Jigsaw Clustering for Unsupervised Visual Representation Learning

Pengguang Chen[1] Shu Liu[2] Jiaya Jia[1,2]
The Chinese University of Hong Kong[1] SmartMore[2]
{pgchen, leojia}@cse.cuhk.edu.hk liushuhust@gmail.com

Abstract

Unsupervised representation learning with contrastive learning achieved great success. This line of methods duplicate each training batch to construct contrastive pairs, making each training batch and its augmented version forwarded simultaneously and leading to additional computation. We propose a new jigsaw clustering pretext task in this paper, which only needs to forward each training batch itself, and reduces the training cost. Our method makes use of information from both intra- and inter-images, and outperforms previous single-batch based ones by a large margin. It is even comparable to the contrastive learning methods when only half of training batches are used.

Our method indicates that multiple batches during training are not necessary, and opens the door for future research of single-batch unsupervised methods. Our models trained on ImageNet datasets achieve state-of-the-art results with linear classification, outperforming previous single-batch methods by 2.6%. Models transferred to COCO datasets outperforms MoCo v2 by 0.4% with only half of the training batches. Our pretrained models outperform supervised ImageNet pretrained models on CIFAR-10 and CIFAR-100 datasets by 0.9% and 4.1% respectively.

1. Introduction

Unsupervised visual representation learning, or self-supervised learning, is a long-standing problem, which aims at obtaining general feature extractors without human supervision. This goal is usually achieved by carefully designing pretext tasks without annotation to train feature extractors.

According to the definition of pretext tasks, most mainstream approaches fall into two classes: intra-image tasks and inter-images tasks. Intra-image approaches, including colorization [43, 20] and jigsaw puzzle [29], design a transform of one image and train the network to learn the transform. Since only the training batch itself is forwarded each time, we name them *single-batch methods*. This kind of tasks can be achieved using only one image's information, limiting the learning ability of feature extractors.

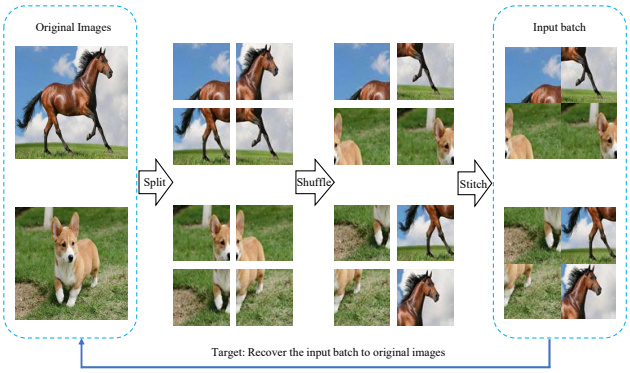

Figure 1. Sketch of our proposed pretext task. Images in the same batch are split into multiple patches, which are shuffled and stitched to form a new batch as input images for the network. The target is to recover the batch similar to the original images. We use two images here as an example.

Inter-images tasks are developed rapidly in recent years, which require the network to discriminate among different images. Contrastive learning is popular now since it reduces the distance between representation of positive pairs and enlarges the distance between representation of negative pairs. To construct positive pairs, another batch of images with different augmented views are used in the training process [5, 15, 26]. Since each training batch and its augmented version are forwarded simultaneously, we name these methods *dual-batches methods*. They greatly raise resource required for training an unsupervised feature extractor. The way to design an efficient single-batch based method with similar performance to dual-batches methods is still an open problem.

In this paper, we propose a framework for efficient training of unsupervised models using Jigsaw Clustering (JigClu). Our method combines advantages of solving jigsaw puzzles and contrastive learning, and makes use of both intra- and inter-image information to guide feature extractor. It learns more comprehensive representations. Our method only needs a single batch during training and yet greatly improves results compared to other single-batch

methods. It even achieves comparable results with dual-batches methods with only *half* of the training batches.

Jigsaw Clustering Task In our proposed Jigsaw Clustering task, every image in a batch is split into different patches. They are randomly permuted and stitched to form a new batch for training. The goal is to recover these disrupted parts back to the original images, as shown in Figure 1. Different from [29], the patches are permuted in a batch instead of a single image. The image each patch belongs to and the location of each patch in the origin are predicted in our work.

Also, we use montage images instead of single patches as input of the network. This modification greatly improves the difficulty for the task of [29] and provides more useful information for the network to learn. The network now has to distinguish between different parts of one image and identifies their original positions to recover the original image from multiple montage input images.

This task allows the network to learn both intra- and inter-images information by only forwarding the stitched images, using half of the training batches compared to other contrastive learning methods.

