Progressive Stage-wise Learning for Unsupervised Feature Representation Enhancement

Zefan Li[14*] leezf@sjtu.edu.cn
Chenxi Liu[2†] cxliu@waymo.com
Alan Yuille[2] alan.l.yuille@gmail.com
Bingbing Ni[14‡] nibingbing@sjtu.edu.cn
Wenjun Zhang[1] zhangwenjun@sjtu.edu.cn
Wen Gao[3] wgao@pku.edu.cn

[1]Shanghai Jiao Tong University [2]Johns Hopkins University [3]Peking University
[4]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University

Abstract

*Unsupervised learning methods have recently shown their competitiveness against supervised training. Typically, these methods use a single objective to train the entire network. But one distinct advantage of unsupervised over supervised learning is that the former possesses more variety and freedom in designing the objective. In this work, we explore new dimensions of unsupervised learning by proposing the **P**rogressive **S**tage-wise **L**earning (**PSL**) framework. For a given unsupervised task, we design multi-level tasks and define different learning stages for the deep network. Early learning stages are forced to focus on low-level tasks while late stages are guided to extract deeper information through harder tasks. We discover that by progressive stage-wise learning, unsupervised feature representation can be effectively enhanced. Our extensive experiments show that PSL consistently improves results for the leading unsupervised learning methods.*

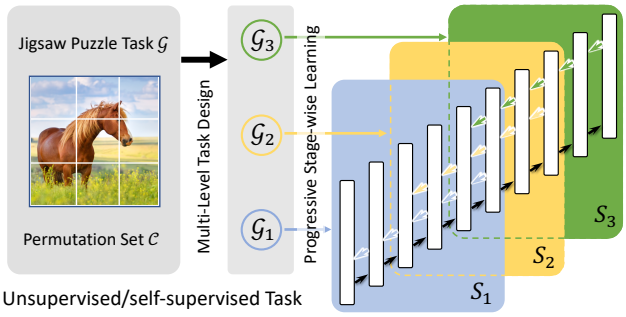

Figure 1. We present the framework of the proposed Progressive Stage-wise Learning (**PSL**) algorithm, aiming for improving unsupervised/self-supervised task. We take the jigsaw puzzle task $\mathcal{G}$ for example. We first do multi-level task partition $\mathcal{G} \rightarrow \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ with an increased task complexity and perform progressive stage-wise training for different learning stages of the network. The black arrow denotes forward pass while colored arrow represents the backward pass of each learning stage (i.e., S_1, S_2, and S_3).

1. Introduction

Aiming at learning features from label-free data, unsupervised representation learning, including self-supervised learning, is an important problem to study. Many efforts have been made, to bridge the performance gap between supervised and unsupervised learning algorithms. These methods can be roughly divided into two categories: i) handcrafted pretext tasks, that learns data-level invariant features (e.g., jigsaw puzzle [38], image rotation [21], image colorization [13]) and ii) contrastive visual representation learning, which learns the similarity and dissimilarity between data pairs [9, 10, 11, 26]. For approaches using pretext tasks, they usually generate pseudo labels based on some data attributes and learn visual features through corresponding objective functions of the pretext tasks. Therefore, the final performance of these approaches is highly related to how the pretext tasks were initially designed. Most pretext tasks are designed heuristically, limiting the quality of learned representation. Contrastive learning methods usually generate positive/negative sample pairs through a set of image transformations and learn visual representation by bringing positive sample pairs closer while pushing negative sample pairs away from each other. The design of the contrastive loss and the configuration of image transformations are essential to the quality of the resulting net-

*Work done while visiting Johns Hopkins University.
†Now at Waymo.
‡Corresponding author.

works. Those methods have shown great promise in the area of unsupervised learning, achieving state-of-the-art results [25, 16, 40, 2]. Some recent methods show it is possible for unsupervised learned features to surpass supervised learning in some downstream applications [9]. However, performance gaps still exists between unsupervised and supervised learning methods in most cases [23]. Therefore, how to fully explore the potential of unsupervised learning and improve the learning quality is a valuable topic.

Instead of designing a new pretext task or a better contrastive learning loss, we try to look into this problem from a new perspective. As curriculum learning [5] suggests, when dealing with a complex learning target, learning things progressively can be very useful. Indeed, as humans, we learn visual concepts from easy to hard, and from elementary to fine-grained. Similarly, learning high-quality feature representations in an unsupervised manner is a challenging task, and may benefit from such ideas. In this paper, we propose ***PSL***, a progressive stage-wise learning framework for unsupervised visual representation learning.

As presented in Fig 1, for a given unsupervised learning task $\mathcal{G}$ (e.g., the jigsaw puzzle pretext task), we first do ***multi-level task design*** $\mathcal{G} \rightarrow \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ with an increased task complexity. Then, a ***stage-wise network partition*** is performed to get early/mid/late stages (i.e., S_1, S_2 and S_3). Each learning stage is assigned with a task, following the principle of easy-to-hard. Then, a ***stage-wise training*** is performed. The training of lower stages become much easier as they focus on more simple tasks. The feature representations learned in upper stages are of better quality because they are trained upon the learning experience of former tasks. Our starting point is to design PSL as a plug-in learning method, which can be applied in any unsupervised learning scenarios under proper multi-level task design. We validate the effectiveness of the proposed PSL framework by evaluating our method on several unsupervised/self-supervised tasks (e.g., the jiasaw puzzle [38] and image rotation [21] pretext task and contrastive learning [10]) and present results on linear classification, semi-supervised learning and transfer learning.

