the underlying image, proved NP-complete [11], [12]. Generally, this task relies on computer vision algorithms, such as contours or features detection [13]. Recent development of deep learning opens bright perspectives for finding better reassembles more efficiently.

Ru Li, Shuaicheng Liu, Guanghui Liu and Bing Zeng are with School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China.

Guangfu Wang is with Megvii Technology, Chengdu 610000, China.



(a) Inputs (b) Results of [14] (c) Results of [15] (d) Ours

Fig. 1. Comparisons with two recent jigsaw puzzle solvers. (a) Input images. (b) Results of Paumard *et al.*'s method [14]. (c) Results of Huroyan *et al.*'s method [15]. (d) Our results. Paumard *et al.*'s method [14] cannot handle these two cases and Huroyan *et al.*'s method [15] fails to solve the house example. The proposed JigsawGAN works well on these cases due to the concentration of edge and semantic information simultaneously.

The first algorithm that attempted to automatically solve general puzzles was introduced by Freeman and Gardner [16] in 1964. The original approach was designed to solve puzzles with 9 pieces by only considering the geometric shapes of the pieces. Various puzzle reconstruction methods are then proposed. They obey a basic operation that the unassigned piece should link to an existing piece by finding its best neighbors according to some affinity function [1], [17], [18] or matching contours [8], [19]–[22]. These algorithms are slow, which repeat themselves until a proper solution is achieved and all parts are put in place. Moreover, they may fail because the unpredictable continuation appears between true neighbors. To alleviate the problem, some subsequent works are proposed to avoid checking all possible permutations of piece placements [23]–[27]. However, they mainly concentrate on the edge information (four boundaries of each piece), whereas ignoring the semantic information which is useful for image understanding. Recently, some methods based on convolutional neural network (CNN) are introduced for predicting the permutations by neural optimization [28], [29] or predicting the relative position of a fragment with respect to another [14], [30], [31]. Similar to aforementioned methods, these algorithms do not consider the semantic content information and only rely on learning the position of each piece.

In this paper, we propose JigsawGAN, a self-supervised method that combines the edge information of each piece

and the semantic information of the global image to solve the jigsaw puzzles, which can better describe the procedure that humans solve jigsaw puzzles. The proposed pipeline consists of two branches, a classification branch (Fig. 2 (b)) and a GAN branch (Fig. 2 (d), (e) and a discriminator), the former one classifies the jigsaw permutations, and the latter one recovers features to images with correct sequences. The two branches are connected by the encoder (Fig. 2 (a)) and a flow-based warp (Fig. 2 (c)).

In the classification branch, a flow-based warp is applied to warp encoder features to the predicted position according to the classification results. Directly placing shuffled features to new positions is problematic because it will block gradient back propagation for the classification network. Therefore, we introduce a novel flow-based warp to reorganize the features along with correct gradient propagation. The GAN branch consists of the decoder and the discriminator, the former one recovers the features to the images and the latter one aims to classify an image that belongs to the generated or the real dataset. The discriminator can be regarded as the process that humans judge whether the reorganization of pieces is correct. The judgment process is essential to equip the pipeline with the capability of high-level image understanding, together with low-level edge clues, to yield a better jigsaw puzzle solver. Corresponding loss terms are applied for concentration on different information, among which a novel edge loss is designed for constraining the piece's edge and GAN loss is used for distinguishing the semantic information. The inputs are obtained by decomposing the source images using a regular $n \times n$ grid of pieces, which are then shuffled. Note that, the definition of 'self-supervised' in our method means we generate pseudo-labels, called reference labels, according to the shuffled input images, whereas conventional self-supervised methods [32], [33] exploit labeling that comes for 'free' with the data. Detailed descriptions are illustrated in Section III-C1.

The proposed JigsawGAN is compatible with existing methods, which can be considered as an improvement technique of existing algorithms, creating new state-of-the-art performances. The reference labels are designed to guide the classification network to converge, while the GAN branch pushes the encoder to generate more informative features, and further obtaining higher classification accuracy. The proposed method is able to improve the reorganization accuracy of existing methods if we select their results as our reference labels. Detailed descriptions and experiments are presented in Section IV-D3.

Fig. 1 shows the comparisons with two recent methods [14], [15]. Paumard *et al.* proposed a CNN-based method to detect the neighbor pieces and applied shortest path optimization to recover the images [14]. Huroyan *et al.* applied graph connection laplacian to determine the edge relationships [15]. Fig. 1 shows two hard examples, in which three grassland pieces in the house example and two leg pieces in the elephant example are easy to be confused. Paumard *et al.*'s method [14] cannot handle the two cases and Huroyan *et al.*'s method [15] fails to solve the house example. In contrast, our proposed JigsawGAN works well on these cases due to the concentration not only on edges, but also on semantic clues. We conduct both visual and quantitative comparisons against several typical methods to validate the superior performance of JigsawGAN. In quantitative assessments, the proposed method achieves the best scores on average.

Overall, the main contributions are summarized as follows:

- We propose a GAN-based architecture to solve jigsaw puzzles, in which both edge and semantic clues are fused for the inference.
- We introduce a flow-based warp in the classification branch to reorganize feature pieces where gradient can be back propagated during the training.
- We provide qualitative and quantitative comparisons with several state-of-the-art methods using JigsawGAN, demonstrating its superiority over existing methods.

II. Related works

Introduced by Freeman and Gardner [16], the puzzle solving problems has been studied by many researchers, even though Demaine *et al.* discovered that puzzle assembling is an NP-hard problem if the dissimilarity is unreliable [12].

