Self-Supervised Learning for Computer Vision

Uzay Macar
Columbia University
oum2000@columbia.edu

Guandong Liu
Columbia University
gl2675@columbia.edu

1 Introduction

Self-supervised learning exploits learning signals that come free with the data as opposed to human annotations which are expensive to collect, not readily available, and often noisy. Most of what humans learn is with self-supervision: infant learn without external supervision in the play stage and become experts on limited sets of objects and concepts, yet demonstrate remarkable out-of-distribution generalization.

Self-supervised learning has revolutionized machine learning across many modalities and tasks, most notably helping transformers achieve state-of-the-art results in natural language understanding tasks. In the new age of deep learning, we will be moving away from vision datasets with millions of images which are carefully curated for supervised classification to vision datasets with billions of images which are collected from the wild without labels and therefore more suited for self-supervised learning. Therefore, we would like the pose the following question: Can we replace self-supervised learning as the predominant pretraining routine in computer vision for finding generalizable representations that adapt well to out-of-distribution tasks?

We have developed a mini-framework that implements several self-supervised learning methods for transfer learning on image classification, and an accompanying comparative performance analysis. We have implemented the following self-supervised tasks: (i) rotation classification [1], (ii) visual counting [2], (iii) jigsaw puzzle [3], (iv) context prediction [4], and (v) colorization [5]. We then pretrained these methods on various subsets of the ImageNet-1K dataset [6]. Finally, we evaluated the performance on the the following downstream image classification datasets: (i) CIFAR-10 [7], (ii) Street View House Numbers (SVHN) [8], and (iii) Oxford FLOWERS [9].

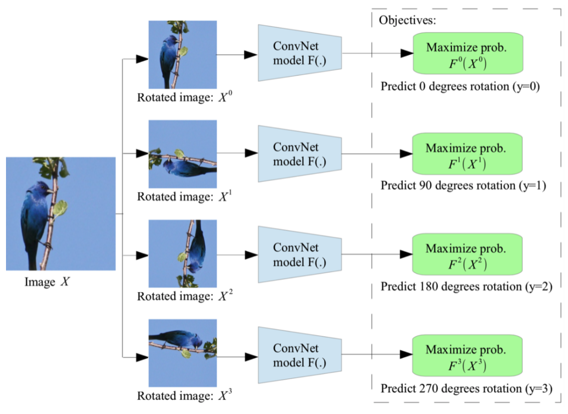
As a myriad of new self-supervised tasks and methods emerge year after year, we hope that our work will enhance reproducibility and allow customizability for adaptation to new datasets, tasks, and problems.

2 Background

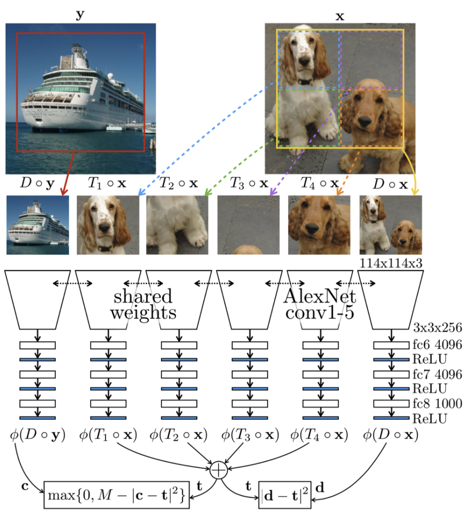
Self-supervision is a fast-emerging field that spans multiple modalities and subfields of machine learning. We can classify self-supervised pretext tasks for visual learning into the following four categories:

Generation-based Methods: Generates images or videos. Utilizes convolutional decoders, AEs, and GANs. Uses adversarial loss on top of the standard reconstruction loss (e.g. MSE).	**Context-based Methods:** Uses context features like image similarity, and spatial and temporal structure. Trained with a reconstruction loss (e.g. MSE).
Free Semantic Label-based Methods: Uses free labels from hard-coded algorithms (e.g. edge detection) and game engines.	**Cross Modal-based Methods:** Uses different channels and views of input data to create correspondence pairs.