In general, the contributions of this paper can be summarized as follows:

- PSL creates new dimensions for unsupervised learning research. Specifically, this includes task series, network partitions, and stage-wise training.
- PSL is design to be a general framework, that can be applied to multiple unsupervised learning tasks belonging to either pretext tasks or contrastive learning (e.g., jigsaw puzzle, image rotation, and SimCLR).
- By experiments of downstream applications (e.g., semi-supervised learning, transfer learning), we show that the feature representations learned by PSL consistently achieve better quality than the original unsupervised task.

2. Related Work

Our method falls in the area of unsupervised visual representation learning. We first revisit two categories of unsupervised learning method. Then, we review methods involving self-paced learning and local network training, which give inspiration in our PSL training scheme.

Handcrafted Pretext Tasks. Many self-supervised methods use a handcrafted pretext task to learning visual representations. Typically, a pretext task involves predicting an explicit property of an image transformation and the network is then trained to learn the feature representation. The quality of the learned representation is highly related to these tasks (e.g., predicting context [14], image rotation [21], image colorization [13, 49, 30, 32, 33], jigsaw puzzle [38, 6, 23] and visual counting [39]). Instead of designing a new pretext task, we propose a plug-in method to enhance the self-supervised learning, which can be used collaboratively with many pretext tasks.

Contrastive learning. Unsupervised contrastive learning recently attracts lots of attention, for achieving state-of-the-art results on ImageNet [10, 26, 9]. Typically, a contrastive learning method learns feature representations by contrasting positive pairs against negative pairs, which is firstly proposed by Hadsell *et al.* [25]. Then, Dosovitskiy *et al.* [16] propose to represent each instance with a parametric vector. Later, the concept of memory bank, which stores the information of instance class representation, is adopted and developed further in many works [51, 44, 37]. Besides, there are many clustering-based methods [7, 8, 1, 20, 29, 47]. Our PSL training scheme can also fit into contrastive learning methods, bringing improvement to the quality of the learned feature representation.

Self-paced Learning Many self-paced learning methods simulate the learning process of "easy-to-hard" [24, 19, 42, 18]. [24] incorporates self-paced learning into deep clustering methods by controlling the number of selected data samples. [19] uses a self-paced learning strategy by identifying reliable and unreliable clusters to improve the accuracy in the re-clustering step. These methods are based on data-level information by defining easy/hard data samples while our method focus on the task-level design.

Local Network Learning. The end-to-end training protocol inaugurated a new era in deep learning. Some works jump out of traditional forward-backward training mode, with inspiration from neuroscience. An early research [35] shows that the way of the brain processing its perceptions

is to maximally preserve the information contained in each layer. Several methods try to explore greedy layer-wise training schemes [4, 3], in which the possibility of scaling this scheme to ImageNet is discussed. Later, GIM [36] argues that with greedy self-supervised training, end-to-end propagation of a supervised loss is not necessary. Based on GIM [36], LoCo [48] improves the performance of local contrastive learning. Inspired by these local training strategy, we propose a stage-wise training algorithm and improve the performance in multiple unsupervised learning task.

3. Method

3.1. Overview

This work aims to introduce an unsupervised pretraining strategy. Many previous methods focus on designing handcrafted pretext tasks in a self-supervised learning setting [38, 21, 13]. Correspondingly, the results are highly related to how the pretext tasks are designed. Another type of work focuses on exploring the potential behind contrastive learning. These works concentrate on learning similar/dissimilar representations from organized similar/dissimilar data pairs. Both kinds are trying to uncover internal information of unlabeled data. As unsupervised learning becomes more and more important and challenging, how to take the best advantage of existing unsupervised learning ways is vital. In another word, how to do unsupervised learning more effectively?

In this work, we provide a new dimension in enhancing the unsupervised learning representation. Inspired by curriculum learning, we try to guide the neural network to learning feature representations in a progressive way (e.g., from easy tasks to hard tasks, from low-level features to high-level). To do so, we introduce our progressive stage-wise learning (PSL) framework for unsupervised learning.

3.2. Progressive Stage-wise Learning

In this sub-section, we explain how our progressive stage-wise learning (PSL) works. Suppose we have a learning target $\mathcal{G}$, which can be an unsupervised contrastive learning task or any pretext task in the self-supervised learning setting. We use a neural network with a block-based architecture (designed by stacking blocks vertically, such as ResNet [27] and Inception [43]). What we do can be summarized into the following steps:

Multi-level Task Design Firstly, for the given learning target $\mathcal{G}$, we design a series of learning tasks that share a similar form (e.g. the same pretext task) but with different task complexity. We sort these tasks according to their complexity: $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$, where $\mathcal{G}_3$ is more complex than $\mathcal{G}_2$, and so is $\mathcal{G}_2$ compared to $\mathcal{G}_1$. We use $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ to represent the corresponding loss function. Instead of focusing on the hardest task (e.g., $\mathcal{G}_3$) at the very beginning, we enable easier learning targets in the early learning stage. By doing so, we can train the neural network in a progressive learning manner, which turns out to be more efficient in many different unsupervised settings. We introduce our multi-level task design for specific tasks in Sec 3.3.

