Scaling and Benchmarking Self-Supervised Visual Representation Learning

Priya Goyal   Dhruv Mahajan   Abhinav Gupta*   Ishan Misra*
Facebook AI Research

Abstract

Self-supervised learning aims to learn representations from the data itself without explicit manual supervision. Existing efforts ignore a crucial aspect of self-supervised learning - the ability to ***scale*** *to large amount of data because self-supervision requires no manual labels. In this work, we revisit this principle and scale two popular self-supervised approaches to* ***100 million images****. We show that by scaling on various axes (including data size and problem 'hardness'), one can largely match or* ***even exceed*** *the performance of supervised pre-training on a variety of tasks such as object detection, surface normal estimation (3D) and visual navigation using reinforcement learning. Scaling these methods also provides many interesting insights into the limitations of current self-supervised techniques and evaluations. We conclude that current self-supervised methods are not 'hard' enough to take full advantage of large scale data and do not seem to learn effective high level semantic representations. We also introduce an* ***extensive benchmark*** *across* ***9 different datasets and tasks****. We believe that such a benchmark along with* ***comparable*** *evaluation settings is necessary to make meaningful progress.*

1. Introduction

Computer vision has been revolutionized by high capacity Convolutional Neural Networks (ConvNets) [42] and large-scale labeled data (e.g., ImageNet [12]). Recently [44, 67], weakly-supervised training on hundreds of millions of images and thousands of labels has achieved state-of-the-art results on various benchmarks. Interestingly, even at that scale, performance increases only log-linearly with the amount of labeled data. Thus, sadly, what has worked for computer vision in the last five years has now become a bottleneck: the size, quality, and availability of supervised data.

One alternative to overcome this bottleneck is to use the self-supervised learning paradigm. In discriminative self-supervised learning, which is the main focus of this work, a model is trained on an auxiliary or 'pretext' task for which ground-truth is available for free. In most cases, the pretext task involves predicting some hidden portion of the data (for example, predicting color for gray-scale images [13, 40, 77]). Every year, with the introduction of new pretext tasks, the performance of self-supervised methods keeps coming closer to that of ImageNet supervised pre-training. The hope around self-supervised learning outperforming supervised learning has been so strong that a researcher has even bet gelato [2].

Yet, even after multiple years, this hope remains unfulfilled. Why is that? In attempting to come up with clever pretext tasks, we have forgotten a crucial tenet of self-supervised learning: **scalability**. Since no manual labels are required, one can easily scale training from a million to billions of images. However, it is still unclear what happens when we scale up self-supervised learning beyond the ImageNet scale to 100M images or more. Do we still see performance improvements? Do we learn something insightful about self-supervision? Do we surpass the ImageNet supervised performance?

In this paper, we explore scalability which is a core tenet of self-supervised learning. Concretely, we scale two popular self-supervised approaches (`Jigsaw` [50] and `Colorization` [77]) along three axes:

1. **Scaling pre-training data:** We first scale up both methods to $100\times$ more data (YFCC-100M [68]). We observe that low capacity models like AlexNet [38] do not show much improvement with more data. This motivates our second axis of scaling.
2. **Scaling model capacity:** We scale up to a higher capacity model, specifically ResNet-50 [30], that shows much larger improvements as the data size increases. While recent approaches [16, 36, 75] used models like ResNet-50 or 101, we explore the relationship between model capacity and data size which we believe is crucial for future efforts in self-supervised learning.
3. **Scaling problem complexity:** Finally, we observe that to take full advantage of large scale data and higher capacity models, we need 'harder' pretext tasks. Specifically, we scale the 'hardness' (problem complexity) and observe that higher capacity models show a larger im-

*Equal contribution

Task	Datasets	Description
Image classification		
§ 6.1	Places205	Scene classification. 205 classes.
(Linear Classifier)	VOC07	Object classification. 20 classes.
	COCO2014	Object classification. 80 classes.
Low-shot image classification		
§ 6.2	VOC07	$\leq$ 96 samples per class
(Linear Classifier)	Places205	$\leq$ 128 samples per class
Visual navigation		
§ 6.3 (Fixed ConvNet)	Gibson	Reinforcement Learning for navigation.
Object detection		
§ 6.4	VOC07	20 classes.
(Frozen conv body)	VOC07+12	20 classes.
Scene geometry (3D)		
§ 6.5 (Frozen conv body)	NYUv2	Surface Normal Estimation.

Table 1: 9 transfer datasets and tasks used for **Benchmarking** in §6.

provement on 'harder' tasks.

Another interesting question that arises is: how does one quantify the visual representation's quality? We observe that due to the lack of a **standardized evaluation methodology** in self-supervised learning, it has become difficult to compare different approaches and measure the advancements in the area. To address this, we propose an **extensive benchmark suite** to evaluate representations using a consistent methodology. Our benchmark is based on the following principle: a good representation (1) transfers to *many* different tasks, and, (2) transfers with *limited* supervision and *limited* fine-tuning. We carefully choose 9 different tasks (Table 1) ranging from semantic classification/detection to 3D and actions (specifically, navigation). We will open-source this suite for the benefit of the community.

Our results show that by scaling along the three axes, self-supervised learning can **outperform ImageNet supervised** pre-training using the *same* evaluation setup on non-semantic tasks of Surface Normal Estimation and Navigation. For semantic classification tasks, although scaling helps outperform previous results, the gap with supervised pre-training remains significant when evaluating fixed feature representations (without full fine-tuning). Surprisingly, self-supervised approaches are quite competitive on object detection tasks with or without full fine-tuning. For example, on the VOC07 **detection** task, **without any bells and whistles, our performance matches** the supervised ImageNet pre-trained model.

2. Related Work

Visual representation learning without supervision is an old and active area of research. It has two common modeling approaches: generative and discriminative. A generative approach tries to model the data distribution directly. This can be modeled as maximizing the probability of reconstructing the input [45, 53, 70] and optionally estimating latent variables [31, 61] or using adversarial training [17, 46]. Our work focuses on discriminative learning.

One form of discriminative learning combines clustering with hand-crafted features to learn visual representations such as image-patches [15, 65], object discovery [60, 66]. We focus on discriminative approaches that learn representations directly from the the visual input. A large portion of such approaches are grouped under the term 'self-supervised' learning [11] in which the key principle is to automatically generate 'labels' from the data. The label generation can either be domain agnostic [7, 9, 54, 75] or exploit structural properties of the domain, *e.g.*, spatial structure of images [14]. We explore the 'pretext' tasks [14] that exploit structural information of the visual data to learn representations. These approaches can broadly be divided into two types - methods that use multi-modal information, *e.g.* sound [55] and methods that use only the visual data (images, videos). Multi-modal information such as depth from a sensor [19], sound in a video [4, 5, 24, 55], sensors on an autonomous vehicle [3, 32, 82] *etc.* can be used to automatically learn visual representations without human supervision. One can also use the temporal structure in a video for self-supervised methods [22, 29, 43, 48, 49]. Videos can provide information about how objects move [56], the relation between viewpoints [72, 73] *etc.*

In this work, we choose to scale image-based self-supervised methods because of their ease of implementation. Many pretext tasks have been designed for images that exploit their spatial structure [14, 50–52], color information [13, 40, 41, 77], illumination [18], rotation [25] *etc.* These pretext tasks model different properties of images and have been shown to contain complementary information [16]. Given the abundance of such approaches to use, in our work, we focus on two popular approaches that are simple to implement, intuitive, and diverse: `Jigsaw` from [50] and `Colorization` from [77]. A concurrent work [36] also explores multiple self-supervised tasks but their focus is on the architectural details which is complementary to ours.

3. Preliminaries

We briefly describe the two image based self-supervised approaches [51, 77] that we study in this work and refer the reader to the original papers for detailed explanations. Both these methods do *not* use any supervised labels.

3.1. `Jigsaw` Self-supervision

This approach by Noroozi *et al.* [50] learns an image representation by solving jigsaw puzzles created from an input image. The method divides an input image $\boldsymbol{I}$ into $N = 9$ non-overlapping square patches. A 'puzzle' is then created by shuffling these patches randomly and a ConvNet is trained to predict the permutation used to create the puzzle. Concretely, each patch is fed to a N-way Siamese ConvNet with shared parameters to obtain patch representations. The patch representations are concatenated and used to predict the permutation used to create the puzzle. In practice, as the total number of permutations $N!$ can be large, a fixed sub-

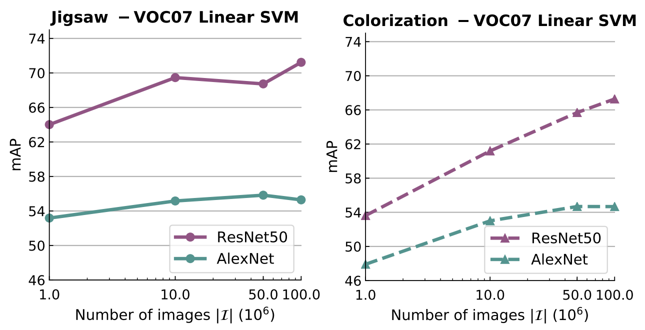


Figure 1: Scaling the Pre-training Data Size: The transfer learning performance of self-supervised methods on the VOC07 dataset for AlexNet and ResNet-50 as we vary the pre-training data size. We keep the problem complexity and data domain (different sized subsets of YFCC-100M) fixed. More details in § 4.1.

set $\mathcal{P}$ of the total $N!$ permutations is used. The prediction problem is reduced to classification into one of $|\mathcal{P}|$ classes.

3.2. `Colorization` Self-supervision

Zhang *et al.* [77] learn an image representation by predicting color values of an input 'grayscale' image. The method uses the CIE *Lab* color space representation of an input image $\boldsymbol{I}$ and trains a model to predict the *ab* colors (denoted by $\boldsymbol{Y}$) from the input lightness L (denoted by $\boldsymbol{X}$). The output *ab* space is quantized into a set of discrete bins $\mathcal{Q} = 313$ which reduces the problem to a $|\mathcal{Q}|$-way classification problem. The target *ab* image $\boldsymbol{Y}$ is soft-encoded into $|\mathcal{Q}|$ bins by looking at the K-nearest neighbor bins (default value K=10). We denote this soft-encoded target explicitly by $\boldsymbol{Z}^K$. Thus, each $|\mathcal{Q}|$-way classification problem has K non-zero values. The ConvNet is trained to predict $\boldsymbol{Z}^K$ from the input lightness image $\boldsymbol{X}$.

4. Scaling Self-supervised Learning

In this section, we scale up current self-supervised approaches and present the insights gained from doing so. We *first* scale up the data size to **100**$\times$ the size commonly used in existing self-supervised methods. However, observations from recent works [34, 44, 67] show that higher capacity models are required to take full advantage of large datasets. Therefore, we explore the *second axis* of scaling: model capacity. Additionally, self-supervised learning provides an interesting *third axis*: the complexity (hardness) of pretext tasks which can control the quality of the learned representations.

Finally, we observe the relationships between these three axes: whether the performance improvements on each of the axes are complementary or they encompass one other. To study this behavior, we introduce a simple investigation setup. Note that this setup is different from the extensive evaluation benchmark we propose in §6.

Symbol	Description
YFCC-XM	Images from the YFCC-100M [68] dataset. We use subsets of size $X \in [1M, 10M, 50M, 100M]$.
ImageNet-22k	The full ImageNet dataset (22k classes, 14M images) [12].
ImageNet-1k	ILSVRC2012 dataset (1k classes, 1.28M images) [59].

