Colorization as a Proxy Task for Visual Understanding

Gustav Larsson
University of Chicago
larsson@cs.uchicago.edu

Michael Maire
TTI Chicago
mmaire@ttic.edu

Gregory Shakhnarovich
TTI Chicago
greg@ttic.edu

Abstract

We investigate and improve self-supervision as a drop-in replacement for ImageNet pretraining, focusing on automatic colorization as the proxy task. Self-supervised training has been shown to be more promising for utilizing unlabeled data than other, traditional unsupervised learning methods. We build on this success and evaluate the ability of our self-supervised network in several contexts. On VOC segmentation and classification tasks, we present results that are state-of-the-art among methods not using ImageNet labels for pretraining representations.

Moreover, we present the first in-depth analysis of self-supervision via colorization, concluding that formulation of the loss, training details and network architecture play important roles in its effectiveness. This investigation is further expanded by revisiting the ImageNet pretraining paradigm, asking questions such as: How much training data is needed? How many labels are needed? How much do features change when fine-tuned? We relate these questions back to self-supervision by showing that colorization provides a similarly powerful supervisory signal as various flavors of ImageNet pretraining.

1. Introduction

The success of deep feed-forward networks is rooted in their ability to scale up with more training data. The availability of more data can generally afford an increase in model complexity. However, this need for expensive, tedious and error-prone human annotation is severely limiting, reducing our ability to build models for new domains, and for domains in which annotations are particularly expensive (e.g., image segmentation). At the same time, we have access to enormous amounts of unlabeled visual data, which is essentially free. This work is an attempt to improve means of leveraging this abundance. We manage to bring it one step closer to the results of using labeled data, but the eventual long term goal of self-supervision may be to supplant supervised pretraining completely.

Alternatives to supervised training that do not need labeled data have seen limited success. Unsupervised learning methods, such as compressed embeddings trained by minimizing reconstruction error, have seen more success in image synthesis [18], than for representation learning. Semi-supervised learning, jointly training a supervised and an unsupervised loss, offers a middle ground [7, 35]. However, recent works tend to prefer a sequential combination instead (unsupervised pretraining, supervised fine-tuning) [4, 5], possibly because it prevents the unsupervised loss from being disruptive in the late stages of training. A related endeavor to unsupervised learning is developing models that work with weaker forms of supervision [2, 40]. This reduces the human burden only somewhat and pays a

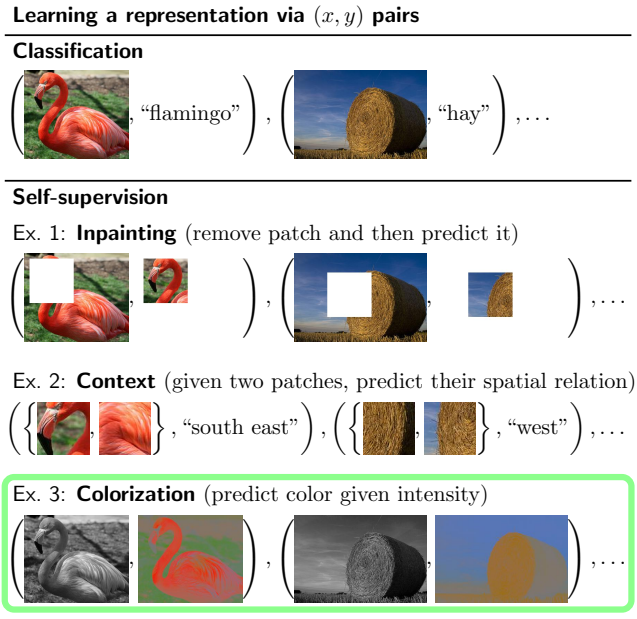

Figure 1. Using a representation that was originally trained for classification on (x, y) pairs to initialize a network has become standard practice in computer vision. Self-supervision is a family of alternative pretraining methods that do not require any labeled data, since labels are "manufactured" through unlabeled data. We focus on colorization, where an image is split into its intensity and color components, the former predicting the latter.

price in model performance.

Recently, *self-supervision* has emerged as a new flavor of unsupervised learning [4, 38]. The key observation is that perhaps part of the benefit of labeled data is that it leads to using a discriminative loss. This type of loss may be better suited for representation learning than, for instance, a reconstruction or likelihood-based loss. Self-supervision is a way to use a discriminative loss on unlabeled data by partitioning each input sample in two, predicting the parts' association. We focus on self-supervised colorization [20, 42], where each image is split into its intensity and its color, using the former to predict the latter.

Our main contributions to self-supervision are:

- State-of-the-art results on VOC 2007 Classification and VOC 2012 Segmentation, among methods that do not use ImageNet labels.
- The first in-depth analysis of self-supervision via colorization. We study the impact of loss, network architecture and training details, showing that there are many important aspects that influence results.
- An empirical study of various formulations of ImageNet pretraining and how they compare to self-supervision.