To recover patches across images, we design a clustering branch and a location branch as shown in Figure 2. Specifically, we first decouple the global feature map of stitched images into the representation of each patch. Then these two branches operate on representation of each patch. The clustering branch is to separate these patches into clusters, each of which only contains patches from the same image. The location branch, on the other hand, predicts location of every patch in an image agnostic manner.

With prediction from these two branches, the Jigsaw Clustering problem is solved. The clustering branch is trained as a supervised clustering task since we know the patches are from the same image, or not. The location branch is considered as a classification problem, where each patch is assigned with a label to indicate its location in the origin image. This branch predicts the label of every patch.

The reason that our method achieves decent results is that models trained with our proposed task can learn different kinds of information. At first, discriminating among different patches in one stitched image forces the model to capture instance-level information inside an image. This level of feature is missing in general in other contrastive learning methods.

Further, clustering different patches from multiple input images helps the model learn image-level features across images. This is the key that recent methods [15, 6, 5] achieve high-quality results. Our method retain this important property. Finally, arranging every patch to the correct location requires detailed location information, which was considered in single-batch methods [29, 43] before. It is, however, ignored in recent methods of [5, 26, 15, 6, 22]. We note this piece of information is still important to further improve results.

Performance of Our Method Learning by our method yields both intra- and inter-images information. This comprehensive learning brings a spectrum of superiority. First, with only one batch during training, our method outperforms other single-batch ones by 2.6% on linear evaluation on the ImageNe-1k dataset. Second, our method is more data-efficient. When the training data size is not large, our method can still produce decent results, much better than many other existing ones. On the ImageNet-100 and ImageNet-10% datasets, our system outperforms MoCo v2 by 6.2% and 6.0% respectively. Our method also converges more quickly with less training time. We use only a quarter of epochs of MoCo v2 to achieve the same results on the ImageNet-100 dataset.

Finally, the comprehensive information learned by our models is suitable for many other vision tasks. On the COCO detection dataset, our result is 0.4% better than MoCo v2, with only half of training batches. On the CIFAR-10 and CIFAR-100 datasets, models tuned with our pretrained weights achieve 0.9% and 4.1% higher results than that with supervised training weights, respectively. The extensive experiments demonstrate the superiority of our proposed pretext method.

2. Related Work

Handcrafted pretext tasks Many pretext tasks were proposed to train unsupervised models. Recovering the input image under corruption is an important topic, with tasks of descriminating synthetic artifacts [18], colorization [20, 43], image inpainting [31], and denoising auto-encoders [37], etc. Besides, many methods generate persuade-labels by transformation to train the network without human annotations. Applications involve predicting relation of two patches [9, 38], solving jigsaw puzzles [29, 19], and discriminating among surrogate classes [12]. [28] is an improved vision of jigsaw puzzles [29], which utilizes more complex methods to choose patches. Video information is also widely used for training unsupervised models [1, 25, 27, 30].

Contrastive learning Our method is also related to contrastive learning, which is first proposed in [14]. Following work [11, 39, 45, 36] further improved performance. Recently, constructing contrastive pairs using different augmentation of images [15, 5, 26, 44] achieves great success. Especially, [44] also utilize both intre- and inter-image information from pixel level. We note much training resource is required for training contrastive learning methods with

multiple batches of images. Our work tackles this problem with newly designed contrastive pairs in a single batch.

3. Jigsaw Clustering

In this section, definition of the task is presented. We then propose a very simple network, which only needs hardly modification of the original backbone network, to accomplish this task. Finally, a novel loss function is designed to better suit our clustering task.

3.1. The Jigsaw Clustering Task

There are n randomly selected images in one batch $\mathbf{X} = x_1, x_2, ..., x_n$. Every image x_i is split into $m \times m$ patches. There are $n \times m \times m$ patches in a batch totally. These patches are randomly permuted to form a new batch of montage images $\mathbf{X}' = x'_1, x'_2, ..., x'_n$. Every new image consists of $m \times m$ patches, which may come from different images in $\mathbf{X}$.

The task is to cluster the $n \times m \times m$ patches given the new batch $\mathbf{X}'$ into n clusters, and predict the location to recover the n original images with every $m \times m$ patches in the same cluster. The process is shown in Figure 1.

The key to the proposed task is to use montage images as input instead of every single patch. It is noteworthy that directly using small patches as input leads to the solution with only global information. Besides, small-size input images are not common for many applications. Only use them here raises the image-resolution difference problem between pretext and other downstream tasks. This may also lead to degradation of performance. Trivially scaling up small patches would violently increase the resource for training.

Our montage images as input nicely avoid these drawbacks. First, the input images form only one batch with the same size as the original batch, which costs half of resource during training compared with recent methods [5, 15]. More importantly, to better complete this task, the network has to learn detailed intra-image features to discriminate among different patches in one image, as well as global inter-image features to pull together different patches from the same original image. We observe that learning of comprehensive features greatly accelerates training of feature extractors. More experimental results are presented in Section 4.