Stage-wise Network Partition In this step, we define different learning stages, which are basically determined by the layer depth. We take ResNet-50 for an example. Based on the resolution of the feature map, we divide all layers into five large block: B_1, B_2, B_3, B_4 and B_5, where B_1 represents the layer `conv1`, B_2 consists of all layers named `conv2_x`, and so on[1]. Further, we group every three consecutive blocks into one learning stage: Stage S_1 consists of B_1, B_2 and B_3; Stage S_2 consists of B_2, B_3 and B_4; Stage S_3 consists of B_3, B_4 and B_5. By doing so, we get three stages (S_1, S_2 and S_3), representing lower, mid and upper learning stages of the network. Notice that there are overlaps between each two learning stages, which is discussed more in Sec 3.4.

Progressive Network Learning After we get a series of multi-level tasks and a proper stage partition, we begin the training process. As shown in Fig 2, the traditional end-to-end training protocol, forces the whole neural network to learn the final target directly. Instead, we train the neural network progressively in a stage-wise manner. More specifically, we set a different learning target for each stage, following the principle of easy-to-hard. We force lower layers of the network (e.g., layers in S_1) to learn low-level features by solving easier tasks (e.g., $\mathcal{G}_1$). As the lower part of the network learns a good feature representation for the easy task, we increase the task complexity for the mid and upper stages by targeting at the task $\mathcal{G}_2$ and $\mathcal{G}_3$. By doing so, we naturally guide the network to gain better feature representation ability as the layer goes deeper. The benefits are three-fold. Firstly, the lower stage only has to focus on the easy task, which leads to easier training. Secondly, the upper stage can take advantage of the lower stages through weight sharing when dealing with a harder task. Thirdly, the backward propagation path is much shorter within each learning stage compared to end to end training. Without gradient error accumulation, the overall training can be much more efficient and effective.

More details of our framework are shown in Fig 2. The training targets $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ are assigned to stages

[1]Here, all layers within the same block shares the same feature map resolution. The layer name `conv1` and `conv2_x` are inherited from the notation of ResNet [27].

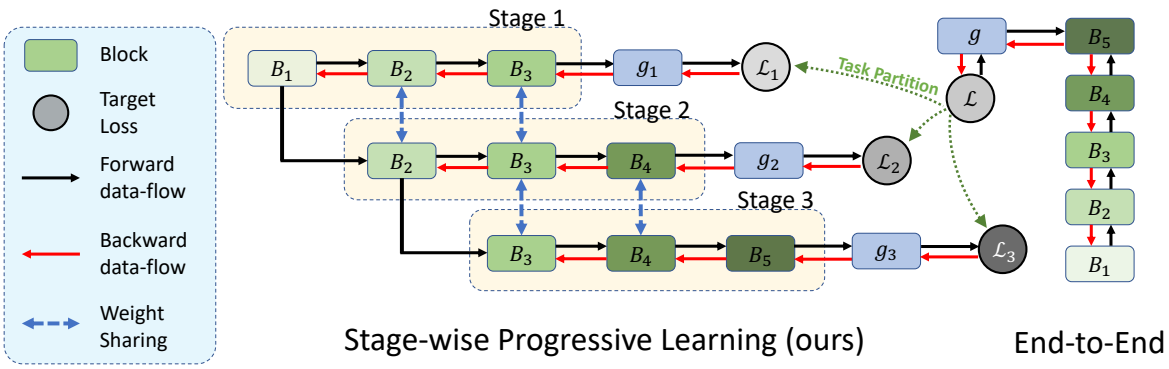

Figure 2. We present the detail of the proposed Stage-wise Progressive Learning framework. In the right is the end-to-end learning scheme while we present PSL in the middle. g and $\{g_i\}_{i=1}^{3}$ are projection heads, mapping the intermediate representation to the target feature space. After the training is completed, we throw away the projection heads and use the backbone network for downstream tasks.

$\{S_1, S_2, S_3\}$ respectively. Notice that different from end-to-end training, backward gradients only flow back to layers in the same stage.

3.3. Multi-level Task Design Cases

In this sub-section, we introduce our multi-level task design for several different unsupervised/self-supervised learning tasks.

3.3.1 Multi-level Jigsaw Puzzle

The jigsaw puzzle task [38] was first introduced to learn visual representations from unlabeled data, which turns out to be very useful in many downstream tasks, such as detection and classification. To create a jigsaw puzzle, a 225×225 pixel window is randomly cropped from an image. Then, the whole window is divided into a 3×3 grid, leading to nine 75×75 pixel cells. For each cell, a 64×64 pixel tile is picked randomly. Then an index permutation is sampled (e.g., $\{1,2,3,4,5,6,7,8,9\} \rightarrow \{3,5,7,8,4,6,9,2,1\}$) from a pre-defined permutation set $\mathcal{C}$. The obtained 9 tiles are reordered according to the permutation. Finally, the reordered tiles of the puzzle are stacked along the channels and fed to the neural network to predict the permutation, which is usually a classification task. There are several factors that influence the complexity of the jigsaw puzzle. According to [38], the permutation set $\mathcal{C}$ influences the jigsaw task from two aspects: the set cardinality of $\mathcal{C}$ and the element similarity of $\mathcal{C}$. Generally, the difficulty of the jigsaw task increases as the set cardinality increases or the element similarity decreases. Under the original task setting, there are $9! = 362880$ different permutations in total for every 9 tiles. Therefore, it is nearly impossible to include all permutations in the permutation set. Previous methods [38, 6] usually define a permutation set with a fixed size (e.g., 1000) in advance.