A. Solving Square-piece Puzzles

Most jigsaw solvers assume that the input consists of equal-size squared pieces of an image. The first work was proposed by [34], where a greedy algorithm and a benchmark were proposed. They presented a probabilistic solver to achieve approximated puzzle reconstruction via a graphical model. Pomeranz *et al.* evaluated some compatibility metrics and proposed a new compatibility metrics to predict the chances of two given parts to be neighbors [18]. Gallagher *et al.* divided the square jigsaw puzzle problems into three types [23]. 'Type 1' puzzle means the orientation of each jigsaw piece is known, and only the location of each piece in the completed puzzle is unknown. 'Type 2' puzzle is defined as non-overlapping square-piece jigsaw puzzle with unknown dimension, unknown piece rotation, and unknown piece position. Meanwhile, the global geometry and position of every jigsaw piece in 'Type 3' puzzle are known, and only the orientation of each piece is unknown. Gallagher *et al.* solved 'Type 1' and 'Type 2' puzzle problems through the minimum spanning tree (MST) algorithm constrained by geometric consistency between pieces. A Markov Random Field (MRF)-based algorithm is also proposed to solve the 'Type 3' puzzle.

Some subsequent works were proposed to solve 'Type 1' puzzle problem. Andaló *et al.* mapped the problem of solving a jigsaw puzzle to the maximization of a constrained quadratic function [24]. Yu *et al.* applied linear programming to exploit all the pairwise matches, and computed the position of each component [35]. There are also some methods aimed to solve more complicated 'Type 2' puzzle problem. Son *et al.* introduced loop constraints for assembling non-overlapping jigsaw puzzles where the rotation and the position of each piece are unknown [25]. Huroyan *et al.* proposed to utilize graph connection laplacian to recover the rotations of the pieces when both shuffles and rotations are unknown [15]. Some variants were investigated subsequently, including handling

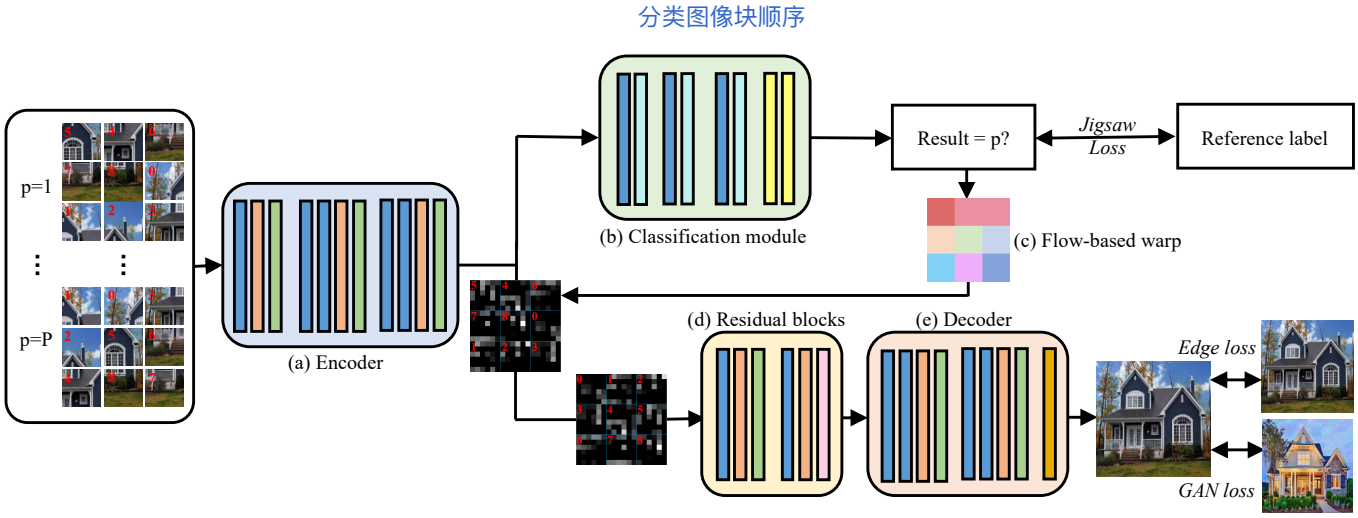


Fig. 2. The pipeline exhibits the architecture of the proposed JigsawGAN, which consists of two streams operating different functions. **The top stream** classifies the jigsaw permutations. In turn, **The bottom stream** recovers features to images. The two streams are connected by the encoder and a flow-based warp. The encoder features are used for classification and decoder simultaneously, and the flow-based warp is applied to warp encoder features to correct order according to the classification results. The encoder processes the inputs through down-convolution layers, and the decoder recovers the warped features by residual blocks and up-convolution layers so as to generate correct reorganization images.

puzzles with missing pieces [36] and solving puzzles with eroded boundaries [14]. Paikin *et al.* calculated dissimilarity between pieces and then proposed a greedy algorithm for handling puzzles of unknown size and with missing entries [36]. Bridger *et al.* proposed a GAN-based architecture to recover the eroded regions and reassembled the images using greedy algorithms [31]. Our method concentrates on solving 'Type 1' non-overlapping square-piece jigsaw puzzles.

Recently, some deep learning (DL)-based methods are proposed to handle puzzle problems with 9 input pieces. Although solving jigsaw puzzles with 9 pieces is relatively simple than reassembling puzzles with hundreds of pieces in conventional methods, the robustness of solving jigsaw puzzles with 9 pieces in DL-based methods has been greatly improved. Dery *et al.* [29] applied a pre-trained VGG model as feature extractor and corrected the order using a pointer network [37]. Paumard *et al.* utilized a CNN-based method to detect the neighbor pieces and used the shortest path optimization to recover the eroded images [14]. These methods obey the conventional rule that applying the neural network to determine piece relationship first and then using optimization techniques to reorganize the pieces. Their performance is also limited by the first step due to the aperture problem based only on the local pieces. We proposed a GAN-based approach to involve global semantic information apart from local pieces for better accuracy.

B. Pre-training Jigsaw Puzzle Solvers

There are numerous self-supervised methods solve jigsaw puzzles as pre-text tasks and transfer the pre-trained parameters to other visual recognition tasks [38]. Noroozi *et al.* introduced a context-free network (CFN) to separate the pieces in convolutional process. Their main architecture focused on a subset of possible permutations involving all the image tiles and solved a classification problem [39]. Santa *et al.* proposed to handle the whole set by approximating the permutation matrix and solving a bi-level optimization problem to recover the right ordering [40]. The above methods tackle the problem by dealing with the separate pieces and then finding a way to recombine them. Carlucci *et al.* proposed JiGen to train the jigsaw classifier and object classifier simultaneously. They focused on domain generalization tasks by considering that the jigsaw puzzle solver can improve semantic understanding [41]. Du *et al.* combined the jigsaw puzzle and progressive training to optimize the fine-grained classification by learning which granularities are the most discriminative and how to fuse information cross multi-granularity [42]. However, solving jigsaw puzzle tasks in these methods are supervised, which depend heavily on the training data. In this paper, we propose a 'self-supervised' approach to solve jigsaw puzzles, where the ground-truth of jigsaw puzzle tasks is unavailable. The definition of 'self-supervised' in our method means that we generate pseudo-labels to constrain the classification network based on the shuffled input images.