The way to divide the image is a crucial part of our method. The choice of m affects the difficulty of the task. Our ablation study on a subset of ImageNet (see Section 5) shows that $m = 2$ achieves the best result. We conjecture that a larger m would exponentially increase the complexity and make the network fail to learn effectively. Besides, we observe that cutting the image into disjoint patches is not optimal. With an extend of intersection as shown in Figure 3, the network learns better features. It is explainable that different regions of some images are too diverse. They cause difficulty for learning without any evidence of overlap. More analysis is presented in Section 5.

3.2. Network Design

We design a new decouple network for this task as illustrated in Figure 2. One module is a feature extractor, which can be any common architectures [16, 34, 42, 41, 35]. There is also a parameter-free decouple module to separate the feature into $m \times m$ parts corresponding to different patches in one input image. Then a multi-layer perceptron (MLP) is used to embed every patch's feature for the clustering task; a fully-connected layer (FC) is used for the localization task.

The decouple module first interpolates the feature map of the backbone into a new one whose side length is a multiple of m. We enlarge the feature map instead of narrowing it to avoid information loss. For example, a typical input size of ImageNet dataset is 224×224. The feature map produced by a ResNet-50 backbone is 7×7. For $m = 2$, we interpolate the feature map into 8×8 by bilinear interpolation. When the length of the feature map is a multiple of m, we use average pooling to downsample the feature map to $n \times m \times m \times \hat{c}$. Then the features of a batch are disentangled to $(n \times m \times m) \times \hat{c}$, which means there are $(n \times m \times m)$ vectors of dimension $\hat{c}$.

Every vector is then embedded into length c with a two-layer MLP to form a set of vectors $\mathbf{Z} = \boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_{nmm}$ for the clustering task. In the meantime, a FC layer is attached after the $(n \times m \times m) \times \hat{c}$ vertors as the classifier to produce logits $\mathbf{L} = \boldsymbol{l}_1, \boldsymbol{l}_2, ..., \boldsymbol{l}_{nmm}$ for the localization task.

Our network is notably efficient, the additional decouple module is parameter-free. Compared to recent work, the computation of taking one batch remains almost the same, and we only need one batch during training. This greatly reduces training cost.

3.3. Loss Functions

The clustering branch is a supervised clustering task, because $m \times m$-size patches are in the same class. The supervised clustering task is convenient, and we use constractive learning to achieve it. We consider the target of clustering as pulling together objects from the same class and pushing away patches from different classes. Cosine similarity is used to measure the distance between patches. So for evey pair of patches in the same cluster, the loss function is

$$\ell_{i,j} = -log \frac{\exp(cos(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\Sigma_{k=1}^{nmm} \mathbb{1}_{k \neq i} \exp(cos(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \quad (1)$$

where $\mathbb{1}$ denotes the indicator function and τ is a temparture parameter to smooth or shappen the distance. The final loss function is summarized over all pairs from the same cluster