Table 2: A list of **self-supervised pre-training datasets** used in this work. We train AlexNet [38] and ResNet-50 [30] on these datasets.

Investigation Setup: We use the task of image classification on PASCAL VOC2007 [20] (denoted as VOC07). We train linear SVMs [8] (with 3-fold cross validation to select the cost parameter) on fixed feature representations obtained from the ConvNet (setup from [55]). Specifically, we choose the best performing layer: `conv4` layer for AlexNet and the output of the last `res4` block (notation from [27]) for ResNet-50. We train on the `trainval` split and report mean Average Precision (mAP) on the `test` split.

4.1. Axis 1: Scaling the Pre-training Data Size

The first premise in self-supervised learning is that it requires 'no labels' and thus can make use of large datasets. But do the current self-supervised approaches benefit from increasing the pre-training data size? We study this for both the `Jigsaw` and `Colorization` methods. Specifically, we train on various subsets (see Table 2) of the YFCC-100M dataset - YFCC-$[1, 10, 50, 100]$ million images. These subsets were collected by randomly sampling respective number of images from the YFCC-100M dataset. We specifically create these YFCC subsets so we can keep the data domain fixed. Further, during the self-supervised pre-training, we keep other factors that may influence the transfer learning performance such as the model, the problem complexity ($|\mathcal{P}| = 2000$, $K = 10$) *etc.* fixed. This way we can isolate the effect of data size on performance. We provide training details in the supplementary material.

Observations: We report the transfer learning performance on the VOC07 classification task in Figure 1. We see that increasing the size of pre-training data improves the transfer learning performance for both the `Jigsaw` and `Colorization` methods on ResNet-50 and AlexNet. We also note that the `Jigsaw` approach performs better compared to `Colorization`. Finally, we make an interesting observation that the performance of the `Jigsaw` model saturates (log-linearly) as we increase the data scale from 1M to 100M.

4.2. Axis 2: Scaling the Model Capacity

We explore the relationship between model capacity and self-supervised representation learning. Specifically, we observe this relationship in the context of the pre-training dataset size. For this, we use AlexNet and the higher capacity ResNet-50 [30] model to train on the same pre-training subsets from § 4.1.

Observations: Figure 1 shows the transfer learning per-

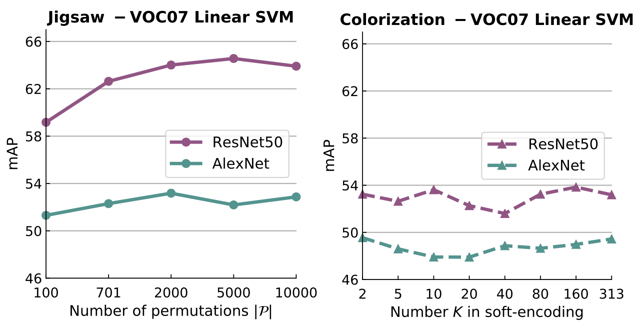


Figure 2: **Scaling Problem Complexity:** We evaluate transfer learning performance of `Jigsaw` and `Colorization` approaches on VOC07 dataset for both AlexNet and ResNet-50 as we vary the problem complexity. The pre-training data is fixed at YFCC-1M (§ 4.3) to isolate the effect of problem complexity.

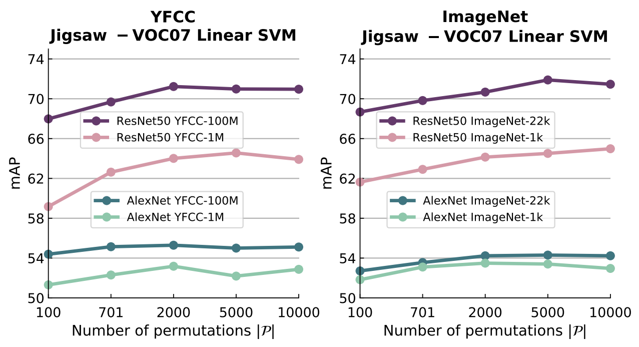


Figure 3: **Scaling Data and Problem Complexity:** We vary the pre-training data size and `Jigsaw` problem complexity for both AlexNet and ResNet-50 models. We pre-train on two datasets: ImageNet and YFCC and evaluate transfer learning performance on VOC07 dataset.

formance on the VOC07 classification task for `Jigsaw` and `Colorization` approaches. We make an important observation that the performance gap between AlexNet and ResNet-50 (as a function of the pre-training dataset size) keeps *increasing*. This suggests that higher capacity models are needed to take full advantage of the larger pre-training datasets.

4.3. Axis 3: Scaling the Problem Complexity

We now scale the problem complexity ('hardness') of the self-supervised approaches. We note that it is important to understand how the complexity of the pretext tasks affects the transfer learning performance.

Jigsaw: The number of permutations $|\mathcal{P}|$ (§ 3.1) determines the number of puzzles seen for an image. We vary the number of permutations $|\mathcal{P}| \in [100, 701, 2k, 5k, 10k]$ to control the problem complexity. Note that this is a $\mathbf{10\times}$ increase in complexity compared to [50].

Colorization: We vary the number of nearest neighbors K for the soft-encoding (§ 3.2) which controls the hardness of the colorization problem.

To isolate the effect of problem complexity, we fix the pre-training data at YFCC-1M. We explore additional ways of increasing the problem complexity in the supplementary material.

Observations: We report the results on the VOC07 classification task in Figure 2. For the `Jigsaw` approach, we see an improvement in transfer learning performance as the size of the permutation set increases. ResNet-50 shows a **5 point** mAP improvement while AlexNet shows a smaller 1.9 point improvement. The `Colorization` approach appears to be less sensitive to changes in problem complexity. We see $\sim$2 point mAP variation across different values of K. We believe one possible explanation for this is in the structure encoded in the representation by the pretext task. For `Colorization`, it is important to represent the relationship between the semantic categories and their colors, but fine-grained color distinctions do not matter as much. On the other hand, `Jigsaw` encodes more spatial structure as the problem complexity increases which may matter more for downstream transfer task performance.

4.4. Putting it together

Finally, we explore the relationship between all the three axes of scaling. We study if these axes are orthogonal and if the performance improvements on each axis are complementary. We show this for `Jigsaw` approach only as it outperforms the `Colorization` approach consistently. Further, besides using YFCC subsets for pretext task training (from § 4.1), we also report self-supervised results for ImageNet datasets (without using any labels). Figure 3 shows the transfer learning performance on VOC07 task as function of data size, model capacity and problem complexity.

We note that transfer learning performance increases on all three axes, *i.e.*, increasing problem complexity still gives performance boost on ResNet-50 even at 100M data size. Thus, we conclude that the three axes of scaling are complementary. We also make a crucial observation that the performance gains for increasing problem complexity are almost negligible for AlexNet but significantly higher for ResNet-50. This indicates that we need higher capacity models to exploit hardness of self-supervised approaches.

5. Pre-training and Transfer Domain Relation

Thus far, we have kept the pre-training dataset and the transfer dataset/task fixed at YFCC and VOC07 respectively. We now add the following pre-training and transfer dataset/task to better understand the relationship between pre-training and transfer performance.

Pre-training dataset: We use both the ImageNet [12] and YFCC datasets from Table 2. Although the ImageNet datasets [12, 59] have supervised labels, we use them (without labels) to study the effect of the pre-training domain.

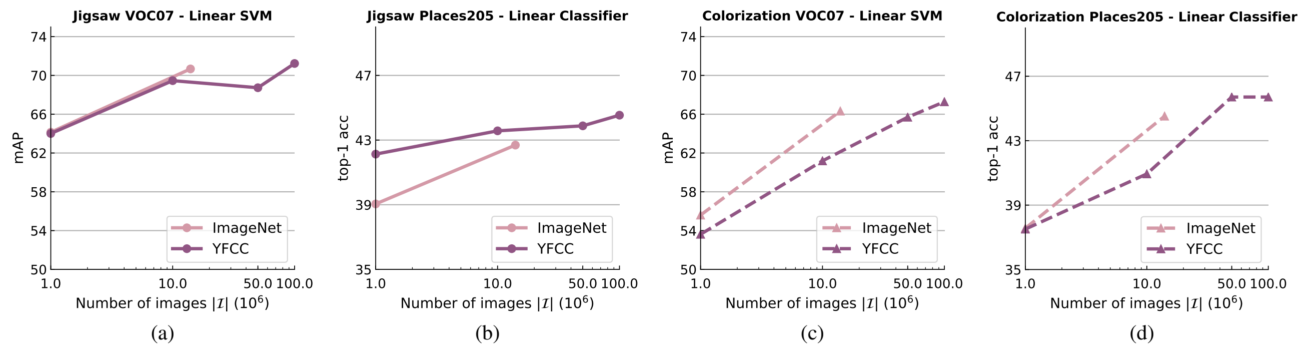


Figure 4: Relationship between pre-training and transfer domain: We vary pre-training data domain - (ImageNet-[1k, 22k], subsets of YFCC-100M) and observe transfer performance on the VOC07 and Places205 classification tasks. The similarity between the pre-training and transfer task domain shows a strong influence on transfer performance.

Transfer dataset and task: We further evaluate on the Places205 scene classification task [80]. In contrast to the object centric VOC07 dataset, Places205 is a scene centric dataset. Following the investigation setup from §4, we keep the feature representations of the ConvNets fixed. As the Places205 dataset has $>$2M images, we follow [78] and train linear classifiers using SGD. We use a batchsize of 256, learning rate of 0.01 decayed by a factor of 10 after every $40k$ iterations, and train for $140k$ iterations. Full details are provided in the supplementary material.

Observations: In Figure 4, we show the results of using different pre-training datasets and transfer datasets/tasks. Comparing Figures 4 (a) and (b), we make the following observations for the `Jigsaw` method:

- On the VOC07 classification task, pre-training on ImageNet-22k (14M images) transfers as well as pre-training on YFCC-100M (100M images).
- However, on the Places205 classification task, pre-training on YFCC-1M (1M images) transfers as well as pre-training on ImageNet-22k (14M images).

We note a similar trend for the `Colorization` problem wherein pre-training ImageNet, rather than YFCC, provides a greater benefit when transferring to VOC07 classification (also noted in [9, 14, 34]). A possible explanation for this benefit is that the domain (image distribution) of ImageNet is closer to VOC07 (both are object-centric) whereas YFCC is closer to Places205 (both are scene-centric). This motivates us to evaluate self-supervised methods on a variety of different domain/tasks and we propose an extensive evaluation suite next.

6. Benchmarking Suite for Self-supervision

We evaluate self-supervised learning on a diverse set of 9 tasks (see Table 1) ranging from semantic classification/detection, scene geometry to visual navigation. We select this benchmark based on the principle that a good representation should *generalize* to many different tasks with *limited* supervision and *limited* fine-tuning. We view self-supervised learning as a way to learn feature representations rather than an 'initialization method' [37] and thus perform limited fine-tuning of the features. We first describe each of these tasks and present our benchmarks.

Consistent Evaluation Setup: We believe that having a consistent evaluation setup, wherein hyperparameters are set fairly for all methods, is important for easier and meaningful comparisons across self-supervised methods. This is crucial to isolate the improvements due to better representations or better transfer optimization[1].