III. Method

We propose a GAN-based architecture for solving jigsaw puzzles with unpaired datasets. The generator G learns the mapping function between different domains, while the discriminator D aims to optimize G by distinguishing source domain images from the generated ones. Specifically, we generate a wide diversity of shuffled images $\{x_i\}_{i=1,\dots,N} \in X$ as source domain data, and a collection of natural images $\{y_j\}_{j=1,\dots,M} \in Y$, as the target domain data. The data distributions of the two domains are denoted as $x \sim p_{\text{data}}(x)$ and $y \sim p_{\text{data}}(y)$, respectively.

A. Network Architecture

We present the classification network C, generator network G in Fig. 2, where C, G and the discriminator D are simultaneously optimized during the training process. Their detailed layer configurations are presented in Table I. Given shuffled

(a) Encoder

Conv			BN	Activation
Kernel	Stride	Channel	Channel	
7	1	64	64	ReLU
3	2	128	-	-
3	1	128	128	ReLU
3	2	256	-	-
3	1	256	256	ReLU

(b) Classification module

Conv			Activation	Max pooling	FC
Kernel	Stride	Channel		Stride	Channel
3	2	256	ReLU	2	-
3	2	384	ReLU	2	-
3	2	384	ReLU	2	-
-	-	-	-	-	4096
-	-	-	-	-	P

(c) Residual blocks and decoder

Conv			BN	Activation
Kernel	Stride	Channel	Channel	
3	2	256	256	ReLU
3	1	256	256	ES
3	1/2	128	-	-
3	1	128	128	ReLU
3	1/2	64	-	-
3	1	64	64	ReLU
7	1	3	-	-

(d) Discriminator

Conv			BN	Activation
Kernel	Stride	Channel	Channel	
3	1	32	-	LReLU
3	2	64	-	LReLU
3	1	64	64	LReLU
3	2	128	-	LReLU
3	1	128	128	LReLU
3	1	256	256	LReLU
3	1	1	-	-

TABLE I

LAYER CONFIGURATIONS OF THE ARCHITECTURE, IN WHICH 'BN' INDICATES THE BATCH NORMALIZATION AND 'ES' REPRESENTS ELEMENTWISE SUM.

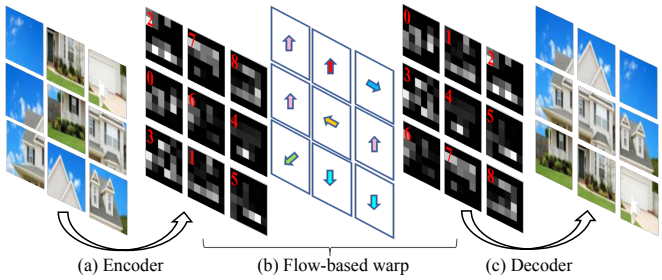


Fig. 3. The flow-based warp can warp the shuffled features to correct order according to the classification results.

images as input, the encoder (Fig. 2 (a)) extracts features of the shuffled pieces. The features are used for the classification branch (Fig. 2 (b)) and GAN branch (Fig. 2 (d) and (e) and the discriminator). The classification branch predicts the jigsaw permutations, which are constrained by the pseudo-labels, called reference labels. During the training, we find that an unsupervised classification network is difficult to converge. To solve the problem, we generate the reference labels to guide C. The classification results provide information to generate flow fields (Fig. 2 (c)) that can shuffle the features. The warped features with the new order are then sent to the residual blocks (Fig. 2 (d)) and the decoder (Fig. 2 (e)) to recover natural images. When the real permutation labels and the correct natural images are unavailable, we adopt GAN architecture to achieve unsupervised optimization. The vanilla GAN consists of two components, a generator G and a discriminator D, where G is responsible for capturing the data distribution while D tries to distinguish whether a sample comes from the real training data or the generator [43], [44]. This framework corresponds to a min-max two-player game, which provides a powerful way to estimate target distribution and generate new samples. The GAN branch provides global constraints to enable the network to concentrate on semantic clues. The decoder and the discriminator as well as extra losses enable the encoder to generate more informative features and further improve the classification network.

Classification Network: The classification network consists of two parts: the convolutional blocks (the encoder) and the classification module. Useful signals are extracted by the encoder for downstream transformation. The classification module aims to distinguish different permutations, which includes convolutional layers, max pooling layers and fully connected layers.

Generator: The generator network is composed of three parts, the encoder, eight residual blocks and the decoder. The encoder of classification network also extracts features for the generator. Afterward, eight residual blocks with identical layouts are adopted to construct the content and the manifold feature. The decoder consists of two identical transposed convolutional blocks and a final convolutional layer.

Discriminator: The discriminator network is complementary to the generator, which aims to classify each image into a real or fake one. The discriminator network just includes several convolutional blocks. Such a simple discriminator uses fewer parameters and can work on images of arbitrary sizes.

B. Flow-based Warp

The GAN branch cannot recover the extracted features directly because the features are also shuffled. We introduce a flow-based warp to reassemble the shuffled encoder features and ensure the decoder can output natural images. In detail, we construct optical flows according to the predicted labels, which are used to warp the shuffled encoder features into the correct order. The decoder can recover perfect reorganized images if the classification labels are accurate, as shown in Fig. 3. Some wrong classification can be tolerated when the network is initialized because the GAN optimization can correct the mistakes gradually.