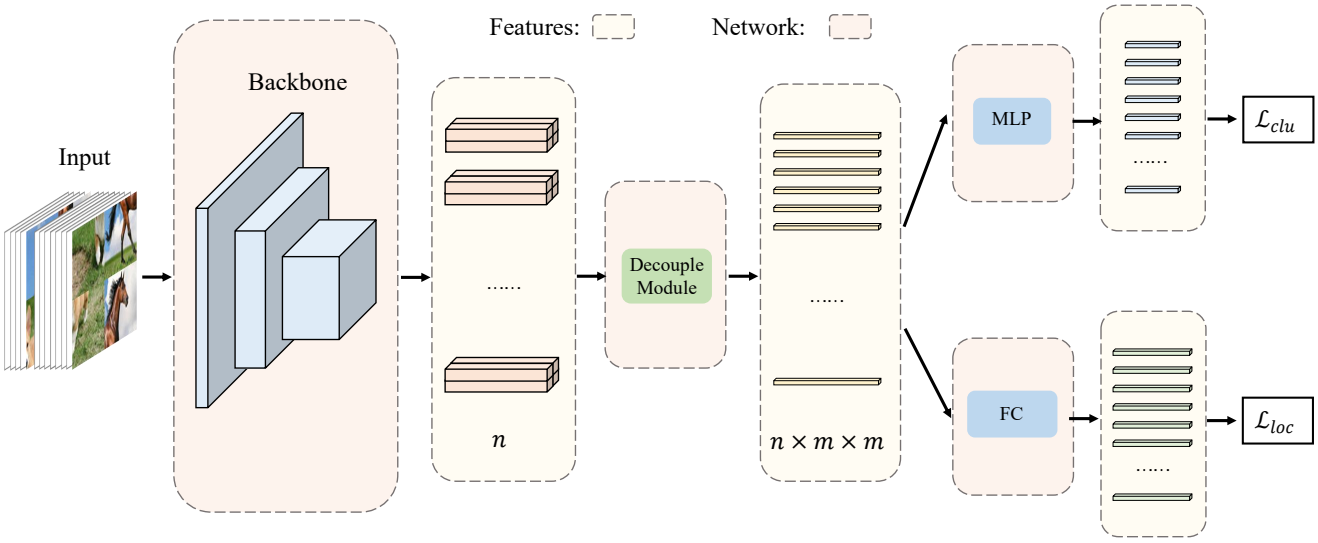

Figure 2. Pipeline of our method. We use light yellow rectangles to represent features produced by different parts of the network and light pink rectangles to represent parts of the network. The input images first go through the backbone network to produce n feature maps. Then the n feature maps are decoupled into $n \times m \times m$ vectors, each corresponding to one patch through a parameter-free decouple module. Afterwards, a MLP and a FC are used to embed vectors into logits to compute clustering loss and localization loss separately.

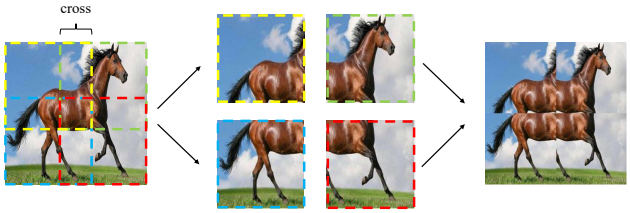

Figure 3. Patches split in images have a level of overlap.

as

$$\mathcal{L}_{clu} = \frac{1}{nmm} \Sigma_i \left(\frac{1}{mm-1} \Sigma_{j \in \mathbf{C}_i} \ell_{i,j} \right), \tag{2}$$

where $\mathbf{C}_i$ denotes the set of patch indices in the same cluster of i.

The location branch is considered as a classification task. The loss function is simply cross-entropy loss, and the loss of localization is formulated as

$$\mathcal{L}_{loc} = CrossEntropy(\mathbf{L}, \mathbf{L}_{gt}), \tag{3}$$

where $\mathbf{L}_{gt}$ denotes the ground truth for every-patch location.

The final objective of our proposed Jigsaw Clustering task is to optimize

$$\mathcal{L} = \alpha \mathcal{L}_{clu} + \beta \mathcal{L}_{loc}, \tag{4}$$

where α and β are hyperparameters to balance these two tasks. In our experiments, $\alpha = \beta = 1$ produces reasonable results.

4. Experiments

We report the performance of our unsupervised training method on ImageNet-1k [8] and ImageNet-100 datasets.

ImageNet-1k is a widely used classification dataset. There are 1.2+ million images uniformly distributed in 1,000 classes. We use the training set without labels to train our models.

ImageNet-100 is a subset of ImageNet-1k dataset, which is introduced in [36]. This dataset randomly chooses 100 classes of ImageNet-1k, containing around 0.13 million images. It is also well balanced in terms of class distribution. We use this small dataset to verify data efficiency of our method and perform fast ablation studies.

Unsupervised Training We use SGD to optimize our network with momentum 0.9. The weight decay is set to be $1e-4$. We train all models using batch size 256 on four GPUs. The learning rate is initialized as 0.03 and is decayed with cosine policy. All models are trained for 200 epochs if there is no further explanation.

4.1. Linear Evaluation

We first evaluate the feature learned by our method with a linear classification protocol. We train a ResNet-50 backbone on the ImageNet dataset with unsupervised learning. Then a supervised linear classifier is trained on the top of the

Method	# of Batch in Training	Accuracy
Supervised	single-batch	77.2
Colorization [43]	single-batch	39.6
JigPuz [29]	single-batch	45.7
DeepCulster [4]	single-batch	48.4
NPID [39]	single-batch	54.0
BigBiGan [10]	single-batch	56.6
LA [45]	single-batch	58.8
SeLa [2]	single-batch	61.5
CPC v2 [17]	single-batch	63.8
JigClu (Ours)	single-batch	**66.4**
MoCo [15]	dual-batches	60.6
PIRL [26]	dual-batches	63.6
SimCLR [5]	dual-batches	64.3
PCL [22]	dual-batches	65.9
MoCo v2 [6]	dual-batches	67.7

Table 1. Linear evaluation results of ResNet-50 models on the ImageNet-1k dataset. Our model outperforms previous single-batch methods by a large margin, achieving comparable results with dual-batches methods.

fixed backbone. The linear evaluation results on ImageNet-1k dataset are summarized in Table 1. Our method outperforms previous single-batch based methods by a large margin, greatly reduces the gap from dual-batch based methods.