Common Setup (Pre-training, Feature Extraction and Transfer): The common transfer process for the benchmark tasks is as follows:

- First, we perform self-supervised **pre-training** using a self-supervised pretext method (`Jigsaw` or `Colorization`) on a pre-training dataset from Table 2.
- We **extract features** from various layers of the network. For AlexNet, we do this after every `conv` layer; for ResNet-50, we extract features from the last layer of every residual stage denoted, *e.g.*, `res1`, `res2` (notation from [27]) *etc*. For simplicity, we use the term `layer`.
- We then evaluate quality of these features (from different self-supervised approaches) by transfer learning, *i.e.*, benchmarking them on various **transfer** datasets and tasks that have supervision.

We summarize these benchmark tasks in Table 1 and discuss them in the subsections below. For each subsection, we provide *full details* of the training setup: model architecture, hyperparameters *etc*. in the supplementary material.

6.1. Task 1: Image Classification

We extract image features from various layers of a self-supervised network and train linear classifiers on these fixed

[1]We discovered inconsistencies across previous methods (different image crops for evaluation, weights re-scaling, pre-processing, longer fine-tuning schedules *etc*.) which affects the final performance.

Method	`layer1`	`layer2`	`layer3`	`layer4`	`layer5`
ResNet-50 ImageNet-1k Supervised	14.8	32.6	42.1	50.8	52.5
ResNet-50 Places205 Supervised	16.7	32.3	43.2	54.7	62.3
ResNet-50 Random	12.9	16.6	15.5	11.6	9.0
ResNet-50 (NPID) [75]◁	18.1	22.3	29.7	42.1	45.5
ResNet-50 `Jigsaw` ImageNet-1k	15.1	28.8	36.8	41.2	34.4
ResNet-50 `Jigsaw` ImageNet-22k	11.0	30.2	36.4	41.5	36.4
ResNet-50 `Jigsaw` YFCC-100M	11.3	28.6	**38.1**	44.8	37.4
ResNet-50 `Coloriz.` ImageNet-1k	18.1	28.4	30.2	31.3	30.4
ResNet-50 `Coloriz.` ImageNet-22k	**18.6**	**30.4**	33.2	34.8	34.1
ResNet-50 `Coloriz.` YFCC-100M	15.4	19.2	37.3	**45.8**	**41.4**

Table 3: ResNet-50 top-1 center-crop accuracy for linear classification on Places205 dataset (§ 6.1). Numbers with ◁ use a different fine-tuning procedure. All other models follow the setup from Zhang *et al*. [78].

	Places205				
Method	`layer1`	`layer2`	`layer3`	`layer4`	`layer5`
AlexNet ImageNet-1k Supervised	22.4	34.7	37.5	39.2	38.0
AlexNet Places205 Supervised	23.2	35.6	39.8	43.5	44.8
AlexNet Random	15.7	20.8	18.5	18.2	16.6
AlexNet (Jigsaw) [50]	19.7	26.7	31.9	32.7	30.9
AlexNet (Colorization) [77]	16.0	25.7	29.6	30.3	29.7
AlexNet (SplitBrain) [78]	21.3	30.7	34.0	34.1	32.5
AlexNet (Counting) [51]	23.3	33.9	36.3	34.7	29.6
AlexNet (Rotation) [25]◁	21.5	31.0	35.1	34.6	33.7
AlexNet (DeepCluster) [9]	17.1	28.8	35.2	36.0	32.2
AlexNet `Jigsaw` ImageNet-1k	23.7	33.2	36.6	36.3	31.9
AlexNet `Jigsaw` ImageNet-22k	**24.2**	34.7	37.7	37.5	31.7
AlexNet `Jigsaw` YFCC-100M	**24.2**	**34.8**	**38.2**	**38.6**	32.0
AlexNet `Coloriz.` ImageNet-1k	17.6	28.2	30.4	30.9	30.3
AlexNet `Coloriz.` ImageNet-22k	18.6	30.4	33.2	34.8	34.1
AlexNet `Coloriz.` YFCC-100M	18.4	30.1	33.4	35.0	**34.7**

Table 4: AlexNet top-1 center-crop accuracy for linear classification on Places205 dataset (§ 6.1). Numbers for [50, 77] are from [78]. Numbers with ◁ use a different fine-tuning schedule.

representations. We evaluate performance on the classification task for three datasets: Places205, VOC07 and COCO2014. We report results for ResNet-50 in the main paper; AlexNet results are in the supplementary material.

Places205: We strictly follow the training and evaluation setup from Zhang *et al*. [78] so that we can draw comparisons to existing works (and re-evaluate the model from [9]). We use a batchsize of 256, learning rate of 0.01 decayed by a factor of 10 after every $40k$ iterations, and train for $140k$ iterations using SGD on the `train` split. We report the top-1 center-crop accuracy on the `val` split for ResNet-50 in Table 3 and AlexNet in Table 4.

VOC07 and COCO2014: For smaller datasets that fit in memory, we follow [55] and train linear SVMs [8] on the frozen feature representations using LIBLINEAR package [21]. We train on `trainval` split of VOC07 dataset, and evaluate on `test` split of VOC07. Table 5 shows results on VOC07 for ResNet-50. AlexNet and COCO2014 results are provided in the supplementary material.

Observations: We see a significant accuracy gap between self-supervised and supervised methods despite our scaling efforts. This is expected as unlike self-supervised methods, both the supervised pre-training and benchmark transfer tasks solve a semantic image classification problem.

Method	`layer1`	`layer2`	`layer3`	`layer4`	`layer5`
ResNet-50 ImageNet-1k Supervised	24.5	47.8	60.5	80.4	88.0
ResNet-50 Places205 Supervised	28.2	46.9	59.1	77.3	80.8
ResNet-50 Random	9.6	8.3	8.1	8.0	7.7
ResNet-50 `Jigsaw` ImageNet-1k	**27.1**	45.7	56.6	64.5	57.2
ResNet-50 `Jigsaw` ImageNet-22k	20.2	**47.7**	57.7	**71.9**	**64.8**
ResNet-50 `Jigsaw` YFCC-100M	20.4	47.1	**58.4**	71.0	62.5
ResNet-50 `Coloriz.` ImageNet-1k	24.3	40.7	48.1	55.6	52.3
ResNet-50 `Coloriz.` ImageNet-22k	25.4	42.8	52.9	66.3	63.5
ResNet-50 `Coloriz.` YFCC-100M	26.1	42.3	53.8	67.2	61.4

Table 5: ResNet-50 Linear SVMs mAP on VOC07 classification (§ 6.1).

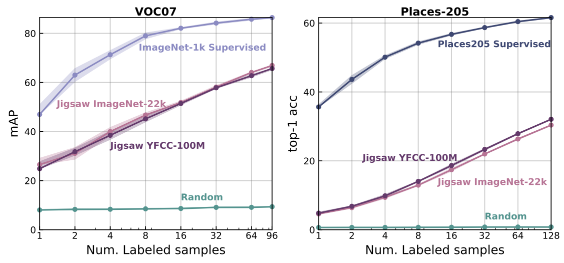


Figure 5: Low-shot Image Classification on the VOC07 and Places205 datasets using linear SVMs trained on the features from the best performing layer for ResNet-50. We vary the number of labeled examples (per class) used to train the classifier and report the performance on the `test` set. We show the mean and standard deviation across five runs (§ 6.2).

6.2. Task 2: Low-shot Image Classification

It is often argued that a good representation should not require many examples to learn about a concept. Thus, following [74], we explore the quality of feature representation when per-category examples are few (unlike § 6.1).

Setup: We vary the number k of positive examples (per class) and use the setup from § 6.1 to train linear SVMs on Places205 and VOC07 datasets. We perform this evaluation for ResNet-50 only. For each combination of k/dataset/method, we report the mean and standard deviation of 5 independent samples of the training data evaluated on a fixed test set (`test` split for VOC07 and `val` split for Places205). We show results for the `Jigsaw` method in Figure 5; `Colorization` results are in the supplementary material as we draw the same observations.

Observations: We report results for the best performing layer `res4` (notation from [27]) for ResNet-50 on VOC07 and Places205 in Figure 5. In the supplementary material, we show that for the lower layers, similar to Table 3, the self-supervised features are competitive to their supervised counterpart in low-shot setting on both the datasets. However, for both VOC07 and Places205, we observe a significant gap between supervised and self-supervised settings on their 'best' performing layer. This gap is much larger at lower sample size, *e.g*., at $k=1$ it is 30 points for Places205, whereas at higher values (full-shot in Table 3) it is 20 points.

Method	VOC07	VOC07+12
ResNet-50 ImageNet-1k Supervised*	66.7 $\pm$ 0.2	71.4 $\pm$ 0.1
ResNet-50 ImageNet-1k Supervised	**68.5** $\pm$ 0.3	**75.8** $\pm$ 0.2
ResNet-50 Places205 Supervised	65.3 $\pm$ 0.3	73.1 $\pm$ 0.3
ResNet-50 `Jigsaw` ImageNet-1k	56.6 $\pm$ 0.5	64.7 $\pm$ 0.2
ResNet-50 `Jigsaw` ImageNet-22k	**67.1** $\pm$ 0.3	**73.0** $\pm$ 0.2
ResNet-50 `Jigsaw` YFCC-100M	62.3 $\pm$ 0.2	69.7 $\pm$ 0.1

Table 6: Detection mAP for frozen `conv` body on VOC07 and VOC07+12 using Fast R-CNN with ResNet-50-C4 (mean and std computed over 5 trials). We freeze the `conv` body for all models. Numbers with * use `Detectron` [27] default training schedule. All other models use slightly longer training schedule (see § 6.4).

6.3. Task 3: Visual Navigation

In this task, an agent receives a stream of images as input and learns to navigate to a pre-defined location to get a reward. The agent is spawned at random locations and must build a contextual map in order to be successful at the task.

Setup: We use the setup from [62] who train an agent using reinforcement learning (PPO [63]) in the Gibson environment [76]. The agent uses *fixed* feature representations from a ConvNet for this task and only updates the policy network. We evaluate the representation of layers `res3`, `res4`, `res5` (notation from [27]) of a ResNet-50 by separately training agents for these settings. We use the training hyperparameters from [62], who use a rollout of size 512 and optimize using Adam [35].

Observations: Figure 6 shows the average training reward (and variance) across 5 runs. Using the `res3` layer features, we observe that our `Jigsaw` ImageNet model gives a much **higher training reward** and is more **sample efficient** (higher reward with fewer steps) than its supervised counterpart. The deeper `res4` and `res5` features perform similarly for the supervised and self-supervised networks. We also observe that self-supervised pre-training on the ImageNet domain outperforms pre-training on the YFCC domain.

6.4. Task 4: Object Detection

Setup: We use the `Detectron` [27] framework to train the Fast R-CNN [26] object detection model using Selective Search [69] object proposals on the VOC07 and VOC07+12 datasets. We provide results for Faster R-CNN [58] in the supplementary material. We note that we use the *same training schedule* for both the supervised and self-supervised methods since it impacts final object detection performance significantly. We report mean and standard deviation result of 5 independent runs for ResNet-50 only as `Detectron` does not support AlexNet.

We freeze the full `conv` body of Fast R-CNN and only train the RoI heads (last ResNet-50 stage `res5` onwards). We follow the same setup as in `Detectron` and only change the training schedule to be slightly longer. Specifically, we train on 2 GPUS for $22k/8k$ schedule on VOC07 and for $66k/14k$ schedule on VOC07+12 (compared to original $15k/5k$ schedule on VOC07 and $40k/15k$ schedule on VOC07+12). This change improves object detection performance for both supervised and self-supervised methods.