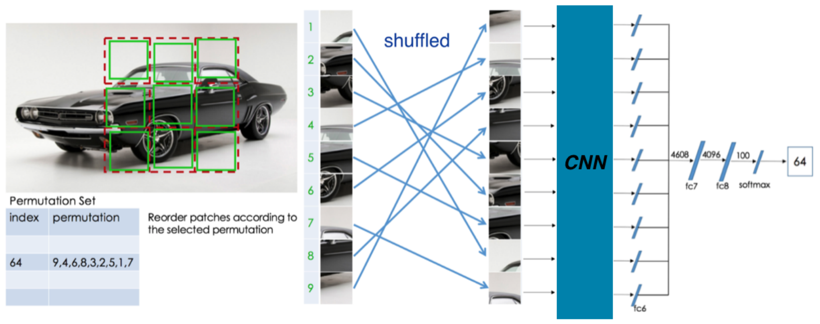
The five self-supervised methods we have implemented, as depicted in Figure 1, fall under the generation-based and context-based methods. The rest of this section covers the previous work done on each of the methods we have implemented.


Objectives:
Image X
Rotated image: X^0
ConvNet model F(.)
Maximize prob. $F^0(X^0)$
Predict 0 degrees rotation (y=0)
Rotated image: X^1
ConvNet model F(.)
Maximize prob. $F^1(X^1)$
Predict 90 degrees rotation (y=1)
Rotated image: X^2
ConvNet model F(.)
Maximize prob. $F^2(X^2)$
Predict 180 degrees rotation (y=2)
Rotated image: X^3
ConvNet model F(.)
Maximize prob. $F^3(X^3)$
Predict 270 degrees rotation (y=3)

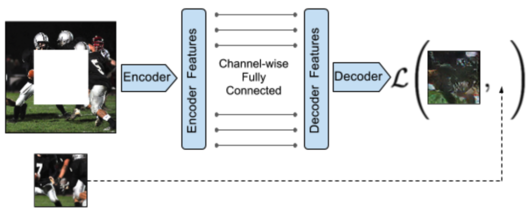
(a) Rotation Classification Task


y
x
$D \circ \mathbf{y}$
$T_1 \circ \mathbf{x}$
$T_2 \circ \mathbf{x}$
$T_3 \circ \mathbf{x}$
$T_4 \circ \mathbf{x}$
$D \circ \mathbf{x}$
114x114x3
shared weights
AlexNet conv1-5
3x3x256
fc6 4096
ReLU
fc7 4096
ReLU
fc8 1000
ReLU
$\phi(D \circ \mathbf{y})$
$\phi(T_1 \circ \mathbf{x})$
$\phi(T_2 \circ \mathbf{x})$
$\phi(T_3 \circ \mathbf{x})$
$\phi(T_4 \circ \mathbf{x})$
$\phi(D \circ \mathbf{x})$
c
$\max\{0, M - |\mathbf{c} - \mathbf{t}|^2\}$
t
t
$|\mathbf{d} - \mathbf{t}|^2$
d

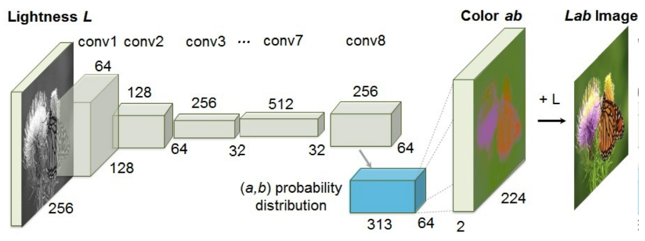
(b) Visual Counting Task



(c) Jigsaw Puzzle Task



(d) Context Prediction Task



(e) Colorization Task

Figure 1: Visualizations for the five self-supervised methods.

2.1 Rotation Classification

The input image is passed to the data collator and is rotated in 0, 90, 180, and 270 degrees, creating 4 synthetic images from one and increasing the effective batch size by a factor of 4. The rotation degrees are then taken as the ground truth pseudo-labels, and a siamese neural network is trained to maximize the probability between the predicted class and the ground truth class.

2.2 Visual Counting

The input image x is passed to the data collator and a random crop from the center is taken which fragments the image into four equal-area square tiles: T_1, T_2, T_3, and T_4. Then, a siamese neural network f with parameters θ is trained such that the summation of the output vectors for tiles must be equivalent to the output vector for the original input image x: $f_\theta(T_1) + f_\theta(T_2) + f_\theta(T_3) + f_\theta(T_4) = f_\theta(x)$. This is called the *equivariance relation*. It turns out that this problem formulation comes with the trivial solution of $f(\cdot) = 0$. We prevent converging to the trivial solution by using a contrastive loss, $M - f_\theta(y) - f_\theta(x)$, where x is the original input image, y is another image sampled randomly from the dataset, and M is a constant scalar which is set to 10 in the original paper.