Comparison with JigPuz JigPuz [29] also solves the jigsaw puzzle for unsupervised learning. It defines the problem as sorting the patches inside every single image. Our method, contrarily, solves the jigsaw problem from a general perspective, and enrichs the feature learned from it. The Jigsaw Clustering task outperforms JigPuz by 19.9% on the linear evaluation pipeline.

Comparison with Clustering Methods DeepCluster [4] and SeLa [2] are also based on clustering. But they use unsupervised clustering to guide the learning of models. We, instead, split images into different patches to generate ground truth for the clustering tasks. The supervised clustering task is more powerful for learning for our task and leads to much better representation.

Comparison with Contrastive-based Methods SimCLR [5] and MoCo [15, 6] are recently proposed based on contrastive learning. They achieve high-quality results at the cost of more training resource, since an additional batch is required during training. These methods need to scan twice of the batches compared with single-batch based methods. We also utilize contrastive loss for our clustering task, but do not need extra batches.

Dataset	ImageNet-100	ImageNet-10%
SimCLR	70.5	35.8
MoCo v2	74.7	38.3
JigClu (Ours)	**80.9**	**44.3**

Table 2. Linear evaluation results of ResNet-50 models on ImageNet-100 and ImageNet-10% datasets. Our results are significantly better than those of other methods on small datasets.

Our method achieves comparable results with state-of-the-art dual-batch based methods with only half of the training batches. MoCo v2 models yield slightly better results than ours on linear evaluation. Since MoCo v2 learns more of the inter-image information, it is suitable for the classification tasks. In contrast, our models learn comprehensive information, and therefore outperforms MoCo v2 on the detection tasks as shown in Section 4.3.

Data Efficiency We also experiment with our method on ImageNet-100 and a subset of ImageNet, which contains 10% data of every class in ImageNet. We train the ResNet-50 model on these datasets with unsupervised methods first. We report the linear evaluation results on the dataset to represent model ability. The results are presented in Table 2, notably better than those of other contrastive learning methods on relatively small datasets. This is because our method makes use of both intra- and inter-image information. The comprehensive learning strategy utilizes limited data more effectively.

Convergence We train unsupervised models on the ImageNet-100 dataset with different epochs and show the linear evaluation results in Figure 4. Our method achieves decent results with a very small number of training epochs, while other contrastive learning methods require longer training time to reach the same accuracy.

We explain that effective contrastive pairs are more frequent in our pretext tasks because of the split of input images. For exmaple, pairs in SimCLR are similar and are easy to recognize, leading to futile pairs. But the patches in the same cluster of our method come from different regions of the image, helpfully improve the quality of positive pairs.

4.2. Semi-supervised Learning

We also finetune the unsupervised model under the semi-supervised setting on the ImageNet-1k dataset with 10% and 1% labels. The labels are still class-balanced, provided in [5]. We finetune our model with a randomly initialized linear classifier on the labeled data. The results are summarized in Table 3.

Results of MoCo v2 are produced by us with the official model offered in [33]. We train it for 200 epochs for

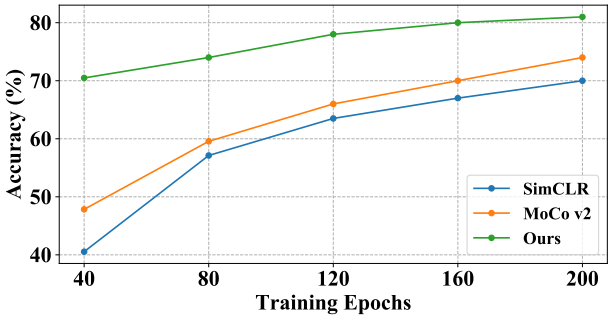

Figure 4. Results of ResNet-50 model on the ImageNet-100 dataset with linear evaluation protocol. The accuracy increases along with more of the total training epochs. Our method converges quickly. Note MoCo v2 costs around 160 epochs to reach the same level of output from our method in the 40th epoch.