Observations: We report results in Table 6 and note that the self-supervised initialization is competitive with the ImageNet pre-trained initialization on VOC07 dataset even when fewer parameters are fine-tuned on the detection task. We also highlight that the performance gap between supervised and self-supervised initialization is very low.

6.5. Task 5: Surface Normal Estimation

Setup: We use the surface normal estimation task [23], with the evaluation, and dataset splits as formulated in [6, 47, 71]. We use the NYUv2 [64] dataset which consists of indoor scenes and use the surface normals calculated by [39]. We use the state-of-the-art PSPNet [79] architecture (implementation [81]). This provides a much stronger baseline (our scratch model outperforms the best numbers reported in [73]). We fine-tune `res5` onwards and train all the models with the same hyperparameters for 150 epochs. The scratch model (initialized randomly) is trained for 400 epochs. We use the training hyperparameters from [81], *i.e.*, batchsize of 16, learning rate of 0.02 decayed polynomially with a power of 0.9 and optimize using SGD.

Observations: We report the best `test` set performance for `Jigsaw` in Table 7 and results for `Colorization` are provided in the supplementary material. We use the metrics from [23] which measure the angular distance (error) of the prediction as well as the percentage of pixels within t° of the ground truth. We note that our `Jigsaw` YFCC-100M self-supervised model **outperforms both** the supervised models (ImageNet-1k and Places205 supervised) across all the metrics by a significant margin, *e.g.*, a **5 point** gain compared to the Places205 supervised model on the number of pixels within $t^\circ = 11.5$ metric. We, thus, conclude that self-supervised methods provide better features compared to supervised methods for 3D geometric tasks.

	Angle Distance		Within t°		
Initialization	Mean	Median	11.25	22.5	30
	(Lower is better)		(Higher is better)		
ResNet-50 ImageNet-1k supervised	26.4	17.1	36.1	59.2	68.5
ResNet-50 Places205 supervised	23.3	14.2	41.8	65.2	73.6
ResNet-50 Scratch	26.3	16.1	37.9	60.6	69.0
ResNet-50 `Jigsaw` YFCC-100M	**22.4**	**13.1**	**44.6**	**67.4**	**75.1**
ResNet-50 `Jigsaw` ImageNet-22k	22.6	13.4	43.7	66.8	74.7
ResNet-50 `Jigsaw` ImageNet-1k	24.2	14.5	41.2	64.2	72.5

Table 7: Surface Normal Estimation on the NYUv2 dataset. We train ResNet-50 from `res5` onwards and freeze the `conv` body below (§ 6.5).

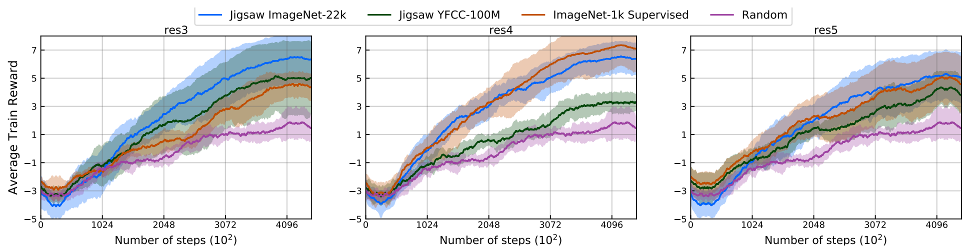


Figure 6: Visual Navigation. We train an agent on the navigation task in the Gibson environment. The agent is trained using reinforcement learning and uses fixed ConvNet features. We show results for different layers features of ResNet-50 trained on both supervised and self-supervised settings (§ 6.3).

Method	VOC07	VOC07+12
ResNet-50 ImageNet-1k Supervised*	69.1 ± 0.4	<u>76.2</u> ± 0.4
ResNet-50 ImageNet-1k Supervised	**70.5** ± 0.4	**76.2** ± 0.1
ResNet-50 Places205 Supervised	67.2 ± 0.2	74.5 ± 0.4
ResNet-50 `Jigsaw` ImageNet-1k	61.4 ± 0.2	68.3 ± 0.4
ResNet-50 `Jigsaw` ImageNet-22k	**69.2** ± 0.3	**75.4** ± 0.2
ResNet-50 `Jigsaw` YFCC-100M	66.6 ± 0.1	73.3 ± 0.4

Table 8: Detection mAP for full fine-tuning on VOC07 and VOC07+12 using Fast R-CNN with ResNet-50-C4 (mean and std computed over 5 trials) (§7). Numbers with * use `Detectron` [27] default training schedule. All other models use a slightly longer training schedule.

	ImageNet-1k				
Method	`layer1`	`layer2`	`layer3`	`layer4`	`layer5`
AlexNet ImageNet-1k Supervised	19.4	37.1	42.5	48.0	49.6
AlexNet Places205 Supervised	18.9	35.5	38.9	40.9	37.3
AlexNet Random	11.9	17.2	15.2	14.8	13.5
AlexNet (Jigsaw) [50]	16.2	23.3	30.2	31.7	29.6
AlexNet (Colorization) [77]	13.1	24.8	31.0	32.6	31.8
AlexNet (SplitBrain) [78]	17.7	29.3	35.4	35.2	32.8
AlexNet (Counting) [51]	**23.3**	**33.9**	36.3	34.7	29.6
AlexNet (Rotation) [25]$^{\triangleleft}$	18.8	31.7	38.7	38.2	36.5
AlexNet (DeepCluster) [9]	13.4	28.5	37.4	**39.2**	35.7
AlexNet `Jigsaw` ImageNet-1k	20.2	32.9	36.5	36.1	29.2
AlexNet `Jigsaw` ImageNet-22k	20.1	**33.9**	**38.7**	37.9	27.5
AlexNet `Jigsaw` YFCC-100M	20.2	<u>33.4</u>	38.1	37.4	25.8
AlexNet `Coloriz.` ImageNet-1k	14.1	27.5	30.6	32.1	31.1
AlexNet `Coloriz.` ImageNet-22k	15.0	30.5	35.5	37.9	**37.4**
AlexNet `Coloriz.` YFCC-100M	14.7	29.0	33.4	35.2	34.1

Table 9: AlexNet top-1 center-crop accuracy for linear classification on ImageNet-1k. Numbers for [50, 77] are from [78]. Numbers with $^{\triangleleft}$ use a different fine-tuning schedule.

7. Legacy Tasks and Datasets

For completeness, we also report results on the evaluation tasks used by previous works. As we explain next, we do not include these tasks in our benchmark suite (§6).

Full fine-tuning for transfer learning: This setup fine-tunes all parameters of a self-supervised network and views it as an initialization method. We argue that this view evaluates not only the quality of the representations but also the initialization and optimization method. For completeness, we report results for AlexNet and ResNet-50 on VOC07 classification in the supplementary material.

VOC07 Object Detection with Full Fine-tuning: This task fine-tunes *all* the weights of a network for the object detection task. We use the same settings as in § 6.4 and report results for supervised and `Jigsaw` self-supervised methods in Table 8. Without any bells and whistles, our self-supervised model initialization **matches the performance** of the supervised initialization on both VOC07 and VOC07+12. We note that self-supervised pre-training on ImageNet performs better than YFCC (similar to §5).

ImageNet Classification using Linear Classifiers: While the task itself is meaningful, we do not include it in our benchmark suite for two reasons:

1. For supervised representations, the widely used baseline is trained on ImageNet-1k dataset. Hence, evaluating also on the same dataset (ImageNet-1k) does *not* test generalization of the supervised baseline.
2. Most existing self-supervised approaches [14, 78] use ImageNet-1k for pre-training and evaluate the representations on the same dataset. As observed in §5, pre-training and evaluating in the *same* domain biases evaluation. Further, the bias is accentuated as we pre-train the self-supervised features and learn the linear classifiers (for transfer) on *identical* images.

To compare with existing methods, we report results on ImageNet-1k classification for AlexNet in Table 9 (setup from § 6.1). We report results on ResNet-50 in the supplementary material.

8. Conclusion

In this work, we studied the effect of scaling two self-supervised approaches along three axes: data size, model capacity and problem complexity. Our results indicate that transfer performance increases log-linearly with the data size. The quality of the representations also improves with higher capacity models and problem complexity. More interestingly, we observe that the performance improvements on the the three axes are complementary (§4). We obtain **state-of-the-art** results on linear classification using the ImageNet-1k and Places205 datasets. We also propose a benchmark suite of 9 diverse tasks to evaluate the quality of our learned representations. Our self-supervised learned representation: (a) **outperforms supervised** baseline on task of surface normal estimation; (b) performs competitively (or better) compared to supervised-ImageNet baseline on navigation task; (c) **matches the supervised object detection** baseline even with little fine-tuning; (d) performs worse than supervised counterpart on task of image classification and low-shot classification. We believe future work should focus on designing tasks that are complex enough to exploit large scale data and increased model capacity. Our experiments suggest that scaling self-supervision is crucial but there is still a long way to go before definitively surpassing supervised pre-training.

Acknowledgements: We would like to thank Richard Zhang, Mehdi Noroozi, and Andrew Owens for helping understand the experimental setup in their respective works. Rob Fergus and Léon Bottou for helpful discussions and valuable feedback. Alexander Sax, Bradley Emi, and Saurabh Gupta for helping with the Gibson experiments; Aayush Bansal and Xiaolong Wang for their help in the surface normal experiments. Ross Girshick and Piotr Dollár for helpful comments on the manuscript.

References

[1] CSAILVision Segmentation. https://github.com/CSAILVision/semantic-segmentation-pytorch. Accessed: 2019-03-20.
[2] The Gelato Bet. https://people.eecs.berkeley.edu/~efros/gelato_bet.html. Accessed: 2019-03-20.
[3] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015.
[4] R. Arandjelovic and A. Zisserman. Look, listen and learn. In *ICCV*, 2017.
[5] R. Arandjelovic and A. Zisserman. Objects that sound. In *ECCV*, 2018.
[6] A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *CVPR*, pages 5965–5974, 2016.
[7] P. Bojanowski and A. Joulin. Unsupervised learning by predicting noise. In *ICML*, 2017.
[8] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.
[9] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.
[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
[11] V. R. de Sa. Learning classification with unlabeled data. In *NIPS*, 1994.
[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.
[13] A. Deshpande, J. Rock, and D. Forsyth. Learning large-scale automatic image colorization. In *ICCV*, 2015.
[14] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, pages 1422–1430, 2015.
[15] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros. What makes paris look like paris? *ACM Transactions on Graphics*, 31(4), 2012.
[16] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. In *ICCV*, 2017.
[17] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.
[18] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *TPAMI*, 38(9):1734–1747, 2016.
[19] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.
[20] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2), 2010.
[21] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.
[22] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *CVPR*, 2017.
[23] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3d primitives for single image understanding. In *ICCV*, 2013.
[24] R. Gao, R. Feris, and K. Grauman. Learning to separate object sounds by watching unlabeled video. In *ECCV*, 2018.
[25] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.
[26] R. Girshick. Fast r-cnn. In *ICCV*, 2015.
[27] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron, 2018.
[28] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.
[29] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.
[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
[31] F. J. Huang, Y.-L. Boureau, Y. LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007.
[32] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015.
[33] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
[34] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *ECCV*, 2016.
[35] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[36] A. Kolesnikov, X. Zhai, and L. Beyer. Revisiting self-supervised visual representation learning. *arXiv preprint arXiv:1901.09005*, 2019.
[37] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
[39] L. Ladickỳ, B. Zeisl, and M. Pollefeys. Discriminatively

trained dense surface normal estimation. In *ECCV*, 2014.