2.3 Jigsaw Puzzle

9 neighboring tiles are extracted from an image. Notice that a small gap is placed between the tiles and color jitter augmentation is applied in order to prevent the model from cheating using cues of edges and continuity. The 9 tiles are shuffled and inputted to a siamese neural network, which concatenates the features of each of the tiles and passes them onto a fully-connected network with softmax. The jigsaw puzzle task is formulated as a 1000-way classification problem where the class of the given 9 tiles is the permutation index that relates the tiles back to their original location in the input image. There are a total of 9! total permutations, but the authors propose to use 1000 predefined permutations as the classes as the original number is astronomically large.

2.4 Context Prediction

A moderately sized square region is removed from the input image. The input image is then passes onto first the context encoder and then the context decoder to reconstruct the missing piece in the image. The context encoder is a convolutional neural network trained to generate the contents of an arbitrary image region conditioned on its surroundings. In order to succeed at this task, context encoders need to both understand the content of the entire image, as well as be able produce plausible hypotheses for the missing part(s).

2.5 Colorization

The input image in the RGB color space is converted to the LAB color space. Channels A and B are removed from the input image. The remaining L channel is then passed through a neural network which tries to construct the other two channels A and B. LAB color space was chosen because it matches with human perception better compared to other color spaces.

3 Objectives & Approach

Self-supervision, in the context computer vision, trains a convolutional neural network to capture semantic visual features of inputs in order to be able to successfully solve a pretext task whose pseudo-labels can be automatically generated without human supervision. We observed two main problems in the current literature, and our objectives, as discussed in the subsequent subsections, are aligned to address them:

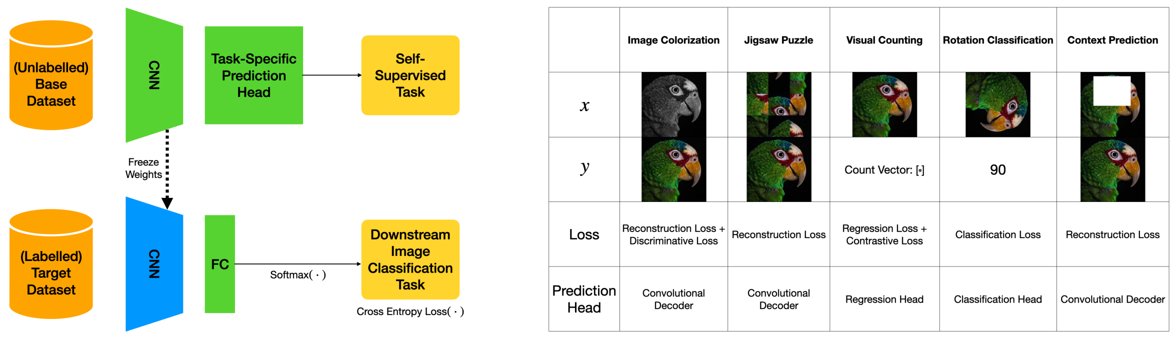
- Open-source implementations for these methods are often highly task, architecture, and dataset dependent.
- Performance comparisons for self-supervised pretraining on downstream image classification tasks have often ignored other variables such as the performance sensitivity to the number of data points (i.e. base dataset size).

3.1 Mini-Framework

The mini-framework is built on PyTorch and provides off-the-shelf implementations of several self-supervised methods, where each method is realized through a data loader with a custom data collator that returns a batch with set of inputs x, and a set of pseudo labels y in each iteration, a loss function that computes the error between predictions $\hat{y}$ and pseudo labels y, and a task-specific head (e.g. a fully-connected layer with softmax activation, a convolutional decoder, etc.) that can be attached to any convolutional neural network. For purposes of simplicity and reproducibility, our mini-framework currently uses the AlexNet architecture with batch normalization and dropout regularization all throughout. The pseudo labels y are calculated by a function evaluation $f(x)$, and the end-user is able to either select $f(\cdot)$ and the loss function based on a predefined set of choices or set them to custom ones.

We have followed a fairly standard transfer learning pipeline with the following steps as shown in Figure 2a. In the **pretraining** phrase, an encoder (e.g. a convolutional neural network) and a task-specific prediction head on top is trained to solve a pretext (i.e. self-supervised) task which involves minimizing the error between model predictions and pseduo labels. In the **finetuning** phase, the weights for the pretrained encoder is frozen and a fully-connected, linear layer with softmax activation on top is trained for a target image classification task.