Method	Label fraction			
	1%		10%	
	Top-1	Top-5	Top-1	Top-5
Supervised	25.4	48.4	56.4	80.4
Methods using label-propagation:				
Pseudo-label [21]	-	51.6	-	82.4
Entropy-Min [13]	-	47.0	-	83.4
S^4L-Rotation [3]	-	53.4	-	83.8
UDA* [40]	-	68.8	-	88.5
Methods using unsupervised learning:				
NPID [39]	-	39.2	-	77.4
PIRL [26]	-	57.2	-	83.8
MoCo v2 [6]	34.5	62.2	61.1	83.9
JigClu (Ours)	**40.7**	**68.9**	**63.0**	**85.2**

Table 3. Results of our pretrained model on the semi-supervised ImageNet classification tasks. Our method outperforms previous unsupervised learning ones. * indicates using RandAugment [7].

fair comparison. Compared with state-of-the-art representation learning methods, we achieve better results with only half of the training batches. The results of semi-supervised learning further manifest the superiority of our method. Result of UDA is with higher accuracy because it is specially designed for semi-supervised learning and utilizes powerful RandAugment [7].

4.3. Transfer Learning

We apply our pretrained ResNet-50 models to other vision tasks to prove generalization of our ResNet-50 models trained on ImageNet.

Objection Detection Following [33], we finetune our pretrained weights on the COCO detection dataset [24] with the Faster-RCNN R-50-FPN framework [32, 23]. The results are summarized in Table 4. Our results are better than those of MoCo v2 pretrained weights. Our models learn comprehensive information including instance-level discrimination and location recognition, useful for the detection tasks.

Models	AP	AP50	AP75	APs	APm	APl
MoCo v2	38.9	58.8	42.5	23.3	41.8	**50.0**
JigClu (Ours)	**39.3**	**59.4**	**42.5**	**23.6**	**42.5**	49.7

Table 4. Results of Faster-RCNN R50-FPN models trained on COCO detection dataset with pretrained weights provided by unsupervised training on ImageNet.

	Models	CIFAR-10	CIFAR100
finetune	Rand init.	88.4	61.6
	Supervised	88.6	60.6
	JigClu (Ours)	**89.5**	**64.7**
linear	Supervised	62.5	41.0
	JigClu (Ours)	**68.8**	**45.0**

Table 5. Results of ResNet-50 models trained on CIFAR-10 and CIFAR-100 datasets with different initialization.

Image Classification We also apply our pretrained weights to CIFAR-10 and CIFAR-100 datasets. The classifiers of models are randomly initialized and backbones are initialized in different ways including random values, supervised training models on ImageNet, and unsupervised pretrained models on ImageNet in our JigClu task.

The results are listed in Table 5. We use finetuning and linear evaluation to test representation learned by JigClu. In the finetuning setting, the backbone and classifier are trained on the target dataset together. Our weights provide the best initialization for both CIFAR-10 and CIFAR-100 datasets. In the linear evaluation process, only the linear classifier is trained on new datasets. Our model is better than the supervised pretrained model on ImageNet, which demonstrates the generality of our learned representation.

5. Analysis

5.1. Montage Input

We split every image in the batch into $m \times m$ patches and randomly permute them to form a new batch. The new batch used in our method consists of montage images as shown in Figure 5(c).

Using montage images as input is better than directly using patches. On the one hand, if we do not scale up the patches (Figure 5(a)), the network is trained with small images, which greatly reduces the representation ability when dealing with high-resolution images. On the other hand, after we resize the patches into a larger shape (Figure 5(b)),

(a) (b) (c)

Figure 5. (a) Small size patches. (b) Scaled-up patches. (c) Montage images.

Input Format	Accuarcy (%)	Time	Memory
(a) Small-size Patch	67.0	5.5h	2700MB
(b) Scaled-up Patch	71.3	16.8h	7300MB
(c) Montage Image	70.9	5.7h	3000MB

Table 6. Linear evaluation results of ResNet-18 models on the ImageNet-100 dataset. The scaled-up images lead to the best result, and yet cost much resource. Our proposed montage images achieve comparable results with scaled-up images, and only cost around 1/3 of the original training resource.

the size of input batches is $m \times m$ times larger than the orginal one, which greatly increases demand of training resource. Putting aside the size of patches, straightly using patches as input may cause missing a lot of instance-level intra-image information.

To prove the superiority of our proposed montage images, we conduct experiments on the ImageNet-100 dataset with ResNet-18 models. We choose ResNet-18 because it is much faster. Using scaled-up patches as input for ResNet-50 also causes the out-of-memory issue on our limited computing resource. We first train the models on the ImageNet-100 dataset, and evaluate them with a linear evaluation protocol.

The results are reported in Table 6. Concluded from the table, using small patches as input is very fast, and yet leads to low performance. Scaled-up patches much improve result quality. But they cost too much resource during training. Using our montage images as input overcomes these limitations, attaining high performance on limited computing resource. Although the way to use montage images in our method is still primary, amazing results are yielded. This opens the door for future research of single-batch unsupervised methods with montage images.