2. Related work

In our work on replacing classification-based pretraining for downstream supervised tasks, the first thing to consider is clever network initializations. Networks that are initialized to promote uniform scale of activations across layers, converge more easily and faster [6, 9]. The uniform scale however is only statistically predicted given broad data assumptions, so this idea can be taken one step further by looking at the activations of actual data and normalizing [23]. Using some training data to initialize weights blurs the line between initialization and unsupervised pretraining. For instance, using layer-wise k-means clustering [3, 19] should be considered unsupervised pretraining, even though it may be a particularly fast one.

Unsupervised pretraining can be used to facilitate optimization or to expose the network to orders of magnitude larger unlabeled data. The former was once a popular motivation, but fell out of favor as it was made unnecessary by improved training techniques (*e.g.* introduction of non-saturating activations [27], better initialization [6] and training algorithms [32, 17]). The second motivation of leveraging more data, which can also be realized as semi-supervised training, is an open problem with current best methods rarely used in competitive vision systems.

Recent methods on self-supervised feature learning have seen several incarnations, broadly divided into methods that exploit temporal or spatial structure in natural visual data:

Temporal. There have been a wide variety of methods that use the correlation between adjacent video frames as a learning signal. One way is to try to predict future frames, which is an analogous task to language modeling and often uses similar techniques based on RNNs and LSTMs [36, 33]. It is also possible to train an embedding where temporally close frames are considered similar (using either pairs [25, 14, 15] or triplets [38]). Another method that uses a triplet loss presents three frames and tries to predict if they are correctly ordered [24]. Pathak *et al.* [30] learn general-purpose representation by predicting saliency based on optical flow. Owens *et al.* [29], somewhat breaking from the temporal category, operate on a single video frame to predict a statistical summary of the audio from the entire clip. The first video-based self-supervision methods were based on Independent Component Analysis (ICA) [37, 10]. Recent follow-up work generalizes this to a nonlinear setting [11].

Spatial. Methods that operate on single-frame input typically use the spatial dimensions to divide samples for self-supervision. Given a pair of patches from an image, Doerch *et al.* [4] train representations by predicting which of eight possible spatial compositions the two patches have. Noroozi & Favaro [28] take this further and learns a representation by solving a 3-by-3 jigsaw puzzle. The task of inpainting (remove some pixels, then predict them) is utilized for representation learning by Pathak *et al.* [31]. There has also been work on using bi-directional Generative Adversarial Networks (BiGAN) to learn representations [5]. This is not what we typically regard as self-supervision, but it does similarly pose a supervised learning task (*real* vs. *synthetic*) on unlabeled data to drive representation learning.

Colorization. Lastly there is colorization [20, 42, 43]. Broadly speaking, the two previous categories split input samples along a spatio-temporal line, either predicting one given the other or predicting the line itself. Automatic colorization departs from this as it asks to predict color over the same pixel as its center of input, without discarding any spatial information. We speculate that this may make it more suitable to tasks of similar nature, such as semantic segmentation; we demonstrate strong results on this benchmark.

Representation learning via colorization was first presented as part of two automatic colorization papers [20, 42]. Zhang *et al.* [42] present results across all PASCAL tasks and show colorization as a front-runner of self-supervision. However, like most self-supervision papers, it is restricted to AlexNet and thus shows modest results compared to recent supervised methods. Larsson *et al.* [20] present state-of-the-art results on PASCAL VOC semantic segmentation, which we improve by almost 10 points from 50.2% to 60.0% mIU. Both papers present the results with little analysis or investigation.

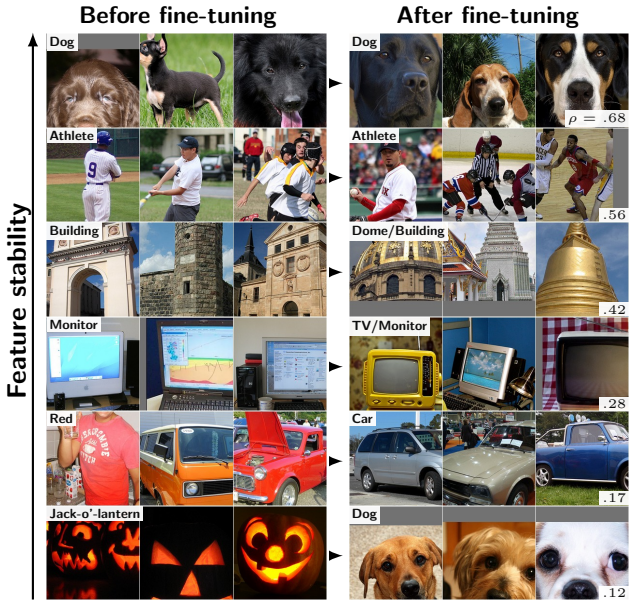

Figure 2. **Feature reuse/repurpose.** The left column visualizes top activations from the colorization network (same as in Fig. 5). The right column visualizes the corresponding feature after the network has been fine-tuned for semantic segmentation. Features are either re-used as is (top), specialized (middle), or scrapped and replaced (bottom). See Fig. 3 for a quantitative study.