C. Loss Function

We design our objective function to include the following three losses: (1) the jigsaw loss $L_{jigsaw}(C)$, which optimizes the classification network to recognize correct permutation; (2) the adversarial loss $L_{\mathrm{GAN}}(G, D)$, which drives the generator

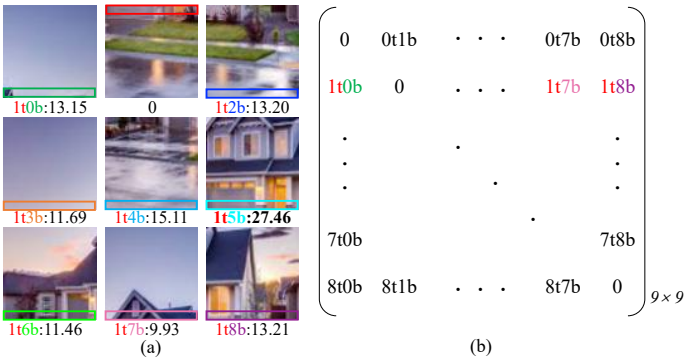


Fig. 4. In this case, we calculate PSNR between the top edges of top-middle piece and the bottom edges of other pieces. Corresponding PSNR values are displayed in (a). These PSNR values compose the second row of the 9×9 top-bottom relationship metric, as shown in (b).

TABLE II
EXPERIMENTS TO SHOW THE INFLUENCE OF HYPER-PARAMETER pix.

	pix = 3	pix = 2	pix = 1
PSNR	20.1	21.5	24.7
Accuracy	56.5%	60.3%	66.4%

network to achieve the desired manifold transformation; (3) the edge loss $L_{edge}(G, D)$, which pushes the decoder of G to recover clear images, and further constrains the encoder to generate useful features that are helpful for the classification network. The full objective function is:

$$L(G, D, C) = L_{jigsaw}(C) + L_{\text{GAN}}(G, D) + L_{edge}(G, D) \quad (1)$$

1) Jigsaw Loss: The inputs are obtained by decomposing the source images using a regular $n \times n$ grid of pieces, which are then shuffled and re-assigned to one of the n^2 grid positions. Out of the $n^2!$ possible permutations, P elements are selected by following the Hamming distance based algorithm in [39], and we assign an index to each entry. A simple classification network is first proposed to recognize correct permutation, which needs to minimize the jigsaw loss:

$$L_{jigsaw}(C) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[CE(C(x) - p)] \quad (2)$$

where CE means cross-entropy loss, $p \in \{1, ..., P\}$ are the jigsaw labels. However, the jigsaw labels are unavailable for unsupervised tasks. Directly training C without jigsaw labels is impossible, which tends to assign labels randomly, achieving 30-40% classification accuracy for three-class tasks and 20-30% for four-class tasks according to our experiments.

A self-supervised classification network is then proposed to achieve better classification performance. As demonstrated in Section I, the definition of 'self-supervised' in our pipeline is different from conventional 'self-supervised' methods. Specifically, we generate pseudo-labels, called reference labels, from the shuffled input images. The reference labels can constrain C when permutation indexes are unavailable. In detail, we apply the following five steps to obtain the reference labels:

1) Cutting four edges from each piece, and the width of the edge is determined by a hyper-parameter pix.

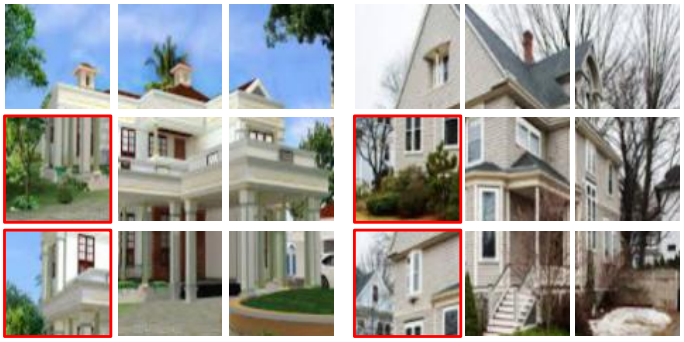


Fig. 5. Two incorrect reorganization results that may confuse the discriminator.

2) Calculating the PSNR values between the top edge of a specific piece and bottom edges of other pieces to obtain a vector with size $1 \times n^2$. Then, a $n^2 \times n^2$ matrix can be obtained when computes all top-bottom relationships, as shown in Fig. 4.
3) Applying the same way to get another $n^2 \times n^2$ matrix to indicate the left-right relationships.
4) There are $n \times (n-1)$ correct top-bottom edge pairs and $n \times (n-1)$ correct left-right edge pairs, so we adopt a greedy algorithm [45] to select $n \times (n-1)$ maximum values from two matrices, respectively.
5) A minimum spanning tree (MST) [23] is then applied to assemble the pieces and the reference permutation p_{ref} is returned.

Empirically, pix is set to 1, which means we cut a row of pixels of each edge. Table II shows the corresponding PSNR values and reorganization accuracy when we select different pix. A larger pix may cut a wider edge strip, however, redundancy could decrease PSNR and reorganization accuracy. We select pix = 1 when we obtain the edge strips, which are more sensitive to boundary discontinuity.

Cross-entropy loss minimizes the KL divergence between the predicted (softmax) distribution and the target distribution (one-hot encoding in classification tasks) over classes, while focal loss (FL) [46] minimizes a regularised KL divergence between these two distributions, which ensures minimization of the KL divergence whilst increasing the entropy of the predicted distribution, thereby preventing the model from becoming overconfident. We replace the cross-entropy loss conventionally used with the focal loss to improve network calibration. Finally, the jigsaw loss can be described as follows:

$$L_{jigsaw}(C) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[FL(C(x) - p_{ref})] \quad (3)$$

2) Adversarial Loss: The performance of classification module C is totally limited by the reference labels. After the encoder module generating distinctive features, we add a decoder module to recover the features to natural images. We do not train an autoencoder network that directly recovers the shuffled inputs because it is useless for the encoder to generate more informative features. Our proposed flow-based warp is able to warp the shuffled features to correct order according to the classification results. The reorganized features are then sent

Fig. 6. Examples generated by the proposed JigsawGAN. The top two rows show the results when hyper-parameter $n = 2$, and the bottom two rows are results when $n = 3$. Different styles can be efficiently reassembled by JigsawGAN.

to the GAN branch to recover the images. The GAN branch is trained to help the encoder module to generate useful features being aware of semantic information, and further improve the classification performance.