5.2. Data Augmentation

Data augmentation is very important in recent contrastive learning methods. We simply use the policy of MoCo v2 as our baseline augmentation. There is a split operation, which divides the image into $m \times m$ patches. We apply the baseline augmentation to every patch independently right after the split operation and before we perform the montage operation, as shown in Figure 6. There are many other choices; but empirically we find this simple strategy suffices. We analyze the position of augmentation in this section.

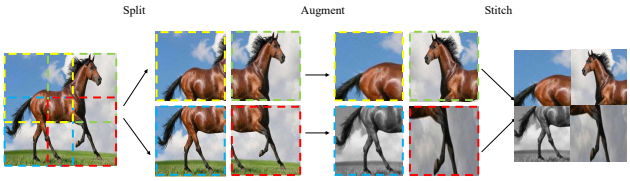

Figure 6. The position of augmentation used in our method. We augment every patch independently right after the split operation. In real cases, patches are mixed across images.

Augmentation position	Accuracy (%)
Aug before split	3.7
Split during aug	39.3
Aug after split	**80.9**
Aug after montage	Na

Table 7. The linear evaluation results of ResNet-50 models on the ImageNet-100 dataset with different augmentation policies. Applying augmentation on patches individually produces the best results.

Using the augmentation after the montage operation is not feasible, because random-crop may cut off some patches from the montage image. The augmentation could be used on original images before the split operation. However, this would cause many problems. First, the clustering branch may learn the augmentation bias instead of image features, because patches from the same images use the same augmentation. Second, the location of patches is hard to learn in an image-agnostic manner, because the augmented images may be at any positions of the original images.

An improvement option is to use the split operation between transform. For example, we could first crop the original images, and then split the cropped image into patches. These patches are further transformed by other operations such as color jitterring. This strategy only partially solves previous problems. Our augmentation right on patches tackles these issues.

We experiment with different augmentation ways, and report the results in Table 7. The ResNet-50 models are unsupervisedly pretrained on ImageNet-100 dataset with different augmentation policies, and the linear evaluation results are used to measure learning of representation. The model learns nothing when images are augmented before the split operation. Bringing forward the crop operation helps the model learn some useful information to produce non-trivial results. Applying augmentation to every patch is clearly a decent choice. It obtains high-quality results.

m	2	3	4
Accuracy (%)	80.9	74.7	70.1

Table 8. The linear evaluation results of ResNet-50 models on the ImageNet-100 dataset with different m. When m is larger than 2, the performance decreases because of the increased difficulty.

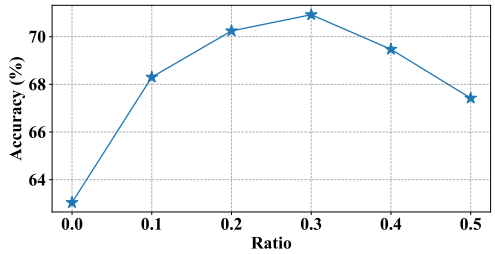

Figure 7. Accuracy in terms of different overlap ratios of adjacent patches. The x-axis is the length of overlapping regions, which is measured in terms of the percentage of image size. When $x = 0$, the patches are completely separated without overlap.

5.3. Split Operation

Number of Patches We split every image into $m \times m$ patches, where the choice of m is highly related to our task. The minimum value for m is 2. So we start from 2 to find an optimal m. The results are listed in Table 8.

We report the linear evaluation results of unsupervised learning ResNet-50 models on the ImageNet-100 datasets. It is obvious to conclude that the result of $m = 3$ is worse than $m = 2$. And we do not try larger ms. This result shows that setting $m = 3$ already makes the model difficult to learn because of the small input size (224×224). In this case, it becomes a problem to distinguish among these many patches inside one montage image. Without further notes, we use $m = 2$ in all our experiments.

Overlapped Region Size The size of overlap between adjacent patches also influences our method. We conduct experiments to find an optimal size for the overlapped regions. We train ResNet-18 on the ImageNet-100 dataset with our proposed unsupervised pretext task and use the linear evaluation to measure learning results. We use $m = 2$ in our experiments.

The results are summarized in Figure 7. Concluded from the figure, when there is no overlap between patches, the model does not learn effective features, because it is hard to distinguish among patches from the same image without any ideas how they are overlapped. When the overlap between patches becomes large, the result quality also drops. In this condition, patches from the same image may be very similar to each other and are easy to recognize, leading to reduction of effective positive pairs.

We find that using 0.3 of the original image's side length produces the best results. This ratio achieves a good balance of difficulty and efficiency for positive pairs. All our experiments are trained using this ratio.

Clustering branch	Location branch	Accuracy (%)
✔		65.1
	✔	3.2
✔	✔	66.4

Table 9. Ablation study of the two branches. The results is measures by the linear evaluation protocol of ResNet-50 models on the ImageNet-1k dataset.