[40] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016.

[41] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017.

[42] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1, 1989.

[43] P. Luc, N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun. Predicting deeper into the future of semantic segmentation. In *ICCV*, 2017.

[44] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

[45] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

[46] L. Mescheder, S. Nowozin, and A. Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *ICML*, 2017.

[47] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016.

[48] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016.

[49] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *ICML*, 2009.

[50] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[51] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. In *ICCV*, 2017.

[52] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *CVPR*, 2018.

[53] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996.

[54] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[55] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016.

[56] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *CVPR*, 2017.

[57] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun. Megdet: A large mini-batch object detector. In *CVPR*, 2018.

[58] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[59] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115, 2015.

[60] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.

[61] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455, 2009.

[62] A. Sax, B. Emi, A. R. Zamir, L. Guibas, S. Savarese, and J. Malik. Mid-level visual representations improve generalization and sample efficiency for learning active tasks. *arXiv preprint arXiv:1812.11971*, 2018.

[63] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[64] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*. Springer, 2012.

[65] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*. 2012.

[66] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *ICCV*, 2005.

[67] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[68] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

[69] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[70] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*. ACM, 2008.

[71] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015.

[72] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

[73] X. Wang, K. He, and A. Gupta. Transitive invariance for self-supervised visual representation learning. In *ICCV*, pages 1329–1338, 2017.

[74] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016.

[75] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

[76] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018.

[77] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016.

[78] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017.

[79] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[80] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014.

[81] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 2018.

[82] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

Supplementary Material

The supplementary material is organized as follows

- Section A provides architecture details for all the self-supervised networks.
- Section B provides architecture details for all the transfer tasks.
- Section C lists the hyperparameters for the self-supervised pre-training step.
- Section D lists the hyperparameters for the benchmark tasks used in Section 6 of the main paper.
- Section E lists the hyperparameters for the legacy tasks used in Section 7 of the main paper.
- Section F shows results using additional ways of increasing problem complexity for the self-supervised tasks.
- Section G shows additional results on object detection, surface normal estimation and image classification.

A. Model architectures for pretext tasks

We describe the exact architecture we use for pre-training on `Jigsaw` and `Colorization` pretext tasks below.

A.1. AlexNet Jigsaw Pretext

We describe the AlexNet architecture used for `Jigsaw` model training. We use the same architecture as [50]. Full details in Table 10.

A.2. AlexNet Colorization Pretext

We use the same architecture setup as [77] and recommend the reader to consult their implementation. Every `conv` layer is followed by `SpatialBN` + `Relu` combination. Full details in Table 11.

A.3. AlexNet Supervised

We follow the CaffeNet BVLC exact architecture and directly use the pre-trained model weights. We refer reader to [33] for exact architecture details. We did not train our AlexNet supervised model to avoid introducing any differences in results.

A.4. ResNet-50 Jigsaw Pretext

The ResNet-50 architecture used to train `Jigsaw` model is described below. The jigsaw model is trained using N-way Siamese ConvNet with shared weights. We describe the one siamese branch only in Table 12. Also note that, after the N-way siamese branches are concatenated, we have single branch left.

A.5. ResNet-50 Colorization Pretext

The ResNet-50 architecture used to train `Colorization` model is described in Table 13. We closely follow the architecture as in A.4.

A.6. ResNet-50 Supervised

We strictly follow the same ResNet architecture as in [28] and refer the reader to the work.

B. Model architectures for Transfer tasks

In this section, we describe the exact model architecture we use for various evaluation tasks (including benchmark suite as described in Section 6 of the main paper).

B.1. AlexNet Colorization Transfer

We use the same architecture setup as [77] and recommend the reader to consult their implementation. Every `conv` layer is followed by `SpatialBN` + `Relu` combination. For evaluation, we downsample `conv` layers so that the resulting feature map has dimension $9k$. Full details in Table 14.

B.2. AlexNet Jigsaw Transfer

For evaluation, we downsample `conv` layers by applying an `avgpool` layer so that the resulting feature map has dimension $9k$. Full details in Table 15.

B.3. AlexNet Supervised Transfer

We follow the CaffeNet BVLC exact architecture and directly use the pre-trained model weights. We refer reader to [33] for exact architecture details.

B.4. ResNet-50 Jigsaw Transfer

Table 16 shows the exact architecture used.

B.5. Colorization ResNet-50 Transfer

Table 17 shows the exact architecture used.

B.6. ResNet-50 Supervised Transfer

We strictly follow the same ResNet architecture as in [28] and refer the reader to the work.

C. Pre-training Hyperparameters

In this section, we describe the pre-training hyperparameters used to pre-train self-supervised methods (`Jigsaw` and `Colorization`) for both AlexNet and ResNet-50 models.

C.1. AlexNet Jigsaw

For training AlexNet on `Jigsaw`, we follow the hyperparameters setting from [50]. For the jigsaw problem, we read the original image from the data source, scale the shorter side to 256 and randomly crop out 255×255 image. We make the images grayscale randomly with 50% probability and we apply color projection with 50% probability (if the image is still colored). We further divide the image into 3x3 grid with each cell of size 85x85. Next, we randomly crop out a patch of size 64x64 from the 85x85 cell. This patch becomes a piece in jigsaw puzzle. Further, following [50], we apply bias decay for the bias parameter of the model and we also do not apply weight decay to the `scale` and `bias` parameter of `SpatialBN` layers. We train the model on 8-gpus, use minibatch size of 256, initial learning rate (LR) of 0.01 with the learning rate dropped by factor of 10 at certain interval. We use momentum of 0.9, weight decay 5e-4 and `SpatialBN` weight decay 0. We use nesterov SGD for optimization. The number of training iterations depends on the dataset size we are training on. We describe that next for each different dataset used and also corresponding to the *best* models reported in the main paper.
Model training iterations for Scaling Data size analysis (Section 4.1 of main paper):

- ImageNet-1k permutation 2k: Train for 70 epochs with LR schedule: $100k/100k/100k/50k$.
- ImageNet-22k permutation 2k: Train for 100 epochs with LR schedule: $1584343/1584343/1584343/792171$.
- YFCC-1M permutation 2k: Train for 70 epochs with LR schedule: $100k/100k/100k/50k$.
- YFCC-10M permutation 2k: Train for 70 epochs with LR schedule: $781250/781250/781250/390625$.
- YFCC-50M permutation 2k: Train for 10 epochs only with LR schedule: $558036/558036/558036/279017$.
- YFCC-100M permutation 2k: Train for 25 epochs with LR schedule: $2790178/2790178/2790178/1395089$.

For the best models (Section 6 of main paper), the training schedule is as follows:

- ImageNet-1k permutation 2k: Train for 70 epochs with LR schedule: $100k/100k/100k/50k$.
- ImageNet-22k permutation 2k: Train for 100 epochs with LR schedule: $1584343/1584343/1584343/792171$.
- YFCC-100M permutation 2k: Train for 25 epochs with LR schedule: $2790178/2790178/2790178/1395089$.

C.2. ResNet-50 Jigsaw

For training ResNet-50 on `Jigsaw`, we closely follow the hyperparameters setting from [50]. Specifically, we read the original image from the data source, scale the shorter side to 256 and randomly crop out 255×255 image. We make the images grayscale randomly with 50% probability and we apply color projection with 50% probability (if the image is still colored). We further divide the image into 3x3 grid with each cell of size 85x85. Next, we randomly crop out a patch of size 64x64 from the 85x85 cell. This patch becomes a piece in jigsaw puzzle. Further, following [50], we apply bias decay for the bias parameter of the model. We train the model on 8-gpus, use minibatch size of 256, initial learning rate (LR) of 0.1 with the learning rate dropped by factor of 10 after certain steps. We use momentum of 0.9, weight decay 1e-4 and `SpatialBN` weight decay 0. We use Nesterov SGD for optimization. The number of training iterations depends on the dataset size we are training on. We describe that next for each different dataset used and also corresponding to the *best* models reported in the main paper. We report the total number of epochs and the steps at which the learning rate is decayed.

Model training iterations for Scaling Data size analysis (Section 4.1 of main paper):

- ImageNet-1k permutation 2k: Train for 90 epochs with LR schedule: $150150/150150/100100/50050$.
- ImageNet-22k permutation 2k: Train for 90 epochs with LR schedule: $1663874/1663874/1109249/554673$.
- YFCC-1M permutation 2k: Train for 90 epochs with LR schedule: $150150/150150/100100/50050$.
- YFCC-10M permutation 2k: Train for 90 epochs with LR schedule: $1171875/1171875/781250/390625$.
- YFCC-50M permutation 2k: Train for 10 epochs only with LR schedule: $651042/651042/434028/217013$.
- YFCC-100M permutation 2k: Train for 10 epochs only with LR schedule: $1302083/1302083/868055/434027$.

The training schedule for the best models (Section 6 of main paper) is as follows:

- ImageNet-1k permutation 5k: Train for 90 epochs with LR schedule: $150150/150150/100100/50050$.
- ImageNet-22k permutation 5k: Train for 90 epochs with LR schedule: $1663874/1663874/1109249/554673$.
- YFCC-100M permutation 10k: Train for 10 epochs with LR schedule: $1302083/1302083/868055/434027$.

C.3. AlexNet Colorization

We closely follow the implementation from [77] and use the 313 bins and priors provided for training the models. Specifically, we read the original image from the data source, convert the image to Lab, scale the shorter side to 256, randomly crop out 180×180 image and randomly flip the image horizontally. Further, following [77], we apply no bias decay for the bias parameter of the model and we also do not apply weight decay to the `scale` and `bias` parameter of `SpatialBN` layers. We train the model on 8-gpus, use a minibatch size of 640, initial learning rate (LR) of 24e-5 with the learning rate dropped by 0.34 at certain interval. We use weight decay 1e-3 and `SpatialBN` weight decay 0. We use Adam for optimization and beta1 0.9, beta2 0.999 and epsilon 1e-8. The number of training iterations depend on the dataset size we are training on. We describe that next for each different dataset used and also corresponding to the *best* models reported in the main paper.

Model training iterations used for `Colorization` models in the main paper:

- ImageNet-1k: Train for 28 epochs with LR schedule: $30027/8008/12011/6205$.
- ImageNet-22k: Train for 112 epochs with LR schedule: $1341749, 347861, 521791, 273319$.
- YFCC-1M: Train for 28 epochs with LR schedule: $30027/8008/12011/6205$.
- YFCC-10M: Train for 56 epochs with LR schedule: $468750/125000/187500/93750$.
- YFCC-50M: Train for 10 epochs only with LR schedule: $1046317/279018/418527/209263$.
- YFCC-100M: Train for 15 epochs only with LR schedule: $1265625/328125/492188/257812$.

C.4. ResNet-50 Colorization

We closely follow the same setup as for AlexNet described above. We use the 313 bins and priors provided for training the models from [77]. We read the original image from the data source, convert the image to Lab, scale the shorter side to 256, randomly crop out $180x180$ image and randomly flip the image horizontally. Further, we apply no bias decay for the bias parameter of the model and we also do not apply weight decay to the `scale` and `bias` parameter of `SpatialBN` layers. We train the model on 8-gpus, use a minibatch size of 640, initial learning rate (LR) of 24e-5 with the learning rate dropped by 0.34 at certain interval. We use weight decay 1e-3 and `SpatialBN` weight decay 0. We use Adam for optimization and beta1 0.9, beta2 0.999 and epsilon 1e-8. The number of training iterations depends on the dataset size we are training on. We describe that next for each different dataset used and also corresponding to the *best* models reported in the main paper.