The components we have implemented in our mini-framework for the chosen self-supervised methods, including inputs, outputs, loss functions, and prediction heads, are shown in 2b. Moreover, the data collators we have implemented allow for them to be used with the same self-supervised tasks in different datasets and scenarios, and follow the general scheme depicted in 3.



(a) Transfer Learning Pipeline with Self-Supervised Pretraining (b) Self-Supervised Tasks

Figure 2: The standard transfer learning pipeline we will follow (left) and a taxonomy of self-supervised tasks we have implemented (right).

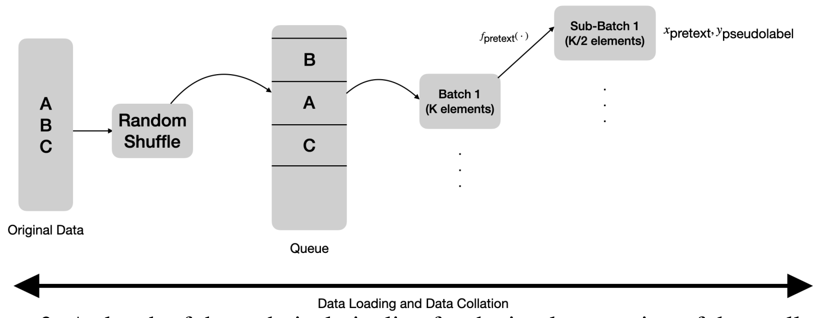

A
B
C
Original Data
Random Shuffle
B
A
C
Queue
Batch 1 (K elements)
$f_{\text{pretext}}(\cdot)$
Sub-Batch 1 (K/2 elements)
$x_{\text{pretext}}, y_{\text{pseudolabel}}$
Data Loading and Data Collation

Figure 3: A sketch of the technical pipeline for the implementation of data collators.

3.2 Implementation Details

We have performed all of the pretraining experiments using 4 RTX 2080 Ti GPUs in a data-parallel distributed training setting, and all of the finetuning experiments using Google Cloud Platform and a single NVIDIA Tesla K80.

The framework we have used is PyTorch. We have implement a custom data collator object which can be passed to the `collate_fn` function of any `torch.utils.data.DataLoader`, and which produces the desired pseudo-labels for each self-supervised task. We have used AlexNet with batch normalization and dropout throughout our experiments, both for pretraining and finetuning. For pretraining, we have used subsets of ImageNet for training and created three dataset sizes: (i) 30K, (ii) 60K, and (iii) 120K. For finetuning, we evaluated the pretraining routines on three different datasets: (i) CIFAR-10 [7], (ii) Street View House Numbers (SVHN) [8], and (iii) Oxford FLOWERS [9]. A general comparison between datasets used in this project is given below.

Dataset Name	**Number of Classes**	**Dataset Size (Train/Validation)**	**Image Size**
ImageNet	1000	30K or 60K or 120K/50K	224×224
SVHN	10	73K/26K	32×32
CIFAR-10	10	50K/10K	32×32
Flowers	102	6552/818	512×512

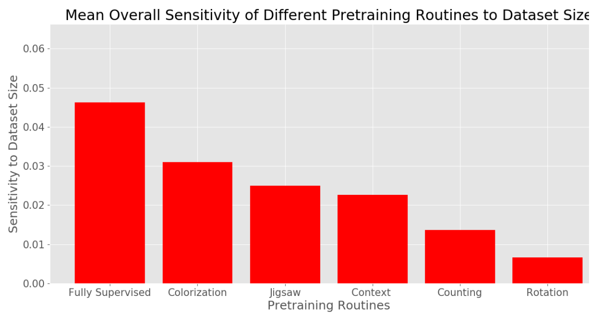
3.3 Experimental Evaluation Design

We provide a comparative performance analysis on several self-supervised learning methods. The analysis will consist of the following components: (i) Performance comparisons between random-initialization, fully-supervised pretraining, and self-supervised learning on three downstream datasets, (ii) Analysis on which pretext tasks help achieve better generalization accuracies for which datasets, and (iii) Analysis on the sensitivity of each method mentioned in (i) to the number of data points (i.e. dataset size) used for pretraining.