Task	Set†	Cardinality	Hamming
$\mathcal{G}_1$	$\mathcal{C}_1$	500	~8.0
$\mathcal{G}_2$	$\mathcal{C}_2$	1000	~8.0
$\mathcal{G}_3$	$\mathcal{C}_3$	2000	~8.0

Table 1. Multi-level task design for jigsaw puzzle. Set† denotes the permutation set.

In this work, we design multi-level jigsaw puzzle tasks by changing cardinality of $\mathcal{C}$. As shown in Table 1, we generate three permutation sets $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ with cardinality 500, 1000, and 2000. Notice that $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \mathcal{C}_3$. We keep the average hamming distance of each permutation set around 8.0 so that the element similarity within each set stays in the same level. The task complexity increases as permutation gets bigger.

In addition to the change of cardinality of permutation sets, there are other methods to control the task complexity. For example, one can increase or decrease the size of the grid (e.g. from 3×3 to 2×2 and 4×4). However, the resulting difficulty gap between adjacent tasks is too larger under our multi-level jigsaw puzzle design. [23] reported that increasing the number of patches (i.e. from 9 to 16) does not necessarily result in a higher quality representation. Therefore, we do not adopt this scheme in this work.

3.3.2 Multi-level Image Rotation

The Image rotation task was firstly designed in self-supervised learning [21]. Image rotation can be considered as one of the image geometric transformations, which is very easy to perform. The core idea for this task is to use the neural network to estimate geometric transformation (i.e., the angle of rotation in this case). A common practice is to define the set of geometric transformation $\mathcal{R}$

Task	Set†	Cardinality	Angle Base
$\mathcal{G}_1$	$\mathcal{R}_1$	2	180°
$\mathcal{G}_2$	$\mathcal{R}_2$	4	90°
$\mathcal{G}_3$	$\mathcal{R}_3$	8	45°

Table 2. Multi-level task design for image rotation. Set† denotes the rotation transformation set.

Task	Set†	scheme
$\mathcal{G}_1$	$\mathcal{T}_1$	random crop
$\mathcal{G}_2$	$\mathcal{T}_2$	crop+color distortion
$\mathcal{G}_3$	$\mathcal{T}_3$	crop+color distortion+filtering

Table 3. Multi-level task design for contrastive learning. Set† denotes the transformation set.

as all the image rotations of 90 degrees (e.g., 2d image rotations by 0, 9, 180 and 270 degrees). Here, we use the size of transformation set $\mathcal{R}$ to control the task difficulty. As shown in Table 2, we defined three set of geometric transformations as all image rotations of 180/90/45 degree, with 2/4/8 operations within each set respectively. For example, $\mathcal{R}_1 = \{0°, 180°\}$, $\mathcal{R}_2 = \mathcal{R}_1 \cup \{90°, 270°\}$ and $\mathcal{R}_3 = \mathcal{R}_2 \cup \{45°, 135°, 225°, 315°\}$. Notice that, $\mathcal{R}_1 \subseteq \mathcal{R}_2 \subseteq \mathcal{R}_3$, which indicates an increase in task complexity.

3.3.3 Multi-level Contrastive Learning

Contrastive learning has recently become a dominant approach in the area of self-supervised learning [10, 26, 11]. Typically, contrastive learning approaches learn representations by contrasting positive pairs against negative pairs. For example, SimCLR [10] makes use of multiple data augmentation operations to generate positive data pairs. Then a based encoder network is trained to maximize the similarity of positive data pairs meanwhile minimize the similarity of negative data pairs using a contrastive loss. We design our multi-level contrastive tasks based on the setting of SimCLR [10]. We control the task difficulty by manipulating the augmentation set $\mathcal{T}$. For the low-level task (e.g., $\mathcal{G}_1$), we use a simple augmentation scheme and we increase the complexity of the augmentation scheme for high-level tasks (e.g., $\mathcal{G}_3$). Three kinds of augmentation are adopted: i) geometric transformation of data, such as cropping and resizing; ii) appearance transformation, such as color distortion and iii) other transformation, such as Gaussian blur and Sobel Filtering. As shown in Table 3, we set $\mathcal{T}_1 = \{\text{Random Crop}\}$ as the first augmentation set for task $\mathcal{G}_1$ and we add color distortion into the augmentation set ,$_2$ for task $\mathcal{G}_2$. Other transformation operations are included in the augmentation set $\mathcal{T}_3$ for task $\mathcal{G}_3$. Notice that $\mathcal{T}_1 \subseteq \mathcal{T}_2 \subseteq \mathcal{T}_3$.

Algorithm 1 Progressive Stage-wise Learning Algorithm.

Input: Learning Target $\mathcal{G}$, Learning Loss $\mathcal{L}$, backbone Network f, projection head g_1, g_2 and g_3, batch size N

$\mathcal{G} \rightarrow \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ ▷ Multi-level Task Design
$\mathcal{L} \rightarrow \{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3\}$
$f \rightarrow \{S_1, S_2, S_3\}$ ▷ Stage-wise Network Partition
for $i \in \{1, 2, 3\}$ **do** ▷ Stage-wise Learning
  for sampled minibatch $\{x_k\}_{k=1}^{N}$ **do**
    Data pre-processing for task $\mathcal{G}_i$
    $h_k = f(x_k|S_i)$ ▷ Forward propagation
    $z_k = g_i(h_k)$
    Computer gradient with respect to $\mathcal{L}_i(x_k, z_k)$
    Update layers within S_i
    Update g_i
  end for
end for
Return f, and discard g_1, g_2 and g_3