Model training iterations used for ResNet-50 `Colorization` models in the main paper:

- ImageNet-1k: Train for 28 epochs with LR schedule: $30027/8008/12011/6205$.
- ImageNet-22k: Train for 84 epochs with LR schedule: $1006312/260896/391343/204989$.
- YFCC-1M: Train for 28 epochs with LR schedule: $30027/8008/12011/6205$
- YFCC-10M: Train for 56 epochs with LR schedule: $468750/125000/187500/93750$.
- YFCC-50M: Train for 30 epochs only with LR schedule: $3138951/837054/1255581/627789$.
- YFCC-100M: Train for 15 epochs only with LR schedule: $1265625/328125/492188/257812$.

D. Hyperparameters used in Benchmark Tasks

In this section, we describe hyperparameter settings for various benchmark tasks described in the main paper.

D.1. Image Classification

VOC07 and COCO2014: We train Linear SVMs on frozen feature representations using LIBLINEAR package [21]. We train a linear SVM per class (20 for VOC07 and 80 for COCO2014) for the cost values $C \in 2^{[-19,-4]} \cup 10^{[-7,2]}$ (*i.e.* 26 C values). We use 3-fold cross-validation to choose the cost parameter per class and then further calculate the mean average precision. To train SVM, we first normalize the features of shape (N, 9k) (where N is number of samples in data and 9k is the resized feature dimension) to have norm=1 along feature dimension. We apply the same normalization step on evaluation data as well. We use the following hyperparameter setting for training using `LinearSVC` sklearn class. We use `class_weight` ratio as 2:1 for positive/negative classes, `penalty`=l2, `loss`=squared_hinge, `tol`=1e-4, `dual`=True and `max_iter`=2000.

Places205: We train linear classifiers on frozen feature representations using Nesterov SGD (in D.6, we discuss the reason for this choice). We freeze the feature representations of various self-supervised networks, resize the features to have dimension 9k and then train linear classifiers. We describe the hyperparameters used for AlexNet and ResNet-50 on both `Jigsaw` and `Colorization` approaches.

1. AlexNet `Colorization`: We strictly follow [77]. Specifically, we train on 8-gpus, use minibatch size of 256, initial learning rate (LR) of 0.01 with the learning rate dropped by factor of 10 at certain interval. We use momentum of 0.9, weight decay 5e-4 and `SpatialBN` weight decay 0. We do not apply bias decay for the bias parameter of the model and we also do not apply weight decay to the `scale` and `bias` parameter of `SpatialBN` layers. We train for 140k iterations total and use the learning rate schedule of $40k/40k/40k/20$. We read the input image, convert it to Lab, resize the shorter side to 256, randomly crop 227x227 image and apply horizontal flip with 50% probability.

2. AlexNet `Jigsaw`: We follow the settings above and train on 8-gpus, use minibatch size of 256, initial learning rate (LR) of 0.01 with the learning rate dropped by factor of 10 at certain interval. We use momentum of 0.9, weight decay 5e-4 and `SpatialBN` weight decay 0. We apply bias decay for the bias parameter of the model and we do not apply weight decay to the `scale` and `bias` parameter of `SpatialBN` layers. We train for 140k iterations total and use the learning rate schedule of $40k/40k/40k/20$. We read the input image, convert it to Lab space, resize the shorter side to 256, randomly crop 227x227 image and apply horizontal flip with 50% probability.

3. ResNet-50 `Colorization`: We closely follow the hyperparameter setting above and train on 8-gpus, use minibatch size of 256, initial learning rate (LR) of 0.01 with the learning rate dropped by factor of 10 at certain interval. We use momentum of 0.9, weight decay 1e-4 and `SpatialBN` weight decay 0. We do not apply bias decay for the bias parameter of the model. We train for 140k iterations total and use the learning rate schedule of $40k/40k/40k/20k$. We read the input image, convert it to Lab, resize the shorter side to 256, randomly crop 224x224 image and apply horizontal flip with 50% probability.

4. ResNet-50 `Jigsaw`: We closely follow the hyperparameter setting above and train on 8-gpus, use minibatch size of 256, initial learning rate (LR) of 0.01 with the learning rate dropped by factor of 10 at certain interval. We use momentum of 0.9, weight decay 1e-4 and `SpatialBN` weight decay 0. We apply bias decay for the bias parameter of the model and we do not apply weight decay to the `scale` and `bias` parameter of `SpatialBN` layers. We train for 140k iterations total and use the learning rate schedule of $40k/40k/40k/20k$. We read the input image, convert it to Lab, resize the shorter side to 256, randomly crop 224x224 image and apply horizontal flip with 50% probability.

D.2. Low-shot Image Classification

We train Linear SVMs on VOC07 and Places205 dataset using the exact same setup as in D.1. The data sampling technique for various low-shot settings are described in the main paper.

D.3. Object Detection

We follow the same settings as [27]. We train on 2-gpus with initial learning rate of 2e-3. For Fast R-CNN, we fine-tune for $22k/8k$ on VOC07 and for $66k/14k$ on VOC07+12. For Faster R-CNN, we fine-tune for $38k/12k$ on VOC07 and for $65k/35k$ on VOC07+12. All other hyperparameters are defaults set in `Detectron`. Note that `Detectron` default settings use single scale inference with scale value 600.

D.4. Surface Normal Estimation

We use the NYUv2 dataset with the surface normals computed by [39]. We use the evaluation metrics from [23] and the problem formulation from [71].

Problem Setup: Following [71], we reduce the problem of surface normal estimation to a classification task. We construct a codebook of size 40 by clustering the surface normals in the `train` split of NYUv2. We then quantize all the surface normals using the codebook and pick the index of the nearest cluster center. This reduces the problem to a 40-way classification problem which we optimize using a multinomial cross-entropy loss. At test time, we predict the distribution over the 40 classes at each pixel location. We convert these per-class distributions into a continuous surface normal prediction by a weighted averaging of the codebook centers with the per-class distribution predictions.

Architecture: We use the PSPNet [79] implementation from [1]. Specifically we use the ResNet50-dilated backbone encoder and the C1 decoder from their implementation. We only train from `res5` onwards. We use a learning rate of 2e-2 with a polynomial decay schedule using a power of 0.9 and a batchsize of 16 across 8 GPUs with Synchronized BatchNorm [57]. We train all models for 150 epochs and report the best test set performance. The scratch model is trained for 400 epochs and all parameters are updated only for this model. We resize the image with a minimum side of $[300, 375, 450, 525, 600]$ pixels for data augmentation during training (and no left-right flipping).

D.5. Visual Navigation

Image Features: We use the implementation and the optimization parameters from [62]. We modify their implementation to use the self-supervised and the supervised ResNet-50 ConvNets to extract features from the images. As their implementation uses an 8 channel feature, we use a random projection of the features from a ResNet-50. For example, we use a random projection to take the 2048 channel `res5` features to a 8 channel features. We do *not* train the ConvNet or the random projection matrix.

Agent network architecture: The Agent uses a recurrent network (GRU [10]) with a state size of 512 dimensions.

Optimization: We use the ADAM [35] optimizer with a fixed learning rate of $1e-4$, clipping the l_2 norm of the gradient at 0.5, a rollout size of 512. We use the PPO algorithm [63] with a replay buffer size of 10000, value loss weight $1e$–3, entropy co-efficient $1e$–4, and a clipping value of 0.1 for the trust region.

D.6. Note on Using SGD based Linear Classifiers vs. DCD

Although, using SGD based Linear Classifiers is a common practice [78] to evaluate representations, we found that optimization hyperparameters can lead to signficantly different results. The SGD based classifiers solve a convex optimization problem, but as also noted in [36], they can demonstrate a *very* slow convergence. Thus, fine-tuning for larger number of iterations or using a different learning rate decay method (at fixed number of fine-tuning iterations) *etc.* can give significantly different results. We obtained more robust results using Dual Coordinate Descent (DCD) as implemented in the LIBLINEAR package [21]. Although, this changes the classifier from a logistic regressor to a linear SVM, we believe this setting provides an easy, robust, and fair comparison of image representations and use this setting for the smaller VOC07 and COCO2014 datasets.

E. Hyperparameters used in Legacy Tasks

In this section, we describe hyperparameter settings for various legacy tasks described in Section 7 main paper.

E.1. ImageNet classification using Linear Classifiers

We use the exact same setting as described in Section D.1 for Places205 dataset.

E.2. VOC07 full fine-tuning

We use the self-supervised weights to initialize the network and fine-tune the full network on VOC07 classification task. We use Nesterov SGD for optimization. We describe hyperparameters used in fine-tuning next.

1. AlexNet `Jigsaw`: We strictly follow [77]. Specifically, we train for 80k iterations on 1-gpu using minibatch size of 16, initial learning rate (LR) of 0.001 with the learning rate dropped by factor of 10 after 10K iterations. We use momentum of 0.9, weight decay 1e-6 and `SpatialBN` weight decay 1e-4. We do not apply bias decay for the bias parameter of the model and we also do not apply weight decay to the `scale` and `bias`

parameter of `SpatialBN` layers. We read the input image, randomly crop 227x227 image and apply horizontal flip with 50% probability.

2. AlexNet `Colorization`: We follow settings above and train for 80k iterations on 1-gpu using minibatch size of 16, initial learning rate (LR) of 0.005 with the learning rate dropped by factor of 10 after 10K iterations. We use momentum of 0.9, weight decay 1e-6 and `SpatialBN` weight decay 0. We do not apply bias decay for the bias parameter of the model. We read the input image, convert it to Lab, randomly crop 227x227 image and apply horizontal flip with 50% probability.

3. ResNet-50 `Jigsaw`: We train for 3000 iterations on 4-gpus using minibatch size of 128, initial learning rate (LR) of 0.01 with the learning rate dropped by factor of 10 after 1600 iterations. We use momentum of 0.9, weight decay 1e-6 and `SpatialBN` weight decay 0. We do not apply bias decay for the bias parameter of the model and we also do not apply weight decay to the `scale` and `bias` parameter of `SpatialBN` layers. We read the input image, randomly crop 224x224 image and apply horizontal flip with 50% probability.

4. ResNet-50 `Colorization`: We train for 3000 iterations on 4-gpus using minibatch size of 128, initial learning rate (LR) of 0.15 with the learning rate dropped by factor of 10 after 1600 iterations. We use momentum of 0.9, weight decay 1e-6 and `SpatialBN` weight decay 0. We do not apply bias decay for the bias parameter of the model. We read the input image, convert it to Lab, randomly crop 224x224 image and apply horizontal flip with 50% probability.

F. Alternative ways of scaling problem complexity

In the main paper (Section 4.3), we showed how to increase the problem complexity for the `Jigsaw` task by increasing the size of the permutation set $|\mathcal{P}|$, and for the `Colorization` task by changing the number of nearest neighbors K for the soft-encoding. We explore additional ways to increase the problem complexity for the `Jigsaw` and `Colorization` methods.

F.1. Jigsaw: Number of patches N

We increase the number of patches N from 9 (default in [50]) to 16. We use $|\mathcal{P}| = 2000$. The input image is resized to 300×300 which is divided into a 4×4 tiles of size 75×75. A 64×64 patch is then extracted from each tile randomly to get 16 patches total. We use the same investigation setup as in Section 4 of the main paper - train linear SVMs on the fixed representations for the VOC07 image classification task. Our results, shown in Table 25, indicate that increasing the number of patches does not result in a higher quality representation. Thus, we only performed further experiments in increasing problem complexity by varying the size of the permutation set $|\mathcal{P}|$.