As in classic GAN networks, the adversarial loss is used to constrain the results of G to look like target domain images. In our task, adversarial loss pushes G to generate outputs that are natural images in the absence of corresponding ground truth. Meanwhile, D aims to distinguish whether a given image belongs to the synthesized or the real target manifold, which tries to correctly classify an image into two categories: the generated image $G(x)$ and the real image y, as formulated in Eq. 4.

$$\begin{aligned} L_{\mathrm{GAN}} = & \mathbb{E}_{y \sim p_{\mathrm{data}}(y)}\big[\log D(y)\big] \\ & + \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}\big[\log(1 - D(G(x))\big] \end{aligned} \tag{4}$$

3) Edge Loss: The adversarial loss ensures that the generated image looks similar to the target domain, it is inadequate for a smooth transaction, especially at the boundary of pieces. For example, in Fig. 5, the incorrect order of patches in house image may confuse the discriminator, such that the discriminator cannot work as we desired. Therefore, it is essential to enforce a more strict constraint to ensure semantic consistency of the reorganized images. To achieve this, we add edge penalty at patch boundaries to punish the incorrect combination. We calculate PSNR values as edge loss to further optimize the generator, among which images with large PSNR edge scores own small edge loss. Note that, PSNR values have a wide range, which make distinctions between different images and difficult to normalize. SSIM plays the same role as PSNR and ranges from 0 to 1, which is more suitable as edge constraint to optimize the GAN network. As such, the edge loss is defined as:

$$L_{Edge}(C) = \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[1 - SSIM(G(x))] \tag{5}$$

IV. Experiments

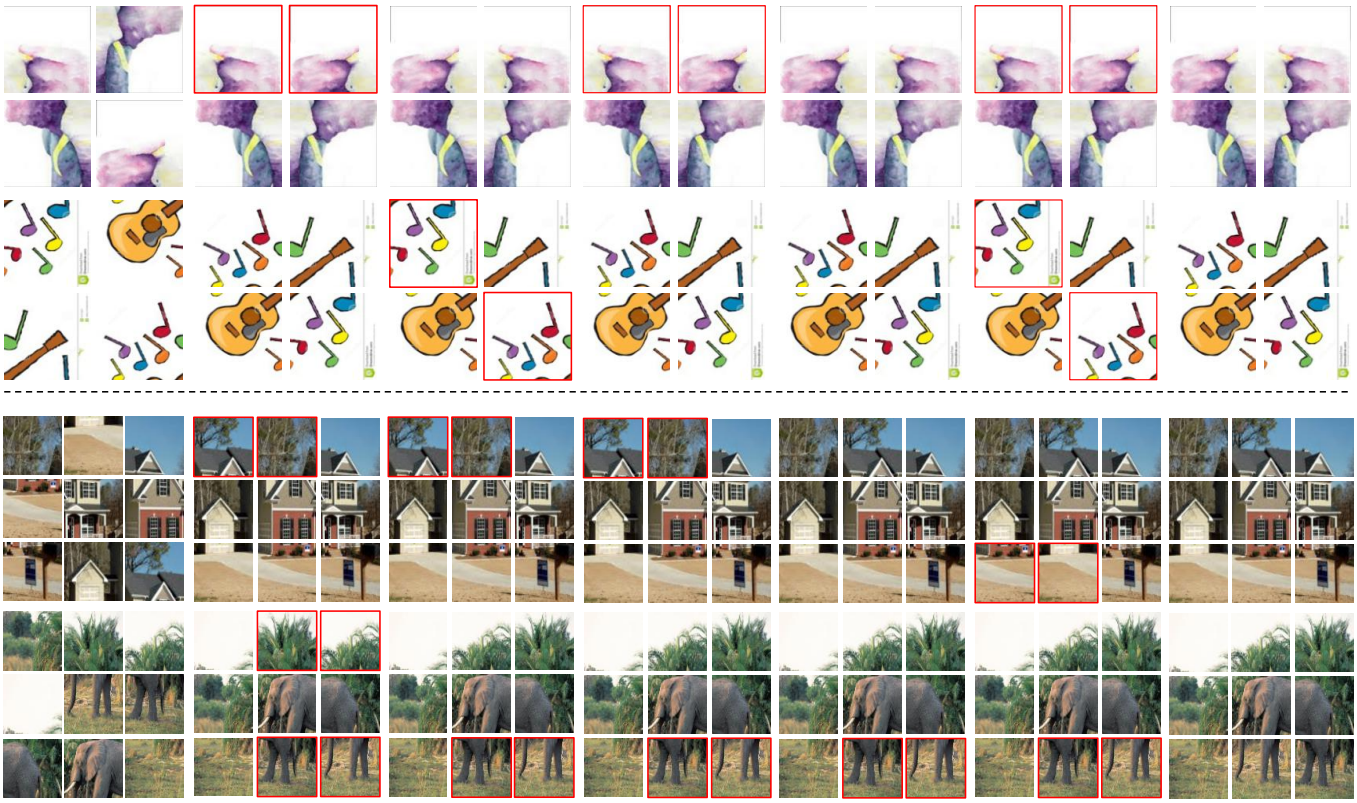
We first illustrate the datasets and implementation details in Section IV-A and then conduct comprehensive experiments to verify the effectiveness of the proposed method, including quantitative comparisons (Section IV-B), qualitative comparisons (Section IV-C) and ablation studies (Section IV-D). Specifically, we first compare our method with several representative jigsaw puzzle works, including a classic jigsaw puzzle solver [23], a linear programming-based method [35], a loop constraints-based method [25], a graph connection laplacian-based method [15], and then compare with a recent deep learning (DL)-based method which relied on shortest path optimization [14]. Some representative methods are designed for handling 'Type 1' and 'Type 2' puzzles, including a number of pieces. Their methods can be easily applied for our 'Type 1' task for 3×3 pieces. Next, ablation studies are performed to illustrate the importance of different components.