3.4. Stage-wise Network Training

As explained in section 3.2, three learning tasks $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ are obtained, corresponding to three losses $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3\}$ and stages S_1, S_2, S_3. Unlike traditional end-to-end learning, we adopt a local learning strategy. To be specific, the gradient of each learning stage does not flow back to other stages. For example, gradients generated by $\mathcal{L}_2$ only influence layers within S_2 (i.e.,B_2, B_3 and B_4) during the second learning stage. This allow the corresponding layers focus on the current learning target, which turns out to be good for overall training. Our learning algorithm is summarized in Algorithm 1. After we finish multi-level task design and stage-wise network partition, we start progressive stage-wise learning. There are three learning stages in total. During the i-th stage, we do data pre-processing first, which is determined by the specific learning task $\mathcal{G}_i$. Then we do forward propagation with $h_k = f(x_k|S_i)$ representing the output feature of stage S_i. Then, h_k is sent to a decoder g_i for further processing before applying the stage loss $\mathcal{L}_i$. Only layers within S_i and the decoder g_i are updated. After the training, all decoders will be removed and no extra computation cost is introduced in f.

Notice that there are overlappings between each stage. Another stage partition approach is to cut the encoder into several non-overlapping parts and train the whole network in a greedy layer-wise manner like GIM [36]. In the case of GIM, upper layers/stages cannot receive gradient feedback from lower layers/stages. However, as the difficulty of the multi-level task increases, the quality of the intermediate representation of lower stages has a large influence on the final performance of the upper stages. Therefore, it is necessary for stages to have overlapping layers, which play a role in connecting and communicating between stages.

4. Experiment

In this section, we conduct experiments to validate the effectiveness of the proposed Progressive Stage-wise Learning (PSL) framework. Firstly, we apply PSL on several different kinds of unsupervised/self-supervised learning tasks to evaluate the quality of the learned representation on ImageNet [12]. Then, we report experiment results of downstream tasks, i.e., semi-supervised and transfer learning.

4.1. Implementation Details

Generally, we conduct contrast experiments on three different unsupervised/self-supervised learning methods. The basic implementation details for each task as follows:

Jigsaw puzzle. We perform multi-level task design $\mathcal{G} \rightarrow \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ following Sec 3.3.1. Correspondingly, we use three different cardinality $\{500, 1000, 2000\}$ for the permutation set $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$, with $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \mathcal{C}_3$, representing an increase in task complexity. We use ResNet-50 [27] as our backbone network and do PSL training on 8-gpus. For each input image, we resize the shorter side to resolution 256, randomly crop a 225x225 image and apply horizontal flip with 50% probability. For the training, we use mini-batch size of 256, initial learning rate of 0.01 with the learning rate dropped by a factor of 10. Following [23], we use momentum of 0.9, weight decay 1e-4 with no decay for the bias parameters. For each training stage of PSL, we train for 60 epochs and use the learning rate schedule of 20/20/10/10.

Image rotation. We perform multi-level task design $\mathcal{G} \rightarrow \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ following Sec 3.3.2. Correspondingly, we set the rotation transformation set $\mathcal{R}_1 = \{0°, 180°\}$, $\mathcal{R}_2 = \{(90 \times i)° | 0 \leq i \leq 3\}$ and $\mathcal{R}_3 = \{(45 \times i)° | 0 \leq i \leq 7\}$. Notice that $\mathcal{R}_1 \subseteq \mathcal{R}_2 \subseteq \mathcal{R}_3$, representing an increase in task complexity. We use RevNet50 [22] as our backbone network and do PSL training on 8-gpus[2]. For each input image, we resize the shorter side to 256 and do the rotation transformation according to the transformation set. We perform a center crop on the rotated image remaining the resolution and then randomly crop 224x224 image. For the training, we use mini-batch size of 256, initial learning rate of 0.01 with the learning rate dropped by a factor of 10. For each training stage of PSL, we train for 60 epochs and use the learning rate schedule of 20/20/10/10.

Contrastive learning. We perform multi-level task design $\mathcal{G} \rightarrow \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ following Sec 3.3.3. Correspondingly, we set the transformation set as: $\mathcal{T}_1 = \{\text{Random crop}\}$, $\mathcal{T}_2 = \mathcal{T}_1 \cup \{\text{Color Distortion}\}$ and $\mathcal{T}_3 = \mathcal{T}_2 \cup \{\text{Gaussian Blur, Sobel Filtering}\}$. Notice that $\mathcal{T}_1 \subseteq \mathcal{T}_2 \subseteq \mathcal{T}_3$, representing an increase in task complexity. We use ResNet-50 as our backbone network and a 2-layer MLP projection head to project the representation to a 128-dimensional space. Limited by computing resources, we use mini-batch size of 256 and train for 200 epochs (60/70/70 epochs for each learning stage) on 8-gpus. Notice that the max performance is not obtained in 200 epochs and 256 batch-size[3], reasonable results and fair comparison can still be achieved. Based on SimCLR [10], We use NT-Xent loss, learned in LARS optimizer with learnig rate of 0.3 with a cosine decay schedule without restart. Similarly, we apply PSL on the data-augmentation part of MoCov2 [11] and get an improvement of 0.6%.

[2]Compared with ResNet [27], RevNet [22] is more suitable in image rotation tasks [31]. We don't use RevNet50w4×, which is reported with better performance, because scaling up model complexity is not discussed in this paper.

