A Simple Framework for Contrastive Learning of Visual Representations

Ting Chen [1] **Simon Kornblith** [1] **Mohammad Norouzi** [1] **Geoffrey Hinton** [1]

Abstract

This paper presents *SimCLR*: a simple framework for contrastive learning of visual representations. We simplify recently proposed contrastive self-supervised learning algorithms without requiring specialized architectures or a memory bank. In order to understand what enables the contrastive prediction tasks to learn useful representations, we systematically study the major components of our framework. We show that (1) composition of data augmentations plays a critical role in defining effective predictive tasks, (2) introducing a learnable nonlinear transformation between the representation and the contrastive loss substantially improves the quality of the learned representations, and (3) contrastive learning benefits from larger batch sizes and more training steps compared to supervised learning. By combining these findings, we are able to considerably outperform previous methods for self-supervised and semi-supervised learning on ImageNet. A linear classifier trained on self-supervised representations learned by SimCLR achieves 76.5% top-1 accuracy, which is a 7% relative improvement over previous state-of-the-art, matching the performance of a supervised ResNet-50. When fine-tuned on only 1% of the labels, we achieve 85.8% top-5 accuracy, outperforming AlexNet with 100× fewer labels. [1]

1. Introduction

Learning effective visual representations without human supervision is a long-standing problem. Most mainstream approaches fall into one of two classes: generative or discriminative. Generative approaches learn to generate or otherwise model pixels in the input space (Hinton et al., 2006; Kingma & Welling, 2013; Goodfellow et al., 2014).

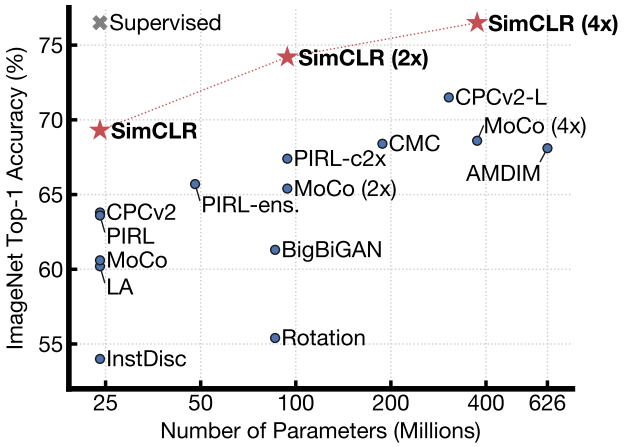


Figure 1. ImageNet Top-1 accuracy of linear classifiers trained on representations learned with different self-supervised methods (pretrained on ImageNet). Gray cross indicates supervised ResNet-50. Our method, SimCLR, is shown in bold.

However, pixel-level generation is computationally expensive and may not be necessary for representation learning. Discriminative approaches learn representations using objective functions similar to those used for supervised learning, but train networks to perform pretext tasks where both the inputs and labels are derived from an unlabeled dataset. Many such approaches have relied on heuristics to design pretext tasks (Doersch et al., 2015; Zhang et al., 2016; Noroozi & Favaro, 2016; Gidaris et al., 2018), which could limit the generality of the learned representations. Discriminative approaches based on contrastive learning in the latent space have recently shown great promise, achieving state-of-the-art results (Hadsell et al., 2006; Dosovitskiy et al., 2014; Oord et al., 2018; Bachman et al., 2019).

In this work, we introduce a simple framework for contrastive learning of visual representations, which we call *SimCLR*. Not only does SimCLR outperform previous work (Figure 1), but it is also simpler, requiring neither specialized architectures (Bachman et al., 2019; Hénaff et al., 2019) nor a memory bank (Wu et al., 2018; Tian et al., 2019; He et al., 2019; Misra & van der Maaten, 2019).

In order to understand what enables good contrastive representation learning, we systematically study the major components of our framework and show that:

[1]Google Research, Brain Team. Correspondence to: Ting Chen <iamtingchen@google.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

[1]Code available at https://github.com/google-research/simclr.

- Composition of multiple data augmentation operations is crucial in defining the contrastive prediction tasks that yield effective representations. In addition, unsupervised contrastive learning benefits from stronger data augmentation than supervised learning.
- Introducing a learnable nonlinear transformation between the representation and the contrastive loss substantially improves the quality of the learned representations.
- Representation learning with contrastive cross entropy loss benefits from normalized embeddings and an appropriately adjusted temperature parameter.
- Contrastive learning benefits from larger batch sizes and longer training compared to its supervised counterpart. Like supervised learning, contrastive learning benefits from deeper and wider networks.