A. Datasets and Implementation Details

1) Data Collection: The dataset includes 7,639 images, among which 5,156 images originate from PACS dataset [47] and 2,483 images are our own collection. The PACS dataset includes many pictures with similar content, so we gathered images from movies or from the Internet that are more distinguishable. Our dataset covers 4 object categories (house,

TABLE III
QUANTITATIVE COMPARISONS OF THE PROPOSED METHOD WITH SEVERAL REPRESENTATIVE JIGSAW PUZZLE SOLVERS IN TERMS OF REORGANIZATION ACCURACY.

	Gallagher2012 [23]	Son2014 [25]	Yu2015 [35]	Huroyan2020 [15]	Paumard2020 [14]	Ours
House	64.5%	70.3%	73.8%	76.4%	61.2%	**80.2%**
Person	63.8%	68.2%	70.6%	73.7%	63.6%	**79.9%**
Guitar	59.4%	64.5%	68.3%	71.1%	60.4%	**77.4%**
Elephant	60.5%	66%	71.4%	73.1%	58.2%	**78.5%**
Mean	62.1%	67.3%	71.0%	74.6%	60.9%	**79.0%**



(a) Inputs (b) Results of [23] (c) Results of [25] (d) Results of [35] (e) Results of [15] (f) Results of [14] (g) Ours

Fig. 7. Comparisons of JigsawGAN with Gallagher *et al.*'s method [23], Son *et al.*'s method [25], Yu *et al.*'s method [35], Huroyan *et al.*'s method [15] and Paumard *et al.*'s method [14]. The top two rows show the results when hyper-parameter $n = 2$, and the bottom two rows are results when $n = 3$. The proposed method assembles both on edge and semantic information in order to generate more accurate results.

person, elephant and guitar), and each of them can be divided into 4 domains (Photo, Art Paintings, Cartoon and Sketches). Each category is divided into three subsets: a jigsaw set (40%) to generate the inputs of the pipeline, a real dataset (40%) to help the discriminator to distinguish the generated images and real images, and a test set (20%) to evaluate the performance of different methods. They are randomly selected from the overall dataset. The shuffled fragments are then prepared according to [41]: giving input images with dimension $3 \times H \times W$, we equally split it into $n \times n$ pieces which have $3 \times \frac{H}{n} \times \frac{W}{n}$ dimensions. The size of input images is 72×72 and permutations P is set to 1,000 in our implementation. The hyper-parameter n and the permutations P are important factors for the network performance. We perform ablation studies on these two factors in Section IV-D2.

2) Training Details: We implement JigsawGAN in PyTorch and all the experiments are performed on an NVIDIA RTX 2080Ti GPU with 100 epochs. For each iteration, every input image is cut into $n \times n$ pieces, and the pieces are parallelly sent to the network. The separated encoder features are combined before reassembling by the flow-based warp. We choose the discrete pieces as inputs in order to prevent the cross-influence between the piece boundaries. Adam optimizer is applied with the learning rate of 2.0×10^{-4} for the generator, discriminator and classification network. The entire training process costs 4 hours on average.

B. Quantitative Comparisons

Table III lists the reorganization accuracy of aforementioned methods and the proposed JigsawGAN. The scores reported are the average results over the test set. Some comparison

methods [15], [23], [25], [35] obey following basic rules to solve jigsaw puzzles: (1) detecting edges to determine the relationship between pieces, and (2) applying different optimization methods to reassemble the images. Specifically, Gallagher *et al.* applied Mahalanobis Gradient Compatibility (MGC) measurement to determine the pieces relationship and a greedy algorithm to reorganize the pieces [23]. A considerable improvement for [23] was demonstrated by [25], by adding loop constraints to the pieces reorganization process. Yu *et al.* combined the advantages of greedy methods and loopy propagation algorithms to introduce a linear programming-based solver [35]. Huroyan *et al.* applied graph connection laplacian to better understand the constructive mechanism [15]. However, these methods are limited by the edge detection step, resulting in some randomness in their results. Paumard *et al.* [14] applied the shortest path optimization algorithm to reorganize the pieces, whose edge relationship is predicted by a neural network. Their pruning strategy for the shortest path algorithm may affect the performance. Compared to these methods, our JigsawGAN achieves better performance, which improves the performance against [14], [23], [25] with a relatively large margin, and has more than 4 point improvements compared with [15], [35].

C. Qualitative Comparisons

In this section, we first show some reorganization results generated by JigsawGAN in Fig. 6, among which the proposed method is able to reassemble the inputs and generates high-quality results. Then, qualitative comparisons of JigsawGAN and aforementioned methods are presented in Fig. 7. The top two rows show the results when hyper-parameter $n = 2$, and the bottom two rows are results when $n = 3$. In the first elephant example, top two pieces are easily confused if do not consider the semantic content information. Results of [23], [35] and [14] fail to solve the case. The guitar example has weak boundary constraints between the pieces, which leads to the failure of [25] and [14]. The strategy of Son *et al.* aims to recover the complete shape from pieces given a dissimilarity metric [25]. The two musical note pieces have small contribution on constructing the guitar structure and confuse Son *et al.*'s method. Paumard *et al.* reassemble pieces with boundary erosion. which directly determine the piece relationships through a neural network and ignore the useful boundary information [14]. In the house example, results of [23], [25] and [35] fail to solve the tree and roof pieces, while result of [14] cannot distinguish two road pieces. In the second elephant example, they all fail to discriminate the leg pieces. We noted that puzzles with low percentages of recovery by these algorithms have large portions of pieces with the same uniform texture and color. In comparison, by considering the global information (GAN loss) and the edge information (edge loss) simultaneously, the proposed JigsawGAN recovers these cases well.

D. Ablation Studies

1) Ablation Study of Loss Terms: We perform the ablation studies on the variants of loss functions to understand how these main modules contribute to the final results. Table IV displays the ablation results of our loss functions. These results show that each component contributes to our objective function.