Method	Arch	# Param(M)	Top 1
Colorization [49]	R50	24	39.6
BigBiGAN [15]	R50	24	56.6
LA [51]	R50	24	58.8
NPID++ [37]	R50	24	59.0
MoCo [26]	R50	24	60.6
PIRL [37]	R50	24	63.6
CPC v2 [28]	R50	24	63.8
PCL [34]	R50	24	65.9
SwAV [9]	R50	24	75.3
Jigsaw [38]	R50	24	45.7
Jigsaw +PSL	R50	24	**50.9**
Rotation [21]	Rv50w4×	86	47.3
Rotation*	Rv50	24	48.6
Rotation+PSL	Rv50	24	**53.3**
SimCLR [10]	R50	24	61.9
SimCLR+PSL	R50	24	**64.3**
MoCov2 [11]	R50	24	67.5
MoCov2+PSL	R50	24	**68.1**

Table 4. ImageNet accuracy of linear classifiers trained on self-supervised learned representations. All are reported as unsupervised pre-training on ImageNet, followed by supervised linear classification and evaluated on the ImageNet validation set. Note that Rotation [21] uses $\mathcal{R}_2$ as the transformation set while Rotation* uses $\mathcal{R}_3$ as the transformation set. SimCLR results are obtained by 200 training epochs with batchsize 256.

4.2. Linear Classification

In this subsection, we verify our method by linear classification on ImageNet [12]. Following a common protocol, We conduct contrast experiments on three different unsupervised/self-supervised learning methods. Firstly, we perform unsupervised pre-training on ImageNet. Then, we train a supervised linear classifier (a fully-connect layer followed by softmax). Table 4 summaries the single crop

[3]1000 epochs with batch-size 4096 is reported with the best performance in SimCLR [10].

Method	**Arch**	B_1	B_2	B_3	B_4	B_5
Supervised	R50	11.6	33.3	48.7	67.9	75.5
Jigsaw	R50	12.4	28.0	39.9	45.7	34.2
SL	R50	10.8	27.2	39.6	45.5	34.7
PSL	R50	10.9	27.0	43.2	**50.9**	38.5
PSL_f	R50	10.8	27.3	43.1	51.0	38.2

Table 5. Jigsaw Puzzle task of ResNet-50 top-1 centrer-crop accuracy for linear classification on the ImageNet dataset. Here $B_1 \sim B_5$ represent the blocks defined in Sec 3.2. The supervised results are presented for reference. Jigsaw is end-to-end trained with the task $\mathcal{G}_3$. SL is short for stage-wise learning. In SL, we do stage-wise training with the jiasaw puzzle task $\mathcal{G}_3$ as the learning target of each stage. PSL_f is a full-gradient version of PSL. More discussion can be found in ablation in Sec 4.5.

Method	**Arch**	B_1	B_2	B_3	B_4	B_5
Supervised	Rv50	11.7	32.6	47.8	66.6	74.3
Rotation	Rv50	10.9	30.1	40.2	48.6	46.5
SL	Rv50	11.1	29.1	41.5	50.7	47.9
PSL	Rv50	11.3	30.8	42.9	**53.3**	49.5
PSL_f	R50	10.5	31.1	42.1	52.9	49.7

Table 6. Image rotation task of RevNet-50 top-1 centrer-crop accuracy for linear classification on the ImageNet dataset. Here $B_1 \sim B_5$ represent the block defined in Sec 3.2. The supervised results are presented for reference. Rotation is end-to-end trained with the task $\mathcal{G}_3$. SL is short for stage-wise learning. In SL, we do stage-wise training with the image rotation task $\mathcal{G}_3$ as the learning target of each stage. PSL_f is a full-gradient version of PSL. More discussion can be found in ablation in Sec 4.5.

top-1 classification accuracy on the ImageNet validation set, comparing our results with previous approaches [49, 15, 51, 37, 26, 28, 34, 11, 9] as well as three baseline method [38, 21, 10]. Compared with the vanilla jigsaw puzzle task, PSL training improves the results by 5.2% (45.7%→ 50.9%). Compared with the vanilla image rotation task, PSL training improves the results by 4.7% (48.6%→ 53.3%). As for the SimCLR contrastive learning, PSL training improves the results by 2.4% (61.9% → 64.3%), under the setting of 200 training epoch and batch-size 256. These results indicate the PSL training can effectively improve the unsupervised learning quality. Specifically, we extract image features from five different layers (i.e., the output of B_1, B_2, B_3, B_4 and B_5) after unsupervised pre-training and train linear classifiers on these fixed representations. For the jigsaw puzzle task, detailed results are shown in Table 5. Results of the image rotation task are presented in Table 6.

Task	1% labels	10% labels
Supervised	48.4	80.4
Jigsaw [38]	45.4	79.6
Jigsaw+PSL	**48.7**	**83.5**
Rotation [21]	52.1	82.5
Rotation+PSL	**54.8**	**83.7**

Table 7. **Semi-supervised learning on ImageNet.** We use ResNet-50 as our backbone networks and report single-crop top-5 accuracy on the ImageNet validation set. All models are self-supervised trained on ImageNet and finetuned on 1% and 10% of the ImageNet training data, following [46, 37]. The supervised results are presented for reference.

4.3. Semi-supervised Learning

Following the experiment setup of [46, 37], we perform semi-supervised image classification on ImageNet to evaluate the effectiveness of the pre-trained network in data-efficient settings. We do a class-balanced data selection to obtain 1% and 10% of the ImageNet training data and fine-tuned the whole pre-trained network. Table 7 reports the top-5 accuracy of the resulting models on the ImageNet validation set. By contrast experiments on two pretext tasks, the proposed PSL training method can improve the top-5 accuracy by a large margin. In the jigsaw puzzle task, PSL brings an improvement of 3.3% and 3.9% for 1% and 10% labeled data respectively. For the image rotation task, PSL improves the performance by 2.7% and 1.2% respectively.