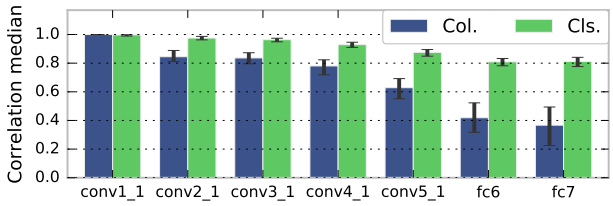

Figure 3. **Feature shift.** The correlation between feature activations for layers of VGG-16 before and after fine-tuning for semantic segmentation. The bar heights indicate median correlation and error bars indicate interquartile range. See Fig. 2 for qualitative examples.

3. Colorization as the target task

Training an automatic colorizer for the purpose of being able to convert grayscale photos to color is an active area of research [20, 42, 12]. Recent methods train deep convolutional neural networks to predict color [12] or distributions over color [20, 42]. The latter approach is followed by instantiating a color from the histogram prediction in order to produce a final result. For optimal colorization results, these networks are initialized with a classification-based network, in order to leverage its high-level features and thus better predict color. In this section we describe how to train colorization, revisiting some of the design decisions that were made with the goal of producing aesthetic color images and instead consider their impact on learning representations.

3.1. Training

Our experimental setup borrows heavily from Larsson *et al.* [20], using Caffe [16] and their public source code release for training the colorization network. For downstream tasks, we use TensorFlow [1] and provide testing code as well as trained models.[1]

Loss. We consider both a regression loss for L*a*b color values [20, 42, 12], as well as a KL divergence loss for hue/chroma histograms [20]. For the latter, the histograms are computed from a 7-by-7 window around each target pixel and placed into 32 bins for hue and 32 bins for chroma. We evaluate their ability to learn representations, disregarding their ability to do colorization. In our comparison, we make sure that the losses are scaled similarly, so that their effective learning rates are as close as possible.

Hypercolumn. The networks use hypercolumns [22, 26, 8] with sparse training [20]. This means that for each image, only a small sample of hypercolumns are computed. This reduces memory requirements and allows us to train on larger images. Note that hypercolumns can be used for colorization pretraining, as well as for segmentation as a downstream task. Since we have reasons to believe that hypercolumn training may disrupt residual training, we do not train our ResNet colorizer with hypercolumns.

Dataset. We train on 3.7M *unlabeled* images by combining 1.3M from ImageNet [34] and 2.4M from Places205 [45]. The dataset contains some grayscale images, but we do not make an effort to sort them out, since there is no way to tell a legitimately achromatic image from a desaturated one.

Training. All training is done with standard Stochastic Gradient Descent with momentum set to 0.9. The colorization network is initialized with Xavier initialization [6] and trained with batch normalization without re-biasing or re-scaling parameters [13]. Each time an image is processed, it is randomly mirrored and the image is randomly scaled such that the shortest side is between 352 and 600. Finally, a 352-by-352 patch is extracted and desaturated and then fed through the network. In our comparative studies, we train using a colorization loss for 3 epochs (spending 2 epochs on the initial learning rate). In our longer running experiments, we trained for about 10 epochs. For our best ResNet model, we trained significantly longer (35 epochs), although on smaller inputs (224-by-224); we found large input sizes to be more important during downstream training.

4. Colorization as a proxy task

Shifting our focus to using a colorization network purely for its visual representations, we describe how it can help improve results on classification and segmentation.

[1] https://github.com/gustavla/self-supervision

4.1. Training

The downstream task is trained by initializing weights from the colorization-from-scratch network. Some key considerations follow:

Early stopping. Training on a small sample size is prone to overfitting. We have found that the most effective method of preventing this is carefully cross validating the learning rate schedule. Models that initialize differently (random, colorization, classification), need very different early stopping schedules. Finding a method that works well in all these settings was key to our study. We split the training data 90/10 and only train on the 90%; the rest is used to monitor overfitting. Each time the 10% validation score (not surrogate loss) stops improving, the learning rate is dropped. After this is done twice, the training is concluded. For our most competitive experiments (Tab. 1), we then retrain using 100% of the data with the cross-validated learning rate schedule fixed.

Receptive field. Previous work on semantic segmentation has shown the importance of large receptive fields [26, 41]. One way of accomplishing this is by using dilated convolutions [41, 39], however this redefines the interpretation of filters and thus requires re-training. Instead, we add two additional blocks (2-by-2 max pooling of stride 2, 3-by-3 convolution with 1,024 features) at the top of the network, each expanding the receptive field with 160 pixels per block. We train on large input images (448-by-448) in order to fully appreciate the enlarged receptive field.