TABLE IV
ABLATION EXPERIMENTS TO SHOW THE IMPORTANCE OF DIFFERENT LOSS TERMS.

	p_{ref}	w/o. L_{GAN}	w/o. L_{Edge}	Ours
Accuracy	66.4%	68.1%	75.8%	79.0%

TABLE V
THE REORGANIZATION ACCURACY WHEN SELECT DIFFERENT n AND P.

	$P = 10$	$P = 100$	$P = 1000$
$n = 2$	91.9%	87.0%	83.2%
$n = 3$	85.6%	83.3%	79.0%

We have illustrated the importance of L_{jigsaw} in Section III-C1. Directly training the classification network without jigsaw loss is impossible. The network tends to assign results randomly, achieving 30-40% classification accuracy for three-class tasks and 20-30% for four-class tasks. Table IV further shows the importance of L_{GAN} and L_{Edge}. As shown in Table IV, removing L_{GAN} substantially degrades the results, so as removing the L_{Edge}. The third column shows the reorganization accuracy without L_{GAN}, which indicates the discriminator and corresponding loss are removed. The accuracy of without L_{GAN} is apparently inferior to the final results. The fourth column displays the results without L_{Edge}, which means the decoder is only constrained by the global GAN loss. The results are also not as good as our final results. We also conduct the experiment that without both L_{GAN} and L_{Edge}, which represents the GAN branch is not involved, where the final classification accuracy only depends on the reference labels p_{ref}. The result in second column is worse than the final results. We conclude that all three loss terms are critical.

2) Ablation Study of Grids and Permutations: The selection of different n and P have significant influence on the reorganization accuracy. As shown in Table V, increasing the permutation P obviously degrades the accuracy, so as increasing the hyper-parameter n. Increasing P makes the classification more complicated, while increasing n improves the difficulty of detecting piece relationship. Moreover, the choice of hyper-parameter n should satisfy the condition that the size of the pieces is smaller than the receptive field.

3) Ablation Study of Reference Labels: The proposed method can be considered as an improvement technique of existing methods, which is able to improve the reorganization accuracy of them if we select their results as our reference labels. The improvement is benefited from the GAN branch and corresponding losses. Table VI shows the improvements of aforementioned methods if we choose their results as our reference labels, respectively. As for our results, higher reference label accuracy leads to higher reorganization accuracy because the reference labels can better guide the network to learn useful information, such as achieving 81.0% accuracy for method [15]. The improvements of [23] and [25] are

TABLE VI
THE IMPROVEMENT WHEN SELECT DIFFERENT METHODS AS OUR REFERENCE LABELS.

	Gallagher2012 [23]	Son2014 [25]	Yu2015 [35]	Huroyan2020 [15]	Paumard2020 [14]
Original accuracy	62.1%	67.3%	71.0%	74.6%	60.9%
Ours	75.6%	78.4%	79.2%	81.0%	69.4%
Improvements	+13.5%	+11.1%	+7.8%	+6.4%	+8.5%

significant, while the improvements of [35] and [15] are relatively smaller. This is because there is an upper bound of the network. As for reference labels with lower accuracy, the GAN branch can promote them significantly. Moreover, it is reasonable that the improvement for [14] is small because [14] is mainly designed for their proposed dataset with eroded boundaries. Overall, no matter how the reference labels are obtained, it can be promoted by utilizing our proposed architecture.

V. CONCLUSION

We have proposed JigsawGAN, a GAN-based self-supervised method for solving jigsaw puzzles when the prior knowledge of the initial images is unavailable. The proposed method can apply the edge information of pieces and the semantic information of generated images, which is helpful for solving jigsaw puzzles more accurately. The architecture contains two branches, which are connected by an encoder and a flow-based warp. The two branches play different roles and promote each other. The GAN branch drives the encoder to generate more information features and further improve the classification branch. GAN loss and a novel edge loss are introduced to constrain the network to focus on semantic information and edge information, respectively. We have conducted comprehensive comparisons, include quantitative assessment, qualitative comparisons and ablation studies, with several typical methods to demonstrate the effectiveness of JigsawGAN.

REFERENCES

[1] C. Papaodysseus, T. Panagopoulos, M. Exarhos, C. Triantafillou, D. Fragoulis, and C. Doumas, "Contour-shape based reconstruction of fragmented, 1600 bc wall paintings," *IEEE Trans. on Signal Processing*, vol. 50, no. 6, pp. 1277–1288, 2002.

[2] D. Koller and M. Levoy, "Computer-aided reconstruction and new matches in the forma urbis romae," *Bullettino Della Commissione Archeologica Comunale di Roma*, vol. 2, pp. 103–125, 2006.

[3] B. J. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, S. Rusinkiewicz, and T. Weyrich, "A system for high-volume acquisition and matching of fresco fragments: Reassembling theran wall paintings," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–9, 2008.

[4] C. Toler-Franklin, B. Brown, T. Weyrich, T. Funkhouser, and S. Rusinkiewicz, "Multi-feature matching of fresco fragments," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 1–12, 2010.

[5] R. Pintus, K. Pal, Y. Yang, T. Weyrich, E. Gobbetti, and H. Rushmeier, "A survey of geometric analysis in cultural heritage," *Computer Graphics Forum*, vol. 35, no. 1, pp. 4–31, 2016.

[6] E. Justino, L. S. Oliveira, and C. Freitas, "Reconstructing shredded documents through feature matching," *Forensic Science International*, vol. 160, no. 2-3, pp. 140–147, 2006.

[7] M. A. Marques and C. O. Freitas, "Reconstructing strip-shredded documents using color as feature matching," in *ACM symposium on Applied Computing*, pp. 893–894, 2009.

[8] H. Liu, S. Cao, and S. Yan, "Automated assembly of shredded pieces from multiple photos," *IEEE Trans. on Multimedia*, vol. 13, no. 5, pp. 1154–1162, 2011.

[9] A. Deever and A. Gallagher, "Semi-automatic assembly of real cross-cut shredded documents," in *in Proc. ICIP*, pp. 233–236, 2012.

[10] W. Marande and G. Burger, "Mitochondrial dna as a genomic jigsaw puzzle," *Science*, vol. 318, no. 5849, pp. 415–415, 2007.