4.4. Transfer Learning

To further investigate the generalization ability of our method, we conduct transfer learning experiments including object detection on PASCAL VOC [17] and image classification results on three datasets. In this subsection, we use ResNet-50 as our backbone network both jigsaw puzzle and image roration tasks. Results are reported in Table 8 and Table 9.

4.4.1 Object Detection

Following previous works [23, 37], we perform object detection experiments on the PASCAL VOC dataset [17] using VOC07+12 training split. We use Faster RCNN [41] object detector and ResNet-50 C4 [27] backbone. We follow the same training schedule as [23, 37] to finetune all models on VOC with BatchNorm parameters fixed during finetuning. We report our performance of $\{\text{AP}_{50}, \text{AP}_{75}, \Delta\text{AP}_{75}\}$ in Table 8. Compared with the vanilla pretext task, our PSL training scheme can substantially improve the three indicators, representing an enhancement of the learned feature representation. In the jigsaw puzzle pretext task, we get an improvement of $\{2.2, 3.8, 2.4\}$ respectively, while in the

Task	Object Detection			
	AP_{all}	AP_{50}	AP_{75}	ΔAP_{75}
Supervised	52.6	81.1	57.4	0.0
Jigsaw [38]	48.9	75.1	52.9	-4.5
Jigsaw+PSL	**51.1**	**78.9**	**55.3**	**-2.1**
Rotation [21]	46.3	72.5	49.3	-8.1
Rotation+PSL	**47.6**	**74.6**	**51.1**	**-6.3**

Table 8. Object detection results of transfer learning on PASCAL VOC dataset. We report AP on the test set after finetuning Faster R-CNN modelswith a ResNet-50 backbone, that are unsupervised pre-trained on ImageNet. The supervised results are presented for reference.

Task	Transfer Dataset		
	PASCAL	Places	iNat18
Supervised	87.5	51.5	45.4
Jigsaw [38]	64.5	41.2	21.3
Jigsaw+PSL	**67.2**	**44.7**	**24.1**
Rotation [21]	63.5	41.9	23.0
Rotation+PSL	**65.9**	**43.0**	**25.9**

Table 9. Image classification results of transfer learning on PASCAL [17], Places [50] and iNat18 [45]. We use linear classifiers on image representations obtained by self-supervised learners that are pre-trained on ImageNet. We report mAP on the PASCAL dataset and top-1 accuracy on Places and iNat18. We compare results on the jigsaw puzzle and image rotation pretext tasks with the proposed PSL. The supervised results are presented for reference.

image rotation pretext task, PSL brings an improvement of $\{1.3, 2.1, 1.8\}$.

4.4.2 Image Classification on other datasets

Next, we conduct transfer learning experiments on the image classification task. We use models pre-trained on ImageNet and assess the quality of learned features by training linear classifiers on fixed image representations. Following the setting of [23], we evaluate feature representations extracted from five intermediate blocks of the pre-trained network, and report the best classification results in Table 9. We report transfer learning performance on PASCAL [17], Places [50] and iNat18 [45]. In PASCAL, our PSL training improves the jigsaw puzzle task by 2.7%, and image rotation task by 2.4%. In Places, the PSL training improves the performance by 3.5% and 1.1% while in iNat18, the improvement is 2.8% and 2.9% for these two tasks respectively.

4.5. Analysis

Ablation: Progressive Mechanism. Here we discuss the effectiveness of the progressive mechanism. We design comparison experiments in ImageNet linear classification to compare stage-wise learning w/o progressive learning, namely **PSL** *vs.* **SL**. For SL, we use the same learning task ($\mathcal{G}_3$) in each learning stage, which means the network is trained to learn the hardest task for each learning stage. For PSL, the task complexity increases for learning stages S_1, S_2 and S_3. We present PSL *vs.* SL in Table 5 and Table 6. For both tasks, PSL leads to better results than SL without the progressive mechanism.

Ablation: Gradient Association. As discussed in Section 3.4, we adopt a stage-wise training scheme with limited a gradient association in each learning stage. Specifically, the gradient of $\mathcal{L}_2$ will not flow back to B_1 and the gradient of $\mathcal{L}_3$ will not influence B_1 and B_2. We implement the full gradient version of PSL where in each learning stage, all layers are updated without gradient restriction. We show the linear classification results on ImageNet dataset as PSL_f in Table 5 and Table 2. From the results, we conclude that full gradient training does not necessarily bring an improvement in the unsupervised training performance. Therefore, we set gradient restriction in each learning stage, which will also reduce the computation cost during training.

Local Training. We do not adopt a greedy block-wise learning scheme like [36]. Instead, we enhance stage-wise the connection by enabling graident flow between stages. A similar approach is adopted in LoCo [48], where gradient connections are enabled in adjacent blocks. In PSL, block connections are further enhanced (e.g., learning stage S_3 have impact on B_3, which is also included in learning stage S_1). By comparing PSL and PSL_f in Table 5 and 6, we show that this design helps enhance the learning quality of various unsupervised tasks.