F.2. Colorization: Number of color bins $|\mathcal{Q}|$

We increase the size of the color bins $|\mathcal{Q}|$ that are used to quantize the color space (see Section 3.2 of the main paper). This increases the number of colors the ConvNet predicts for the `Colorization` problem. We evaluate the quality of the features by transfer learning on the Places205 dataset (same setup as in Section 5 of the main paper). As Table 26 shows, using $|\mathcal{Q}| \in [313, 262]$ gives the best results. Thus, we use 313 bins in the main paper which is also the default in [77].

We also experimented with the bandwidth of the Gaussian used to compute the soft-encoding $\boldsymbol{Z}^K$ but did not see any significant improvements.

G. Additional Results

G.1. Image Classification using Linear Classifiers on ImageNet

Similar to Section 7 of the main paper, we train linear classifiers on frozen representations from different layers of a ResNet-50 model in Table 18. We strictly follow the setup from Zhang *et al*. [78] and compare against earlier works that also use ResNet-50 for self-supervised pre-training.

G.2. Image Classification using Linear SVM on COCO2014

Complete results of **Section 6.1** in the main paper. We report results on the COCO2014 dataset in Table 19.

G.3. Image Classification using Full fine-tuning on VOC07

We perform full fine-tuning (hyperparameters in appendix E.2) of the self-supervised networks for the VOC07 classification task. We provide results for the ResNet-50 model in Table 20 and for AlexNet in Table 21. The ResNet-50 model matches the performance of a supervised pre-trained Places205 model on the VOC07 classification task. We note that obtaining a comparable evaluation setting to prior work using AlexNet is difficult because of differences in fine-tuning schedule and weight re-scaling methods.

G.4. Layerwise Results for Low-shot

We report results on low-shot classification (Section 6.2 in the main paper) in Figure 7. We show the results for both the `Colorization` and `Jigsaw` self-supervised methods for a ResNet-50 model.

G.5. Surface Normal Estimation using a ResNet-50 Colorization model

We report results on the Surface Normal Estimation task using a ResNet-50 self-supervised model on the `Colorization` method (setup from Section 6.4 of the paper). These results are in Table 22.

G.6. Faster R-CNN results on VOC07 and VOC07+12

Similar to Section 6.3 of the main paper, we freeze the `conv` body for all the models. We train the ROI-heads and the classifier (`res5` onwards). We report these results in Table 23. For all the methods (including supervised and self-supervised), we use a slightly longer fine-tuning schedule of $38k/12k$ for VOC07 and $65k/35k$ for VOC07+12. All other parameters are kept the same as in `Detectron` [27].

Full fine-tuning: We evaluate in the *full* fine-tuning setting to draw comparisons with [75]. We use the default parameters in `Detectron` [27] for this setting. Our ResNet-50 `Jigsaw` ImageNet-22k model fine-tuned on VOC07 `trainval` obtains 68.9 mAP on VOC07 `test` compared to 65.4 reported in [75]. We show full results on this setting in Table 24.

Layer	X	C	K	S	P	G	D
data	64	27	–	–	–	–	–
data_split	64	3	–	–	–	–	–
conv1	27	96	11	2	0	1	1
pool1	13	96	3	2	0	–	1
conv2	13	256	5	1	2	2	1
pool2	6	256	3	2	0	–	1
conv3	6	384	3	1	1	1	1
conv4	6	384	3	1	1	2	1
conv5	6	256	3	1	1	2	1
pool5	2	256	3	2	0	–	1
fc6	1	1024	1	1	0	–	2
concat	1	9216	1	1	0	–	1
fc7	1	4096	1	1	0	–	1
fc8	1	*	1	1	0	–	1

Table 10: AlexNet architecture used for Jigsaw pretext task. **X** spatial resolution of layer, **C** number of channels in layer; **K** conv or pool kernel size; **S** computation stride; **D** kernel dilation; **P** padding; **G** group convolution, last layer is removed during transfer evaluation. Number with * depends on the size per permutation set used to train jigsaw puzzle.

Layer	X	C	K	S	P	G	D
data	180	1	–	–	–	–	–
conv1	45	96	11	4	5	1	1
pool1	23	96	3	2	1	–	1
conv2	23	256	5	1	2	2	1
pool2	12	256	3	2	1	–	1
conv3	12	384	3	1	1	1	1
conv4	12	384	3	1	1	2	1
conv5	12	256	3	1	1	2	1
pool5	12	256	3	1	1	–	1
fc6	12	4096	1	1	0	–	2
fc7	12	4096	1	1	0	–	1
fc8	12	*	1	1	0	–	1

Table 11: AlexNet architecture used for Colorization pretext task. **X** spatial resolution of layer, **C** number of channels in layer; **K** conv or pool kernel size; **S** computation stride; **D** kernel dilation; **P** padding; **G** group convolution, last layer is removed during transfer evaluation. Number with * depends on the colorization bin size.

Layer	X	C	K	S	P	G
data	64	27	–	–	–	–
data_split	64	3	–	–	–	–
conv1	32	64	7	2	3	1
maxpool	16	64	3	2	1	–
res2	16	256	*	*	*	1
res3	8	512	*	*	*	1
res4	4	1024	*	*	*	1
res5	2	2048	*	*	*	1
avgpool	1	2048	2	1	0	–
fc1†	1	1024	1	1	0	1
concat	1	9216	–	–	–	–
fc2	1	$\triangleleft$	1	1	0	1

Table 12: ResNet-50 architecture used for Jigsaw pretext task. **X** spatial resolution of layer, **C** number of channels in layer; **K** conv or pool kernel size; **S** computation stride; **D** kernel dilation; **P** padding; **G** group convolution. Layers denoted with res prefix represent the bottleneck residual block. Number with * use the original setting as in [30]. Layer with † is implemented as a conv layer. Number with $^\triangleleft$ depend on the size of permutation set used for training Jigsaw model (see Section 4.3 in main paper)

.

Layer	X	C	K	S	P	G
data	180	1	–	–	–	–
conv1	90	64	7	2	3	1
maxpool	45	64	3	2	1	–
res2	45	256	*	*	*	1
res3	23	512	*	*	*	1
res4	12	1024	*	*	*	1
res5	12	2048	*	*	*	1
avgpool	12	2048	3	1	1	–
fc1†	12	313	6	1	5	1

Table 13: ResNet-50 architecture used for `Colorization` pretext task. **X** spatial resolution of layer, **C** number of channels in layer; **K** conv or pool kernel size; **S** computation stride; **D** kernel dilation; **P** padding; **G** group convolution. Layers denoted with `res` prefix represent the bottleneck residual block. Number with * use the original setting as in [30]. Layer with † is implemented as a `conv` layer.

Layer	X	C	K	S	P	G	D	X_d	K_d	S_d	P_d
data	227	1	–	–	–	–	–	–	–	–	–
conv1	55	96	11	4	0	1	1	10	19	4	0
pool1	27	96	3	2	0	–	1	–	–	–	–
conv2	27	256	5	1	2	2	1	6	12	3	0
pool2	13	256	3	2	0	–	1	–	–	–	–
conv3	13	384	3	1	1	1	1	5	9	1	0
conv4	13	384	3	1	1	2	1	5	9	1	0
conv5	13	256	3	1	1	2	1	6	8	1	0
pool5	6	256	3	1	1	–	1	–	–	–	–
fc6	1	4096	1	1	5	–	2	–	–	–	–
fc7	1	4096	1	1	0	–	1	–	–	–	–
fc8	1	*	1	1	0	–	1	–	–	–	–

Table 14: AlexNet architecture used for `Colorization` finetuning. **X** spatial resolution of layer, **C** number of channels in layer; **K** conv or pool kernel size; **S** computation stride; **D** kernel dilation; **P** padding; **G** group convolution, last layer is removed during transfer evaluation. Number with * depends on the colorization bin size. For evaluation, we downsample `conv` layers so that the resulting feature map has dimension $9k$. X_d downsampled spatial resolution; K_d kernel size of downsample `avgpool` layer; S_d stride of downsample `avgpool` layer; P_d padding of downsample using `avgpool` layer.

Layer	X	C	K	S	P	G	D	X_d	K_d	S_d	P_d
data	227	3	–	–	–	–	–	–	–	–	–
conv1	109	96	11	2	0	1	1	10	28	9	0
pool1	54	96	3	2	0	–	1	–	–	–	–
conv2	54	256	5	1	2	2	1	6	24	6	0
pool2	26	256	3	2	0	–	1	–	–	–	–
conv3	26	384	3	1	1	1	1	5	14	3	0
conv4	26	384	3	1	1	2	1	5	14	3	0
conv5	26	256	3	1	1	2	1	6	16	2	0
pool5	12	256	3	2	0	–	1	–	–	–	–
fc6	1	4096	1	1	0	–	1	–	–	–	–
fc7	1	4096	1	1	0	–	1	–	–	–	–
fc8	1	*	1	1	0	–	1	–	–	–	–

Table 15: AlexNet architecture used for `Jigsaw` finetuning. **X** spatial resolution of layer, **C** number of channels in layer; **K** conv or pool kernel size; **S** computation stride; **D** kernel dilation; **P** padding; **G** group convolution, last layer is removed during transfer evaluation. Number with * depends on the colorization bin size. For evaluation, we downsample `conv` layers so that the resulting feature map has dimension $9k$. X_d downsampled spatial resolution; K_d kernel size of downsample `avgpool` layer; S_d stride of downsample `avgpool` layer; P_d padding of downsample `avgpool` layer.

Layer	X	C	K	S	P	G	X_d	K_d	S_d	P_d
data	224	1	–	–	–	–	–	–	–	–
conv1	112	64	7	2	3	1	12	10	10	4
maxpool	56	64	3	2	1	–	–	–	–	–
res2	56	256	*	*	*	1	6	16	8	0
res3	28	512	*	*	*	1	4	13	5	0
res4	14	1024	*	*	*	1	3	8	3	0
res5	7	2048	*	*	*	1	2	6	1	0
avgpool	1	2048	7	1	0	–	–	–	–	–
fc1	1	†	1	1	0	–	–	–	–	–

Table 16: ResNet-50 architecture used for `Jigsaw` Transfer task. **X** spatial resolution of layer, **C** number of channels in layer; **K** conv or pool kernel size; **S** computation stride; **D** kernel dilation; **P** padding; **G** group convolution. Layers denoted with `res` prefix represent the bottleneck residual block. Number with * use the original setting as in [30]. Layer with † depend on the number of output classes. For evaluation, we downsample `conv` layers so that the resulting feature map has dimension $9k$. X_d downsampled spatial resolution; K_d kernel size of downsample `avgpool` layer; S_d stride of downsample `avgpool` layer; P_d padding of downsample using `avgpool` layer.

Layer	X	C	K	S	P	G	X_d	K_d	S_d	P_d
data	224	1	–	–	–	–	–	–	–	–
conv1	112	64	7	2	3	1	12	10	10	4
maxpool	56	64	3	2	1	–	–	–	–	–
res2	56	256	*	*	*	1	6	16	8	0
res3	28	512	*	*	*	1	4	13	5	0
res4	14	1024	*	*	*	1	3	8	3	0
res5	14	2048	*	*	*	1	2	12	2	0
avgpool	1	2048	14	1	0	–	–	–	–	–
fc1	1	†	1	1	0	–	–	–	–	–

Table 17: ResNet-50 architecture used for `Colorization` Transfer task. **X** spatial resolution of layer, **C** number of channels in layer; **K** conv or pool kernel size; **S** computation stride; **D** kernel dilation; **P** padding; **G** group convolution. Layers denoted with `res` prefix represent the bottleneck residual block. Number with * use the original setting as in [30]. Layer with † depend on the number of output classes. For evaluation, we downsample `conv` layers so that the resulting feature map has dimension $9k$. X_d downsampled spatial resolution; K_d kernel size of downsample `avgpool` layer; S_d stride of downsample `avgpool` layer; P_d padding of downsample using `avgpool` layer.