We combine these findings to achieve a new state-of-the-art in self-supervised and semi-supervised learning on ImageNet ILSVRC-2012 (Russakovsky et al., 2015). Under the linear evaluation protocol, SimCLR achieves 76.5% top-1 accuracy, which is a 7% relative improvement over previous state-of-the-art (Hénaff et al., 2019). When fine-tuned with only 1% of the ImageNet labels, SimCLR achieves 85.8% top-5 accuracy, a relative improvement of 10% (Hénaff et al., 2019). When fine-tuned on other natural image classification datasets, SimCLR performs on par with or better than a strong supervised baseline (Kornblith et al., 2019) on 10 out of 12 datasets.

2. Method

2.1. The Contrastive Learning Framework

Inspired by recent contrastive learning algorithms (see Section 7 for an overview), SimCLR learns representations by maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space. As illustrated in Figure 2, this framework comprises the following four major components.

- A stochastic *data augmentation* module that transforms any given data example randomly resulting in two correlated views of the same example, denoted $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$, which we consider as a positive pair. In this work, we sequentially apply three simple augmentations: *random cropping* followed by resize back to the original size, *random color distortions*, and *random Gaussian blur*. As shown in Section 3, the combination of random crop and color distortion is crucial to achieve a good performance.
- A neural network *base encoder* $f(\cdot)$ that extracts representation vectors from augmented data examples. Our framework allows various choices of the network architecture without any constraints. We opt for simplicity and adopt the commonly used ResNet (He et al., 2016) to obtain $\boldsymbol{h}_i = f(\tilde{\boldsymbol{x}}_i) = \text{ResNet}(\tilde{\boldsymbol{x}}_i)$ where $\boldsymbol{h}_i \in \mathbb{R}^d$ is the output after the average pooling layer.
- A small neural network *projection head* $g(\cdot)$ that maps representations to the space where contrastive loss is applied. We use a MLP with one hidden layer to obtain $\boldsymbol{z}_i = g(\boldsymbol{h}_i) = W^{(2)}\sigma(W^{(1)}\boldsymbol{h}_i)$ where σ is a ReLU nonlinearity. As shown in section 4, we find it beneficial to define the contrastive loss on $\boldsymbol{z}_i$'s rather than $\boldsymbol{h}_i$'s.
- A *contrastive loss function* defined for a contrastive prediction task. Given a set $\{\tilde{\boldsymbol{x}}_k\}$ including a positive pair of examples $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$, the *contrastive prediction task* aims to identify $\tilde{\boldsymbol{x}}_j$ in $\{\tilde{\boldsymbol{x}}_k\}_{k\neq i}$ for a given $\tilde{\boldsymbol{x}}_i$.

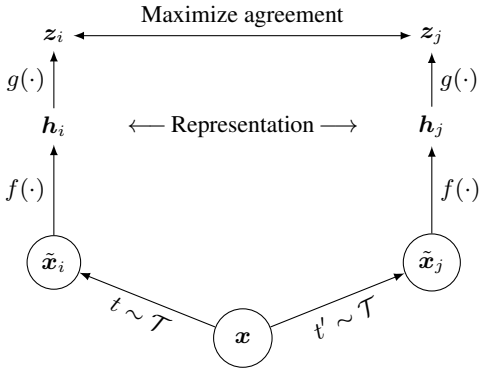


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.

We randomly sample a minibatch of N examples and define the contrastive prediction task on pairs of augmented examples derived from the minibatch, resulting in $2N$ data points. We do not sample negative examples explicitly. Instead, given a positive pair, similar to (Chen et al., 2017), we treat the other $2(N-1)$ augmented examples within a minibatch as negative examples. Let $\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v}/\|\boldsymbol{u}\|\|\boldsymbol{v}\|$ denote the dot product between ℓ_2 normalized $\boldsymbol{u}$ and $\boldsymbol{v}$ (i.e. cosine similarity). Then the loss function for a positive pair of examples (i, j) is defined as

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \quad (1)$$

where $\mathbb{1}_{[k\neq i]} \in \{0, 1\}$ is an indicator function evaluating to 1 iff $k \neq i$ and τ denotes a temperature parameter. The final loss is computed across all positive pairs, both (i, j) and (j, i), in a mini-batch. This loss has been used in previous work (Sohn, 2016; Wu et al., 2018; Oord et al., 2018); for convenience, we term it *NT-Xent* (the normalized temperature-scaled cross entropy loss).