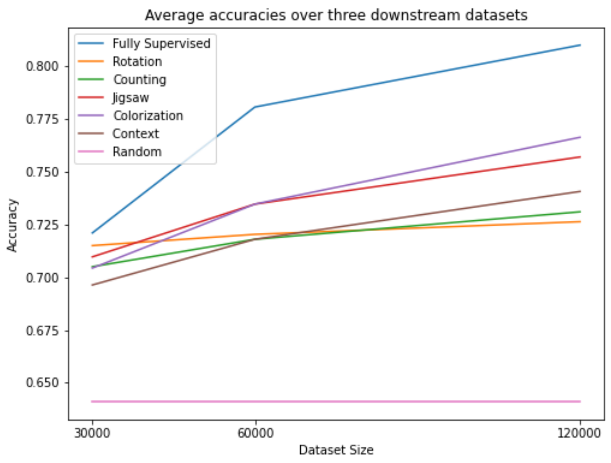
4 Results

4.1 Sensitivity

We evaluate how sensitive is each task to dataset size where sensitivity is measured as the standard deviation of validation accuracies of each task across the three ImageNet dataset sizes averaged over the three downstream datasets, as shown in **??** where (a) shows the mean overall sensitivity of different pretraining and (b) shows average accuracies of finetuning. Fully supervised learning is the most sensitive which exposes the disadvantage of supervised learning which needs enough labeled data. Among five self-supervised methods, colorization is the most sensitive and rotation is the least sensitive. The reason for that could be colorization is a difficult task that requires external world knowledge. On the other hand, distinguishing between a "standing up" image and a rotated image is easier and more intuitive to capture.



(a) Sensitivity to dataset size.



(b) Average accuracies over three downstream datasets.

4.2 Dataset Type

We then study the performance of five self-supervised methods on three downstream datasets and use pretarined AlexNet as topline and random-initialized model as baseline. For CIFAR-10 dataset shown in 5(a), in counting and rotation outperform other three methods and also close to the performance of fully supervised method, with about 2% less accuracy. Since classes in CIFAR-10 are distinct, all five methods reach much better performance than random, improving the accuracy by about 10%.

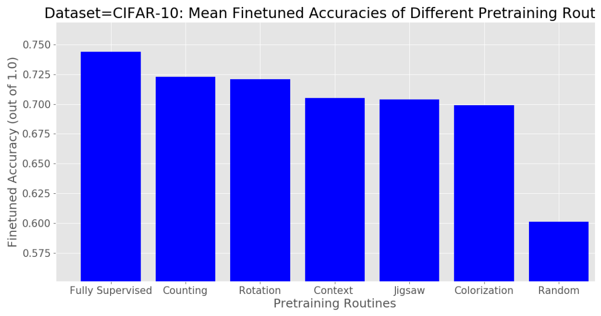
5(b) shows accuracies for SVHN dataset. Context task performs the best since it can capture latent representation of the of digits while colorization, counting, and jigsaw perform like random due to the fact that the digit need to predict are in the center and background are monochromatic in most images, which cannot gain benefits from these tasks.

Finally, in Flowers dataset shown in 5(c), colorization task performs best since images in the same class usually have almost the same color. Rotation task performs worst. Due to the shape of flowers, it is hard to distinguish between a "standing up" flower and a rotated flower here.

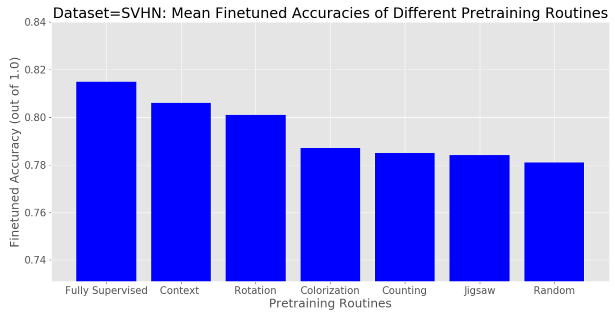
5(d) shows overall accuracies for five self-supervised methods. By averaging over all three ImageNet dataset sizes and all three datasets for downstream, colorization reaches the best performance overall. But the performance differences among five self-supervised learning methods are small. All five methods are around 72.5% on average and improves the overall accuracy about 8% compared to random-initialized model.