Hypercolumn. Note that using a hypercolumn when the downstream task is semantic segmentation is a separate design choice that does not need to be coupled with the use of hypercolumns during colorization pretraining. In either case, the post-hypercolumn parameter weights are never reused. For ResNet, we use a subset of the full hypercolumn.[2]

Batch normalization. The models trained from scratch use parameter-free batch normalization. However, for downstream training, we absorb the mean and variance into the weights and biases and train without batch normalization (with the exception of ResNet, where in our experience it helps). For networks that were not trained with batch normalization and are not well-balanced in scale across layers (*e.g.* ImageNet pretrained VGG-16), we re-balance the network so that each layer's activation has unit variance [20].

Padding. For our ImageNet pretraining experiments, we observe that going from a classification network to a fully convolutional network can introduce edge effects due to each layer's zero padding. A problem not exhibited by the original VGG-16, leading us to suspect that it may be due to the introduction of batch normalization. For the newly trained networks, activations increase close to the edge, even though the receptive fields increasingly hang over the edge of the image, reducing the amount of semantic information. Correcting for this[3] makes activations well-behaved, which was important in order to appropriately visualize top activations. However, it does not offer a measurable improvement on downstream tasks, which means the network can correct for this during the fine-tuning stage.

Color. Since the domain of a colorization network is grayscale, our downstream experiments operate on grayscale input unless otherwise stated. When colorization is re-introduced, we convert the grayscale filters in `conv1_1` to RGB (replicate to all three channels, divide by three) and let them fine-tune on the downstream task.

Initialization		Architecture	Class. %mAP	Seg. %mIU
ImageNet	(+FoV)	VGG-16	86.9	69.5
Random (ours)		AlexNet	46.2	23.5
Random [31]		AlexNet	53.3	19.8
k-means [19, 5]		AlexNet	56.6	32.6
k-means [19]		VGG-16	56.5	-
k-means [19]		GoogLeNet	55.0	-
Pathak *et al.* [31]		AlexNet	56.5	29.7
Wang & Gupta [38]		AlexNet	58.7	-
Donahue *et al.* [5]		AlexNet	60.1	35.2
Doersch *et al.* [4, 5]		AlexNet	65.3	-
Zhang *et al.* (col) [42]		AlexNet	65.6	35.6
Zhang *et al.* (s-b) [43]		AlexNet	67.1	36.0
Noroozi & Favaro [28]		Mod. AlexNet	68.6	-
Larsson *et al.* [20]		VGG-16	-	50.2
Our method		AlexNet	65.9	38.4
	(+FoV)	VGG-16	**77.2**	56.0
	(+FoV)	ResNet-152	**77.3**	**60.0**

Table 1. **VOC Comparison.** Comparison with other initialization and self-supervision methods on VOC 2007 Classification (test) and VOC 2012 Segmentation (val). Note that our baseline AlexNet results (38.4%) are also the most competitive among AlexNet models. The use of a hypercolumn instead of FCN is partly responsible: running Zhang *et al.*'s colorization model with a hypercolumn yields 36.4%, only a slight improvement over 35.6%. Switching to ResNet, adding a larger FoV, and training even longer yields a significantly higher result at 60.0% mIU. Note, the "+FoV" only affects the segmentation results. The modified AlexNet used by Noroozi & Favaro has the same number of parameters as AlexNet, with a spatial reduction of 2 moved from conv1 to pool5, increasing the size of the intermediate activations.

[2] ResNet-152 hypercolumn: `conv1`, `res2{a,b,c}`, `res3b{1,4,7}`, `res4b{5,10,15,20,25,30,35}`, `res5c`

[3] We pad with the bias from the previous layer, instead of with zeros. This is an estimate of the expectation value, since we use a parameter-free batch normalization with zero mean, leaving only the bias.

Pretraining Loss	Seg. (%mIU)
Regression	48.0
Histograms (no hypercolumn)	52.7
Histograms	52.9

Table 2. **Self-supervision loss.** (VGG-16) The choice of loss has a significant impact on downstream performance. However, pretraining with a hypercolumn does not seem to benefit learning. We evaluate this on VOC 2012 Segmentation (val) with a model that uses hypercolumns, regardless of whether or not it was used during pretraining.

5. Results

We first present results on two established PASCAL VOC benchmarks, followed in Section 6 by an investigation into different design choices and pretraining paradigms.

5.1. PASCAL

VOC 2012 Semantic Segmentation. We train on the standard extended segmentation data (10,582 samples) and test on the validation set (1,449 samples). We sample random crops at the original scale. Using our ResNet-152 model with extended field-of-view we achieve 60.0% mIU (see Tab. 1), the highest reported results on this benchmark that do not use supervised pretraining. It is noticeable that this value is considerably higher than the AlexNet-based FCN [21] (48.0%) and even slightly higher than the VGG-16-based FCN (59.4%[4]), both methods trained on ImageNet.