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \dots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \dots, 2N\}$ and $j \in \{1, \dots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

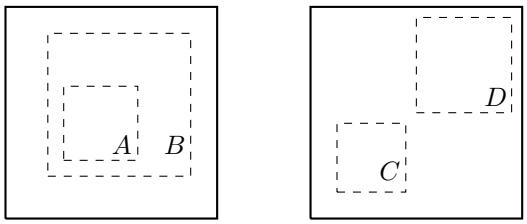
Algorithm 1 summarizes the proposed method.

2.2. Training with Large Batch Size

To keep it simple, we do not train the model with a memory bank (Wu et al., 2018; He et al., 2019). Instead, we vary the training batch size N from 256 to 8192. A batch size of 8192 gives us 16382 negative examples per positive pair from both augmentation views. Training with large batch size may be unstable when using standard SGD/Momentum with linear learning rate scaling (Goyal et al., 2017). To stabilize the training, we use the LARS optimizer (You et al., 2017) for all batch sizes. We train our model with Cloud TPUs, using 32 to 128 cores depending on the batch size.[2]

Global BN. Standard ResNets use batch normalization (Ioffe & Szegedy, 2015). In distributed training with data parallelism, the BN mean and variance are typically aggregated locally per device. In our contrastive learning, as positive pairs are computed in the same device, the model can exploit the local information leakage to improve prediction accuracy without improving representations. We address this issue by aggregating BN mean and variance over all devices during the training. Other approaches include shuffling data examples across devices (He et al., 2019), or replacing BN with layer norm (Hénaff et al., 2019).

[2]With 128 TPU v3 cores, it takes ∼1.5 hours to train our ResNet-50 with a batch size of 4096 for 100 epochs.



(a) Global and local views. (b) Adjacent views.

Figure 3. Solid rectangles are images, dashed rectangles are random crops. By randomly cropping images, we sample contrastive prediction tasks that include global to local view ($B \rightarrow A$) or adjacent view ($D \rightarrow C$) prediction.

2.3. Evaluation Protocol

Here we lay out the protocol for our empirical studies, which aim to understand different design choices in our framework.

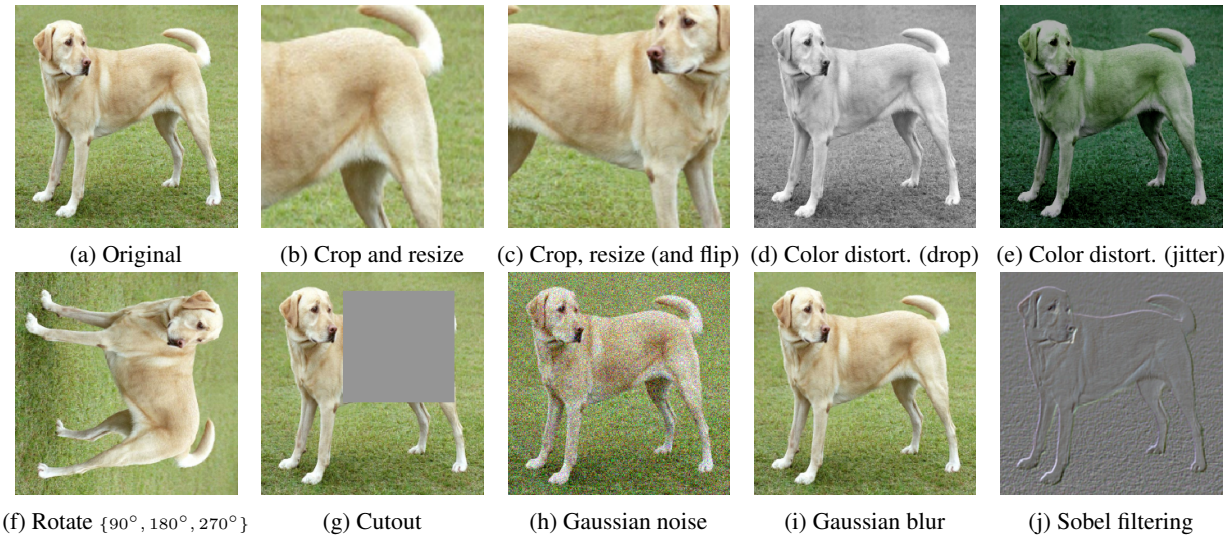
Dataset and Metrics. Most of our study for unsupervised pretraining (learning encoder network f without labels) is done using the ImageNet ILSVRC-2012 dataset (Russakovsky et al., 2015). Some additional pretraining experiments on CIFAR-10 (Krizhevsky & Hinton, 2009) can be found in Appendix B.9. We also test the pretrained results on a wide range of datasets for transfer learning. To evaluate the learned representations, we follow the widely used linear evaluation protocol (Zhang et al., 2016; Oord et al., 2018; Bachman et al., 2019; Kolesnikov et al., 2019), where a linear classifier is trained on top of the frozen base network, and test accuracy is used as a proxy for representation quality. Beyond linear evaluation, we also compare against state-of-the-art on semi-supervised and transfer learning.