5. Conclusion

In this work, we present a Progressive Stage-wise Learning (**PSL**) framework for unsupervised/self-supervised learning. Through multi-level task design and progressive stage-wise training, PSL improves many mainstream unsupervised methods. We provide experiments in three different tasks (i.e., jigsaw puzzle, image rotation and contrastive learning) in this paper and validate the effectiveness of PSL. Our future work involves exploring PSL on other tasks and searching for the optimal architecture for PSL framework. We hope PSL can be a universal unsupervised training approach in enhancing the learned feature representation.

Acknowledgements. This work was supported by National Science Foundation of China (U20B2072, 61976137), NSFC(U19B2035), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), ONR N00014-20-1-2206 and China Scholarship Council (NO.201906230178).

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019. 2

[2] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Adv. Neural Inform. Process. Syst.*, pages 15535–15545, 2019. 2

[3] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International Conference on Machine Learning*, pages 583–593. PMLR, 2019. 3

[4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Adv. Neural Inform. Process. Syst.*, 19:153–160, 2006. 3

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International conference on machine learning*, pages 41–48, 2009. 2

[6] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2229–2238, 2019. 2, 4

[7] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Eur. Conf. Comput. Vis.*, pages 132–149, 2018. 2

[8] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *Int. Conf. Comput. Vis.*, pages 2959–2968, 2019. 2

[9] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Adv. Neural Inform. Process. Syst.*, 33, 2020. 1, 2, 6, 7

[10] 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, 5, 6, 7

[11] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 5, 6, 7

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–255. Ieee, 2009. 6

[13] Aditya Deshpande, Jason Rock, and David Forsyth. Learning large-scale automatic image colorization. In *Int. Conf. Comput. Vis.*, pages 567–575, 2015. 1, 2, 3

[14] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Int. Conf. Comput. Vis.*, pages 1422–1430, 2015. 2

[15] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Adv. Neural Inform. Process. Syst.*, pages 10542–10552, 2019. 6, 7

[16] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Adv. Neural Inform. Process. Syst.*, pages 766–774, 2014. 2

[17] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.*, 111(1):98–136, 2015. 7, 8

[18] Hongchang Gao and Heng Huang. Self-paced network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1406–1415, 2018. 2

[19] Yixiao Ge, Dapeng Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Self-paced contrastive learning with hybrid memory for domain adaptive object re-id. *arXiv preprint arXiv:2006.02713*, 2020. 2

[20] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6928–6938, 2020. 2

[21] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Int. Conf. Learn. Represent.*, 2018. 1, 2, 3, 4, 6, 7, 8

[22] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. In *Adv. Neural Inform. Process. Syst.*, pages 2214–2224, 2017. 6

[23] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Int. Conf. Comput. Vis.*, pages 6391–6400, 2019. 2, 4, 6, 7, 8

[24] Xifeng Guo, Xinwang Liu, En Zhu, Xinzhong Zhu, Miaomiao Li, Xin Xu, and Jianping Yin. Adaptive self-paced deep clustering with data augmentation. *IEEE Transactions on Knowledge and Data Engineering*, 32(9):1680–1693, 2019. 2

[25] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conf. Comput. Vis. Pattern Recog.*, volume 2, pages 1735–1742. IEEE, 2006. 2

[26] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9729–9738, 2020. 1, 2, 5, 6, 7

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016. 3, 6, 7

[28] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 6, 7

[29] Jiabo Huang, Qi Dong, Shaogang Gong, and Xiatian Zhu. Unsupervised deep learning by neighbourhood discovery. *arXiv preprint arXiv:1904.11567*, 2019. 2

[30] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color! joint end-to-end learning of global and local

image priors for automatic image colorization with simultaneous classification. *ACM Trans. Graph.*, 35(4):1–11, 2016. 2

[31] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1920–1929, 2019. 6

[32] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *Eur. Conf. Comput. Vis.*, pages 577–593. Springer, 2016. 2

[33] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6874–6883, 2017. 2

[34] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020. 6, 7

[35] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988. 2

[36] Sindy Lowe, Peter O'Connor, and Bastiaan S. Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *Adv. Neural Inform. Process. Syst.*, pages 3033–3045, 2019. 3, 5, 8

[37] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6707–6717, 2020. 2, 6, 7

[38] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Eur. Conf. Comput. Vis.*, volume 9910 of *Lecture Notes in Computer Science*, pages 69–84, 2016. 1, 2, 3, 4, 6, 7, 8

[39] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Int. Conf. Comput. Vis.*, pages 5898–5906, 2017. 2

[40] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Adv. Neural Inform. Process. Syst.*, pages 91–99, 2015. 7

[42] Enver Sangineto, Moin Nabi, Dubravko Culibrk, and Nicu Sebe. Self paced deep learning for weakly supervised object detection. *IEEE transactions on pattern analysis and machine intelligence*, 41(3):712–725, 2018. 2

[43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1–9, 2015. 3

[44] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 2

[45] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8769–8778, 2018. 8

[46] 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.*, pages 3733–3742, 2018. 7

[47] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487, 2016. 2

[48] Yuwen Xiong, Mengye Ren, and Raquel Urtasun. Loco: Local contrastive representation learning. *Adv. Neural Inform. Process. Syst.*, 33, 2020. 3, 8

[49] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Eur. Conf. Comput. Vis.*, pages 649–666. Springer, 2016. 2, 6, 7

[50] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Adv. Neural Inform. Process. Syst.*, pages 487–495, 2014. 8

[51] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Int. Conf. Comput. Vis.*, pages 6002–6012, 2019. 2, 6, 7