	ImageNet-1k				
Method	`layer1`	`layer2`	`layer3`	`layer4`	`layer5`
ResNet-50 ImageNet-1k Supervised	11.6	33.3	48.7	67.9	75.5
ResNet-50 Places205 Supervised	13.2	31.7	46.0	58.2	51.7
ResNet-50 Random	9.6	13.7	12.0	8.0	5.6
ResNet-50 (Kolesnikov *et al.*) [36]†	–	–	–	47.7	–
ResNet-50 (NPID) [75]$^{\triangleleft}$	15.3	18.8	24.9	40.6	54.0
ResNet-50 `Jigsaw` ImageNet-1k	**12.4**	28.0	<u>39.9</u>	45.7	34.2
ResNet-50 `Jigsaw` ImageNet-22k	7.9	**30.2**	39.0	46.3	35.9
ResNet-50 `Jigsaw` YFCC-100M	7.9	28.2	**41.3**	48.3	34.7
ResNet-50 `Coloriz.` ImageNet-1k	10.2	24.1	31.4	39.6	35.2
ResNet-50 `Coloriz.` ImageNet-22k	10.1	27.0	37.8	**49.3**	**46.2**
ResNet-50 `Coloriz.` YFCC-100M	10.1	27.7	36.2	48.4	40.2

Table 18: ResNet-50 top-1 center-crop accuracy for linear classification on the ImageNet-1k dataset.. Numbers with † are with $10-20\times$ longer fine-tuning and are reported on unofficial ImageNet-1k validation split. Numbers with $^{\triangleleft}$ use different fine-tuning procedure. All other models follow the setup from Zhang *et al.* [78].

Method	layer1	layer2	layer3	layer4	layer5
ResNet-50 Jigsaw ImageNet-1k	19.6	33.9	41.9	47.3	41.1
ResNet-50 Jigsaw ImageNet-22k	14.1	34.0	42.1	52.3	47.0
ResNet-50 Jigsaw YFCC-100M	14.7	34.2	43.7	52.1	44.4
ResNet-50 ImageNet-1k Supervised	17.5	35.5	45.8	60.5	68.5
AlexNet Jigsaw ImageNet-1k	25.8	35.0	38.6	38.6	34.6
AlexNet Jigsaw ImageNet-22k	25.9	35.6	39.6	39.3	34.1
AlexNet Jigsaw YFCC-100M	25.8	35.8	40.2	40.1	33.9
AlexNet ImageNet-1k Supervised	24.4	37.9	43.4	46.5	47.6

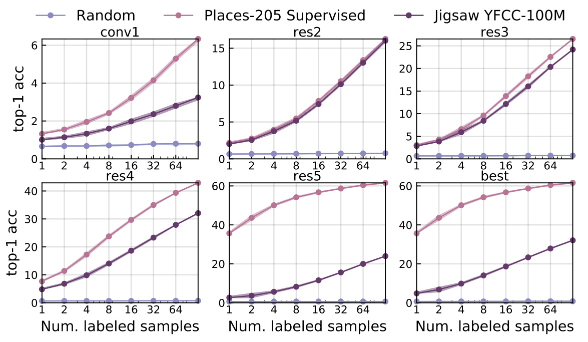
Table 19: Linear SVM classification on the COCO2014 dataset

Method	VOC07
ResNet-50 ImageNet-1k Supervised	90.3
ResNet-50 Places205 Supervised	83.3
ResNet-50 Random*	48.4
ResNet-50 Jigsaw ImageNet-1k	73.9
ResNet-50 Jigsaw ImageNet-22k	**83.2**
ResNet-50 Jigsaw YFCC-100M	<u>82.7</u>
ResNet-50 Coloriz. ImageNet-1k	67.9
ResNet-50 Coloriz. ImageNet-22k	75.9
ResNet-50 Coloriz. YFCC-100M	75.0

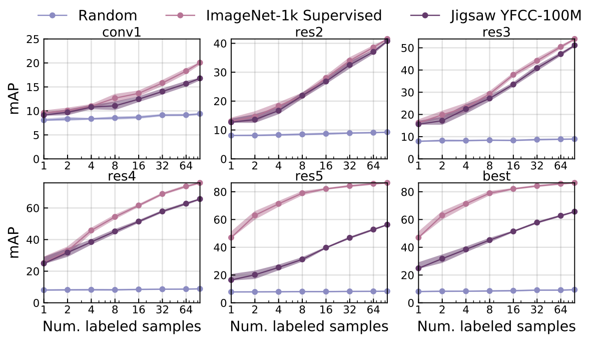
Table 20: ResNet-50 Full fine-tuning image classification (mAP scores) for VOC07. All models are trained using the same setup and we report center crop numbers. Random initialization baseline (denoted with *) is trained for $4\times$ longer fine-tuning schedule on VOC07.

Method	VOC07
AlexNet ImageNet-1k Supervised‡	79.9
AlexNet Places205 Supervised	75.0
AlexNet Random‡	53.3
AlexNet (Context) [14]$^\triangleleft$	65.3
AlexNet (SplitBrain) [78]$^\triangleleft$	67.1
AlexNet (Counting) [51]$^\triangleleft$	67.7
AlexNet (Rotation) [25]$^{\triangleleft,*}$	72.9
AlexNet (DeepCluster) [9]†	70.4
AlexNet Jigsaw ImageNet-1k	59.1
AlexNet Jigsaw ImageNet-22k	63.7
AlexNet Jigsaw YFCC-100M	63.1
AlexNet Coloriz. ImageNet-1k	61.9
AlexNet Coloriz. ImageNet-22k	**66.7**
AlexNet Coloriz. YFCC-100M	65.5

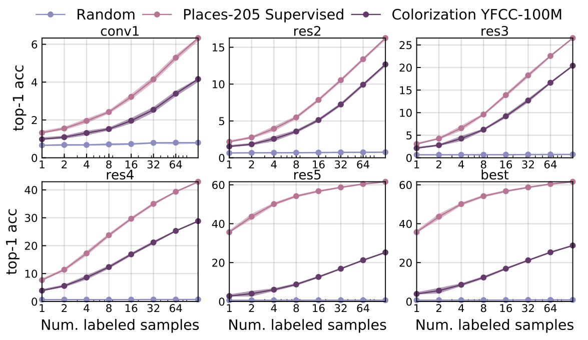
Table 21: AlexNet Full fine-tuning image classification (mAP scores) for VOC07: We report 10-crop numbers as in [78]. Method with † uses a different fine-tuning schedule, $^\triangleleft$ uses weight re-scaling, * we could not determine exact fine-tuning details. Numbers with ‡ taken from [78]. We note that drawing consistent comparisons with (and among) prior work is difficult because of differences in the fine-tuning procedure and thus present these results only for the sake of completeness.



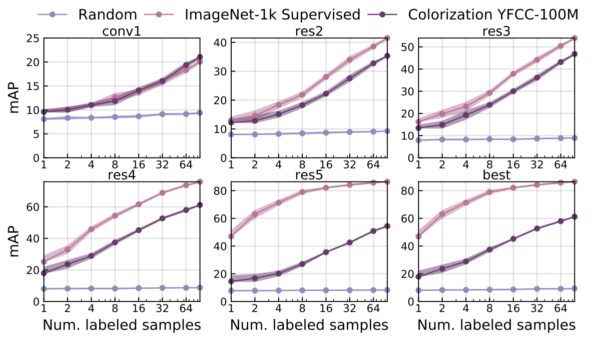
(a) `Jigsaw` Places205



(b) `Jigsaw` VOC07



(a) `Colorization` Places205



(b) `Colorization` VOC07

Figure 7: Low-shot Classification (layerwise) results on the using a ResNet-50. These are additional results following the same setup as in Section 6.2 of the main paper. We train linear classifiers (SVMs) on layer-wise representations.

Initialization	Angle Distance		Within t°		
	Mean	Median	11.25	22.5	30
	(Lower is better)		(Higher is better)		
ResNet-50 ImageNet-1k supervised	26.4	17.1	36.1	59.2	68.5
ResNet-50 Places205 supervised	23.3	14.2	41.8	65.2	73.6
ResNet-50 Scratch	26.3	16.1	37.9	60.6	69.0
`Colorization` YFCC-100M	28.5	22.6	28.1	49.9	61.4
`Colorization` ImageNet-22k	27.1	19.5	32.4	55.1	65.6
`Colorization` ImageNet-1k	29.3	21.0	30.2	52.4	62.8

Table 22: Surface Normal Estimation on the NYUv2 dataset. We train ResNet-50 from `res5` onwards and freeze the `conv` body below.

Method	VOC07	VOC07+12
ResNet-50 ImageNet-1k Supervised	67.1	68.3
ResNet-50 ResNet-50 `Jigsaw` ImageNet-22k	62.7	64.8
ResNet-50 `Jigsaw` YFCC-100M	56.9	59.8

Table 23: Detection mAP for frozen `conv` body on VOC07 and VOC07+12 using Faster R-CNN with ResNet-50-C4. We freeze the `conv` body for all models.

Method	VOC07	VOC07+12
ResNet-50 ImageNet-1k Supervised	70.9	76.4
ResNet-50 ResNet-50 `Jigsaw` ImageNet-1k	64.5	67.3
ResNet-50 `Jigsaw` ImageNet-22k	68.9	75.3
ResNet-50 `Jigsaw` YFCC-100M	66.4	73.9

Table 24: Detection mAP with full fine-tuning on VOC07 and VOC07+12 using Faster R-CNN with ResNet-50-C4. We freeze the `conv` body for all models.

	VOC07				
Number of patches (N)	`layer1`	`layer2`	`layer3`	`layer4`	`layer5`
9	26.7	44.6	53.5	64.1	55.5
16	31.9	42.0	48.1	49.8	37.9

Table 25: Varying number of patches N for a ResNet-50 on `Jigsaw`. We increase the problem complexity of the `Jigsaw` method by increasing the number of patches from 9 (default in [50]) to 16. We keep the size of the permutation set fixed at $|\mathcal{P}| = 2000$. We report the performance of training a linear SVM on the fixed features for the VOC07 image classification task. We do not see an improvement by increasing the number of patches.

	Places205				
Number of bins ($|\mathcal{Q}|$)	`layer1`	`layer2`	`layer3`	`layer4`	`layer5`
968	14.5	27.8	32.0	33.1	36.3
313	14.5	27.5	32.8	37.6	34.6
262	14.8	27.9	32.5	38.6	36.5
124	14.5	26.6	30.7	27.7	35.3
76	15.6	28.1	32.7	33.6	36.5

Table 26: Varying number of colorbins $|\mathcal{Q}|$ for a ResNet-50 on `Colorization`. We increase the problem complexity for the `Colorization` method by increasing the number of colors ($|\mathcal{Q}|$) the ConvNet must predict. We evaluate the feature representation by training linear classifiers on the fixed features. We report the top-1 center crop accuracy on the Places205 dataset.