VOC 2007 Classification. We train on the trainval (5,011 samples) and test on the test set (4,952 samples). We use the same training procedure with 10-crop testing as in [5]. Our results at 77.3% mAP (see Tab. 1) are state-of-the-art when no ImageNet labels are used.

6. Experiments

We present a wide range of experiments, highlighting important aspects of our competitive results. For these studies, in addition to VOC 2012 Semantic Segmentation, we also use two classification datasets that we constructed:

ImNt-100k/ImNt-10k. Similar to ImageNet classification with 1000 classes, except we have limited the training data to exactly 100 and 10 samples/class, respectively. In addition, all images are converted to grayscale. We test on ImageNet val with single center crops of size 224-by-224, making the results easy to compare with full ImageNet training. For our pretraining experiments in Tab. 4, we also use these datasets to see how well they are able to substitute the entire ImageNet dataset for representation learning.

[4]Both of these values refer to VOC 2011 and evaluated on only 736 samples, which means the comparison is imprecise.

Architecture	Init.	Seg.	+FoV	ImNt-100k	10k
		%mIU		%top-5	
AlexNet	Rnd	23.5	24.6	39.1	6.7
AlexNet	Col	36.2	40.8	48.2	17.4
VGG-16	Rnd	32.8	35.1	43.2	8.6
VGG-16	Col	50.7	52.9	59.0	23.3
ResNet-152	Rnd	*9.9	*10.5	42.5	8.1
ResNet-152	Col	52.3	53.9	63.1	29.6

Table 3. **Architectures.** We compare how various networks perform on downstream tasks on random initialization (Rnd) and colorization pretrained (Col). For our segmentation results, we also consider the effects of increasing the receptive field size (+FoV). Training residuals from scratch (marked with a *) is possibly compromised by the hypercolumn, causing the low values.

6.1. Loss

As seen in Tab. 2, regressing on color in the L*a*b space yields a 5-point lower result (48.0%) than predicting histograms in hue/chroma (52.9%). This demonstrates that the choice of loss is of crucial importance to representation learning. This is a much larger difference than Larsson *et al.* [20] report in colorization performance between the two methods (24.25 and 24.45 dB PSNR / 0.318 and 0.299 RMSE). Histogram predictions are meant to address the problem of color uncertainty. However, the way they instantiate an image by using summary statistics from the histogram predictions, means this problem to some extent is re-introduced. Since we do not care about instantiating images, we do not suffer this penalty and thus see a much larger improvement using a loss based on histogram predictions. Our choice of predicting separate histograms in the hue/chroma space also yields an interesting finding in Fig. 5, where we seem to have non-semantic filters that respond to input with high chromaticity as well as low chromaticity, clearly catering to the chroma prediction.

6.2. Network architecture

The investigation into the impact of network architecture has been a neglected aspect of recent self-supervision work, which has focused only on AlexNet. We present the first detailed study into the untapped potential of using more modern networks. These results are presented in Tab. 3.

It is not entirely obvious that an increase in model complexity will pay off, since our focus is small-sample datasets and a smaller network may offer a regularizing effect. Take ImNt-100k, where AlexNet, VGG-16, and ResNet-152 all perform similarly when trained from scratch (39.1%, 43.2%, 42.5%). However, the percentage point improvement when utilizing colorization pretraining follows a clear trend (+9.1, +15.8, +20.6). This shows that self-supervision

Pretraining	Samples	Epochs	Seg. (%mIU)
None	-	-	35.1
C1000	1.3M	80	66.5
C1000	1.3M	20	62.0
C1000	100k	250	57.1
C1000	10k	250	44.4
E10	(1.17M) 1.3M	20	61.8
E50	(0.65M) 1.3M	20	59.4
H16	1.3M	20	60.0
H2	1.3M	20	46.1
R50	1.3M	20	57.3
		40	59.4
R16	1.3M	20	42.6
		40	53.5

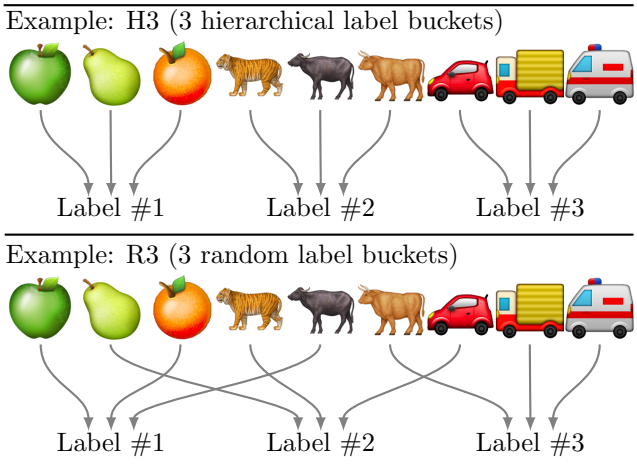

Table 4. **ImageNet pretraining.** We evaluate how useful various modifications of ImageNet are for VOC 2012 Segmentation (val-gray). We create new datasets either by reducing sample size or by reducing the label space. The former is done simply by reducing sample size or by introducing 10% (E10) or 50% (E50) label noise. The latter is done using hierarchical label buckets (H16 and H2) or random label buckets (R50 and R16). The model trained for 80 epochs is the publicly available VGG-16 (trained for 76 epochs) that we fine-tuned for grayscale for 4 epochs. The rest of the models were trained from scratch on grayscale images.