[11] T. Altman, "Solving the jigsaw puzzle problem in linear time," *Applied Artificial Intelligence an International Journal*, vol. 3, no. 4, pp. 453–462, 1989.

[12] E. D. Demaine and M. L. Demaine, "Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity," *Graphs and Combinatorics*, vol. 23, no. 1, pp. 195–208, 2007.

[13] V. Sivapriya, M. Senthilvel, and K. Varghese, "Automatic reassembly of fragments for restoration of heritage site structures," in *International Symposium on Automation and Robotics in Construction*, vol. 35, pp. 1–7, 2018.

[14] M.-M. Paumard, D. Picard, and H. Tabia, "Deepzzle: Solving visual jigsaw puzzles with deep learning and shortest path optimization," *IEEE Trans. on Image Processing*, vol. 29, pp. 3569–3581, 2020.

[15] V. Huroyan, G. Lerman, and H.-T. Wu, "Solving jigsaw puzzles by the graph connection laplacian," *SIAM Journal on Imaging Sciences*, vol. 13, no. 4, pp. 1717–1753, 2020.

[16] H. Freeman and L. Garder, "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition," *IEEE Trans. on Electronic Computers*, no. 2, pp. 118–127, 1964.

[17] M. G. Chung, M. M. Fleck, and D. A. Forsyth, "Jigsaw puzzle solver using shape and color," in *International Conference on Signal Processing*, vol. 2, pp. 877–880, 1998.

[18] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, "A fully automated greedy square jigsaw puzzle solver," in *in Proc. CVPR*, pp. 9–16, 2011.

[19] G. M. Radack and N. I. Badler, "Jigsaw puzzle matching using a boundary-centered polar encoding," *Computer Graphics and Image Processing*, vol. 19, no. 1, pp. 1–17, 1982.

[20] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan, "Solving jigsaw puzzles by computer," *Annals of Operations Research*, vol. 12, no. 1, pp. 51–64, 1988.

[21] R. W. Webster, P. S. LaFollette, and R. L. Stafford, "Isthmus critical points for solving jigsaw puzzles in computer vision," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 21, no. 5, pp. 1271–1278, 1991.

[22] W. Kong and B. B. Kimia, "On solving 2d and 3d puzzles using curve matching," in *in Proc. CVPR*, 2001.

[23] A. C. Gallagher, "Jigsaw puzzles with pieces of unknown orientation," in *in Proc. CVPR*, pp. 382–389, 2012.

[24] F. A. Andaló, G. Taubin, and S. Goldenstein, "Solving image puzzles with a simple quadratic programming formulation," in *SIBGRAPI Conference on Graphics, Patterns and Images*, pp. 63–70, 2012.

[25] K. Son, J. Hays, and D. B. Cooper, "Solving square jigsaw puzzles with loop constraints," in *in Proc. ECCV*, pp. 32–46, 2014.

[26] L. Chen, D. Cao, and Y. Liu, "A new intelligent jigsaw puzzle algorithm base on mixed similarity and symbol matrix," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 32, no. 02, p. 1859001, 2018.

[27] D. Mondal, Y. Wang, and S. Durocher, "Robust solvers for square jigsaw puzzles," in *International Conference on Computer and Robot Vision*, pp. 249–256, 2013.

[28] D. Sholomon, O. E. David, and N. S. Netanyahu, "Dnn-buddies: A deep neural network-based estimation metric for the jigsaw puzzle problem," in *International Conference on Artificial Neural Networks*, pp. 170–178, 2016.

[29] L. Dery, R. Mengistu, and O. Awe, "Neural combinatorial optimization for solving jigsaw puzzles: A step towards unsupervised pre-training," 2017.

[30] M.-M. Paumard, D. Picard, and H. Tabia, "Image reassembly combining deep learning and shortest path problem," in *in Proc. ECCV*, pp. 153–167, 2018.

[31] D. Bridger, D. Danon, and A. Tal, "Solving jigsaw puzzles with eroded boundaries," in *in Proc. CVPR*, pp. 3526–3535, 2020.

[32] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2020.

[33] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2021.

[34] T. S. Cho, S. Avidan, and W. T. Freeman, "A probabilistic image jigsaw puzzle solver," in *in Proc. CVPR*, pp. 183–190, 2010.

[35] R. Yu, C. Russell, and L. Agapito, "Solving jigsaw puzzles with linear programming," *arXiv preprint arXiv:1511.04472*, 2015.

[36] G. Paikin and A. Tal, "Solving multiple square jigsaw puzzles with missing pieces," in *in Proc. CVPR*, pp. 4832–4839, 2015.

[37] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *in Proc. NIPS*, vol. 28, pp. 2692–2700, 2015.

[38] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *in Proc. ICCV*, pp. 1422–1430, 2015.

[39] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *in Proc. ECCV*, pp. 69–84, 2016.

[40] R. Santa Cruz, B. Fernando, A. Cherian, and S. Gould, "Deeppermnet: Visual permutation learning," in *in Proc. CVPR*, pp. 3949–3957, 2017.

[41] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *in Proc. CVPR*, pp. 2229–2238, 2019.

[42] R. Du, D. Chang, A. K. Bhunia, J. Xie, Z. Ma, Y.-Z. Song, and J. Guo, "Fine-grained visual classification via progressive multi-granularity training of jigsaw patches," in *in Proc. ECCV*, pp. 153–168, 2020.

[43] I. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *in Proc. NIPS*, pp. 2672–2680, 2014.

[44] R. Li, C.-H. Wu, S. Liu, J. Wang, G. Wang, G. Liu, and B. Zeng, "Sdp-gan: Saliency detail preservation generative adversarial networks for high perceptual quality style transfer," *IEEE Trans. on Image Processing*, vol. 30, pp. 374–385, 2020.

[45] M.-M. Paumard, D. Picard, and H. Tabia, "Jigsaw puzzle solving using local feature co-occurrences in deep neural networks," in *in Proc. ICIP*, pp. 1018–1022, 2018.

[46] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *in Proc. ICCV*, pp. 2980–2988, 2017.

[47] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Deeper, broader and artier domain generalization," in *in Proc. ICCV*, pp. 5542–5550, 2017.