5.4. Importance of the Two Branches

The proposed pretext task is solved by the two branches: clustering branch and location branch. Each branch has a loss function. The clustering branch is supervised by a contrastive-like loss, aiming to cluster patches from the same original images. This branch dominates the training of models. Both instance- and image-level information is learned from this branch. The location branch is supervised by a classification loss, which predicts the position of every patch in an image-agnostic manner. This branch assists the clustering branch with more detailed location information.

We train branches separately and summarize the results in Table 9. The results are measured on linear evaluation of unsupervised training with ResNet-50 models on the ImageNet-1k dataset.We can observe from the table that only training with the location branch leads to trivial accuracy. It reflects that the location information cannot be effectively learned from such complex montage input individually. However, the location branch is a nice auxiliary for the clustering branch. Joint training of both branches achieves the best results.

6. Conclusion

In this paper, we have proposed a novel Jigsaw Clustering pretext task/method, taking the advantage of both contrastive learning and previous handcrafted pretext tasks. Models trained with our method can learn both intra- and inter-images information with a single batch during training. Our method outperforms previous single-batch ones by large margins, and achieves comparable results with dual-batch methods with only half of the training batches. Our method naturally applies to other tasks.

Our work manifests, intriguingly, that single-batch methods have the potential to be in par with or even outperform dual-batch ones. We believe this line is worth further study. New applications can be expected.

References

[1] Pulkit Agrawal, João Carreira, and Jitendra Malik. Learning to see by moving. In *Int. Conf. Comput. Vis.*, 2015. 2

[2] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *Int. Conf. Learn. Represent.*, 2020. 5

[3] Lucas Beyer, Xiaohua Zhai, Avital Oliver, and Alexander Kolesnikov. S4L: self-supervised semi-supervised learning. In *Int. Conf. Comput. Vis.*, 2019. 6

[4] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Eur. Conf. Comput. Vis.*, 2018. 5

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020. 1, 2, 3, 5

[6] Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297, 2020. 2, 5, 6

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. *arXiv preprint arXiv:1909.13719*, 2019. 6

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2009. 4

[9] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *Int. Conf. Comput. Vis.*, 2015. 2

[10] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Adv. Neural Inform. Process. Syst.*, 2019. 5

[11] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin A. Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2016. 2

[12] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin A. Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Adv. Neural Inform. Process. Syst.*, 2014. 2

[13] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Adv. Neural Inform. Process. Syst.*, 2004. 6

[14] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2006. 2

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 1, 2, 3, 5

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 3

[17] Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aäron van den Oord. Data-efficient image recognition with contrastive predictive coding. *CoRR*, abs/1905.09272, 2019. 5

[18] Simon Jenni and Paolo Favaro. Self-supervised feature learning by learning to spot artifacts. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2

[19] Dahun Kim, Donghyeon Cho, Donggeun Yoo, and In So Kweon. Learning image representations by completing damaged jigsaw puzzles. 2018. 2

[20] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Eur. Conf. Comput. Vis.*, 2016. 1, 2

[21] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. 6

[22] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven C. H. Hoi. Prototypical contrastive learning of unsupervised representations. *CoRR*, abs/2005.04966, 2020. 2, 5

[23] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 6

[24] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Eur. Conf. Comput. Vis.*, 2014. 6

[25] Aravindh Mahendran, James Thewlis, and Andrea Vedaldi. Cross pixel optical-flow similarity for self-supervised learning. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *ACCV*, 2018. 2

[26] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 1, 2, 5, 6

[27] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Eur. Conf. Comput. Vis.*, 2016. 2

[28] T. Nathan Mundhenk, Daniel Ho, and Barry Y. Chen. Improvements to context based self-supervised learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2

[29] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Eur. Conf. Comput. Vis.*, 2016. 1, 2, 5

[30] Deepak Pathak, Ross B. Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 2

[31] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 2

[32] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Adv. Neural Inform. Process. Syst.*, 2015. 6

[33] Fackbook Research. moco. https://github.com/facebookresearch/moco, 2020. 5, 6

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. Learn. Represent.*, 2015. 3

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 3

[36] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *CoRR*, abs/1906.05849, 2019. 2, 4

[37] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. 2008. 2

[38] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *Int. Conf. Comput. Vis.*, 2017. 2

[39] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2, 5, 6

[40] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019. 6

[41] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 3

[42] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Brit. Mach. Vis. Conf.*, 2016. 3

[43] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Eur. Conf. Comput. Vis.*, 2016. 1, 2, 5

[44] Xiao Zhang and Michael Maire. Self-supervised visual representation learning from hierarchical grouping. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2

[45] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Int. Conf. Comput. Vis.*, 2019. 2, 5