allows us to benefit from higher model complexity even in small-sample regimes. Compare this with k-means initialization [19], which does not show any improvements when increasing model complexity (Tab. 1).

Training ResNet from scratch for semantic segmentation is an outlier value in the table. This is the only experiment that trains a residual network from scratch together with a hypercolumn; this could be a disruptive combination as the low numbers suggest.

Initialization	Grayscale input	Color input
Classification	66.5	69.5
Colorization	56.0	55.9

Table 5. **Color vs. Grayscale input.** (VOC 2012 Segmentation, %mIU) Even though our classification-based model does 3 points better using color, re-introducing color yields no benefit.

6.3. ImageNet pretraining

We relate self-supervised pretraining to ImageNet pretraining by revisiting and reconsidering various aspects of this paradigm (see Tab. 4). First of all, we investigate the importance of 1000 classes (C1000). To do this, we join ImageNet classes together based on their place in the WordNet hierarchy, creating two new datasets with 16 classes (H16) and only two classes (H2). We show that H16 performs only slightly short of C1000 on a downstream task with 21 classes, while H2 is significantly worse. If we compare this to our colorization pretraining, it is much better than H2 and only slightly worse than H16.

Next, we study the impact of sample size, using the subsets ImNt-100k and ImNt-10k described in Section 6. ImNt-100k does similarly well as self-supervised colorization (57.1% vs. 56.0% for VGG-16), suggesting that our method has roughly replaced 0.1 million labeled samples with 3.7 million unlabeled samples. Reducing samples to 10 per class sees a bigger drop in downstream results. This result is similar to H2, which is somewhat surprising: collapsing the label space to a binary prediction is roughly as bad as using 1/100th of the training data. Recalling the improvements going from regression to histogram prediction for colorization, the richness of the label space seems critical for representation learning.

We take the 1000 ImageNet classes and randomly place them in 50 (R50) or 16 (R16) buckets that we dub our new labels. This means that we are training a highly complex decision boundary that may dictate that a golden retriever and a minibus belong to the same label, but a golden retriever and a border collie do not. We consider this analogous to self-supervised colorization, since the supervisory signal similarly considers a red car arbitrarily more similar to a red postbox than to a blue car. Not surprisingly, our contrived dataset R50 results in a 5-point drop on our downstream task, and R16 even more so with a 20-point drop. However, we noticed that the training loss was still actively decreasing after 20 epochs. Training instead for 40 epochs showed an improvement by about 2 points for R50, while 11 points for R16. In other words, complex classes can provide useful supervision for representation learning, but training may take longer. This is consistent with our impression of self-supervised colorization; although it converges slowly, it keeps improving its feature generality with

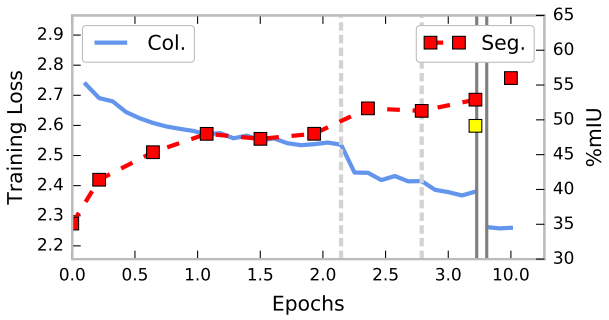

Figure 4. **Learning rate.** The blue line shows colorization training loss and the vertical dashed lines are scheduled learning rate drops. The red squares are results on a downstream task (VOC 2012 Segmentation) initialized by the corresponding snapshot of the colorization network. Some key observations: We quickly get value for our money, with a 6-point improvement over random initialization with only 0.2 epochs of training. Furthermore, improvements on the downstream task do not quickly saturate, with results improving further when trained 10 epochs in total. Dropping the learning rate on the pretraining task helps the downstream task, with a similarly abrupt improvement as with the training loss around 2 epochs. Training the full 3 epochs without ever dropping the learning rate results in 49.1% (yellow square) compared to 52.9% mIU.

more training.

Finally, we test the impact of label noise. When 10% of the training images are re-assigned a random label (E10), it has little impact on downstream performance. Increasing the label noise to 50% (E50) incurs a 2.6-point penalty, but it is still able to learn a competitive representation.