5 Challenges

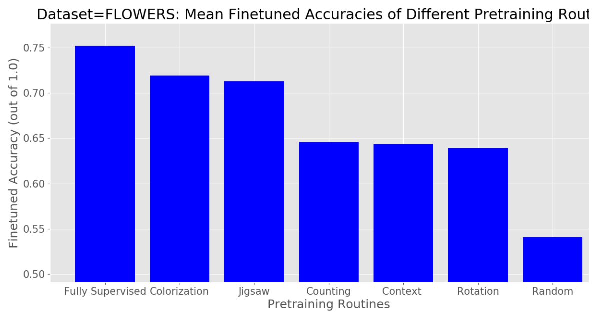
Although self-supervised tasks look easy on paper, they are hard to formulate mathematically and programmatically as you can get stuck in trivial solutions and hence poor local minima. For example, to prevent this, we had to tune the weight of the contrastive loss term in the visual counting task, which took a few days. If the weight was too small, the model would output all 0s, and too large of a weight would yield ungeneralizable representations.



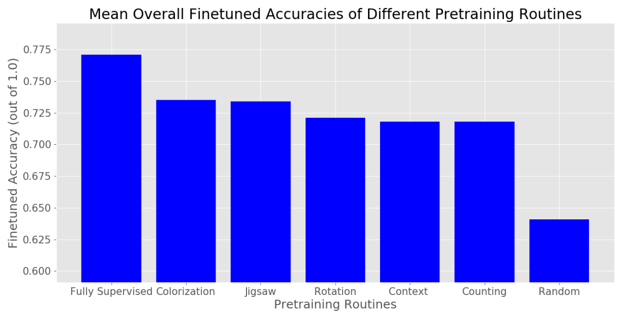
(a) CIFAR-10



(b) SVHN



(c) FLOWERS



(d) Overall

Figure 5: Finetune accuracies on three downstream datasets

We had to use large batch sizes and large mutual information in batches for satisfactory results.

Self-supervised tasks, in our observation, were generally sensitive to their initializations and configurations. For example, we had to shuffle through multiple sets of 1000-permutations for the jigsaw puzzle task to prevent divergence, and the image colorization task completely failed when we switched to the RGB color space.

We had to decide which layers of AlexNet to freeze in finetuning for fair comparison, and followed experimental insights from previous works to freeze AlexNet up to the third convolutional layer.

Due to full-capacity ImageNet-1K training taking a long time (e.g. 30 hours for the visual counting task), we decided to use a data-parallel strategy with 4 GPUs and only used subsets of the ImageNet-1K (at most 120K) for pretraining.

Our downstream datasets had different input image sizes, therefore we used an adaptive pooling layer in our network.

6 Next Steps

There are some future improvements for our work. Proposed self-supervised methods have been only compared to fully-supervised methods in single-task settings. There is an unexplored potential in multi-task self-supervised learning either through multi-objective loss functions or training on each in alternating turns in order to capture more generalizable representations. The decision of up to which layers to freeze for fine-tuning on the downstream is another thing we could investigate. We did all experiments with the third convolutional layer as suggested in previous works. But we could compare the results of choosing different convolutional layers. Our current implementations only evaluate self-supervised learning methods on image classification task. We can explore performances of self-supervised learning methods on other downstream tasks in computer vision, such as object detection and image segmentation.

7 Conclusion

We provided open-source implementations for several self-supervised learning methods for transfer learning on image classification. Compared to fully supervised learning, self-supervised learning takes advantages of generating virtually unlimited labels (pseudo-labels) by carefully designing pretext tasks. Our experiments also show there is a sensitivity-accuracy trade-off where self-supervised learning is less sensitive to dataset size but accuracy is slightly decreased. Among the five tasks, colorization is the most sensitive to dataset size but the least sensitive to dataset type while rotation is the least sensitive to dataset size but much more sensitive to dataset type. Therefore, it is crucial to use external world knowledge to design self-supervised task for different downstream datasets. We hope our work will enhance reproducibility and allow customizability for adaptation to new datasets, tasks, and problems.

References

[1] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018.

[2] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count, 2017.

[3] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles, 2017.

[4] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting, 2016.

[5] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[8] Adam Coates Alessandro Bissacco Bo Wu Andrew Y. Ng uval Netzer, Tao Wang. Reading digits in natural images with unsupervised feature learning nips workshop on deep learning and unsupervised feature learning 2011. `http://ufldl.stanford.edu/housenumbers/`.

[9] Maria-Elena Nilsback and Andrew Zisserman. Flower datasets. `http://www.robots.ox.ac.uk/~vgg/data/flowers/`.