6.4. Training time and learning rate

We show in Fig. 4 that it is crucial for good performance on downstream tasks to reduce learning rate during pretraining. This result was not obvious to us, since it is possible that the late stage of training with low learning rate is too task-specific and will not benefit feature generality.

In addition, we show the importance of training time by demonstrating that training for three times as long (10 epochs, 37M samples) improves results from 52.9% to 56.0% mIU on VOC 2012 Segmentation. Our ResNet-152 model (60.0% mIU) trained for 4 months on a single GPU.

6.5. Latent representation

Good results on secondary tasks only give evidence that our self-supervised network has the potential to be shaped into a useful representation. We investigate if the representation learned through colorization is immediately useful or only holds a latent representation. If the latter, how is our representation different from a good initialization scheme?

First, we visualize features to get a sense of how the colorization network has organized the input into features. We posit that we will find features predictive of color, since we know that the colorization network is able to predict color with good accuracy. In Fig. 5, we visualize top activations from the network's most high-level layer, and indeed we find color-specific features. However, we also find semantic features that group high-level objects with great intra-class variation (color, lighting, pose, etc.). This is notable, since no labeled data was used to train the network. The notion of objects has emerged purely through their common color and visual attributes (compare with [44]). Object-specific features should have high task generality and be useful for downstream tasks. Features that are specific to both object and color (bottom-right quadrant in Fig. 5) can be divided into two categories: The first is when the object generally has a unimodal color distribution (e.g. red bricks, brown wood); the second is when the network has learned a color sub-category of an object with multimodal color distribution (e.g. white clothing, yellow vehicle). These should all have high task generality, since it is easy for a task-specific layer to consolidate several color sub-categories into color-invariant notions of objects.

Fine-tuned layers (VGG-16)		Rnd	Col	Cls
∅	□□□□□□□	3.6	36.5	60.8
fc6, fc7	□□□□□■■	-	42.6	63.1
conv4_1..fc7	□□□■■■■	-	53.6	64.2
conv1_1..fc7	■■■■■■■	35.1	56.0	66.5

Table 6. **VOC 2012 Segmentation.** (%mIU) Classification-based pretraining (Cls) needs less fine-tuning than our colorization-based method (Col). This is consistent with our findings that our network experiences a higher level of feature shift (Fig. 3). We also include results for a randomly initialized network (Rnd), which does not work at all without fine-tuning (3.6%). This is to show that it is not simply by virtue of the hypercolumn that we are able to do reasonably well (36.5%) without any fine-tuning of the base network.

So how much do the features change when fine-tuned? We visualize top activations before and after in Fig. 2 and show in Fig. 3 that the colorization features change much more than label-based features. Some features are completely repurposed, many are only pivoted, and others remain more or less the same. These results are consistent with the four quadrants in Fig. 5, that show that some features are specific to colorization, while others seem to have general purpose.

Next, we look at how much fine-tuning is required for the downstream task. Tab. 6 tells us that even though fine-tuning is more important than for supervised pretraining (consistent with the correlation results in Fig. 3), it is able to perform the task with the colorization features alone similarly well as randomly initializing the network and training it end-to-end from scratch.

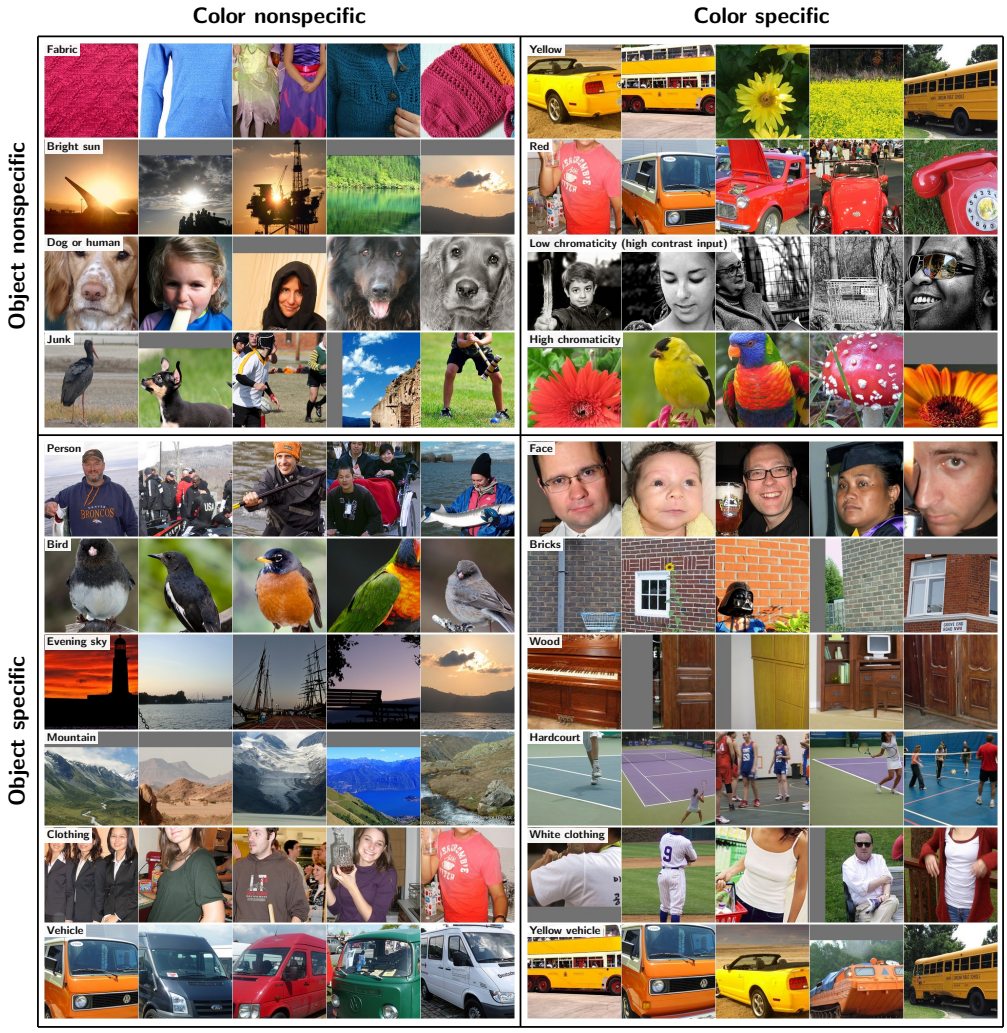

Figure 5. **Feature visualization.** Patches around activations from held-out images are visualized for a select number of `fc7` features (VGG-16). Even though the network takes only grayscale input, we visualize each patch in its original color for the benefit of the reader. As a result, if all the activations are consistent in color (right column), the feature is predictive of color. Similarly, if a feature is semantically coherent (bottom row), it means the feature is predictive of an object class. The names of each feature are manually set based on the top activations.

Somewhat poor results without fine-tuning and a lower percentage of feature re-use supports the notion that the colorization network in part holds *latent* features. However, the visualized features and the strong results overall suggest that we have learned something much more powerful than a good initialization scheme.

6.6. Color

We show in Tab. 5 that re-introducing color yields no benefit (consistent with the findings of Zhang *et al.* [42]). However, concurrent work [43] presents a better method of leveraging the color channels by separately training a network for the "opposite" task (predicting intensity from color). The two separate networks are combined for downstream use.

7. Conclusion

We have presented a drop-in replacement for ImageNet pretraining, with state-of-the-art results on semantic segmentation and small-sample classification that do not use ImageNet labels. A detailed investigation into self-supervised colorization shows the importance of the loss, network architecture and training details in achieving competitive results. We draw parallels between this and ImageNet pretraining, showing that self-supervision is on par with several methods using annotated data.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of GPUs used for this research.

References

[1] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What's the Point: Semantic Segmentation with Point Supervision. In *ECCV*, 2016.

[3] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[4] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[5] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2017.

[6] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[7] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, 2004.

[8] B. Hariharan, P. A. an R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. *CVPR*, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[10] J. Hurri and A. Hyvärinen. Simple-cell-like receptive fields maximize temporal coherence in natural video. *Neural Computation*, 15(3):663–691, 2003.

[11] A. Hyvarinen and H. Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *NIPS*, 2016.

[12] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4), 2016.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[14] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811*, 2015.

[15] D. Jayaraman and K. Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *CVPR*, 2016.

[16] 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.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[18] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[19] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. In *ICLR*, 2016.

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

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[22] M. Maire, S. X. Yu, and P. Perona. Reconstructive sparse code transfer for contour detection and semantic labeling. In *ACCV*, 2014.

[23] D. Mishkin and J. Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[24] I. Misra, C. L. Zitnick, and M. Hebert. Unsupervised learning using sequential verification for action recognition. 2016.

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

[26] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015.

[27] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

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

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

[30] D. Pathak, R. B. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. *CoRR*, abs/1612.06370, 2016.

[31] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[32] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[33] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014.

[34] 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. *International Journal of Computer Vision (IJCV)*, 115(3), 2015.

[35] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *NIPS*, 2016.

[36] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[37] J. H. van Hateren and D. L. Ruderman. Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings of the Royal Society of London B: Biological Sciences*, 265(1412):2315–2320, 1998.

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

[39] Z. Wu, C. Shen, and A. van den Hengel. High-performance semantic segmentation using very deep fully convolutional networks. *CoRR*, abs/1604.04339, 2016.

[40] J. Xu, A. G. Schwing, and R. Urtasun. Learning to segment under various forms of weak supervision. In *CVPR*, 2015.

[41] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

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

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

[44] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.

[45] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, pages 487–495, 2014.