Default setting. Unless otherwise specified, for data augmentation we use random crop and resize (with random flip), color distortions, and Gaussian blur (for details, see Appendix A). We use ResNet-50 as the base encoder network, and a 2-layer MLP projection head to project the representation to a 128-dimensional latent space. As the loss, we use NT-Xent, optimized using LARS with learning rate of 4.8 ($= 0.3 \times \text{BatchSize}/256$) and weight decay of 10^{-6}. We train at batch size 4096 for 100 epochs.[3] Furthermore, we use linear warmup for the first 10 epochs, and decay the learning rate with the cosine decay schedule without restarts (Loshchilov & Hutter, 2016).

3. Data Augmentation for Contrastive Representation Learning

Data augmentation defines predictive tasks. While data augmentation has been widely used in both supervised and unsupervised representation learning (Krizhevsky et al.,

[3]Although max performance is not reached in 100 epochs, reasonable results are achieved, allowing fair and efficient ablations.



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes *random crop (with flip and resize)*, *color distortion*, and *Gaussian blur*. (Original image cc-by: Von.grzanka)

2012; Hénaff et al., 2019; Bachman et al., 2019), it has not been considered as a systematic way to define the contrastive prediction task. Many existing approaches define contrastive prediction tasks by changing the architecture. For example, Hjelm et al. (2018); Bachman et al. (2019) achieve global-to-local view prediction via constraining the receptive field in the network architecture, whereas Oord et al. (2018); Hénaff et al. (2019) achieve neighboring view prediction via a fixed image splitting procedure and a context aggregation network. We show that this complexity can be avoided by performing simple *random cropping* (with resizing) of target images, which creates a family of predictive tasks subsuming the above mentioned two, as shown in Figure 3. This simple design choice conveniently decouples the predictive task from other components such as the neural network architecture. Broader contrastive prediction tasks can be defined by extending the family of augmentations and composing them stochastically.

3.1. Composition of data augmentation operations is crucial for learning good representations

To systematically study the impact of data augmentation, we consider several common augmentations here. One type of augmentation involves spatial/geometric transformation of data, such as cropping and resizing (with horizontal flipping), rotation (Gidaris et al., 2018) and cutout (DeVries & Taylor, 2017). The other type of augmentation involves appearance transformation, such as color distortion (including color dropping, brightness, contrast, saturation, hue) (Howard, 2013; Szegedy et al., 2015), Gaussian blur, and Sobel filtering. Figure 4 visualizes the augmentations that we study in this work.

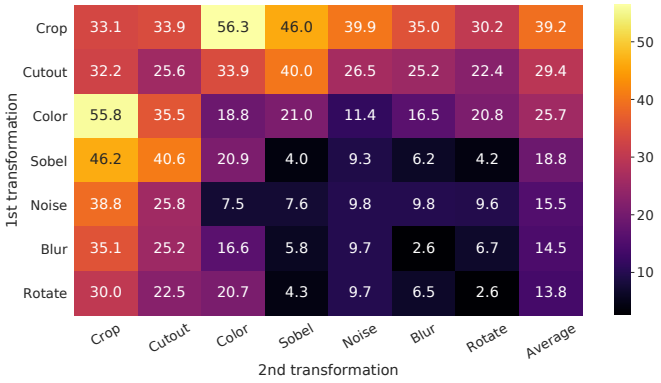
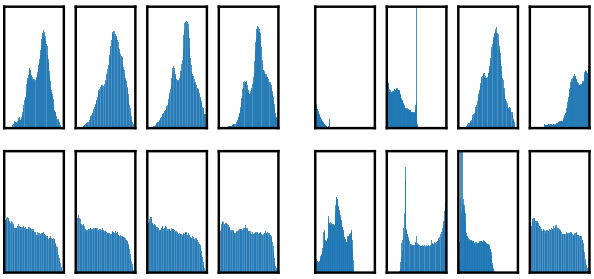


Figure 5. Linear evaluation (ImageNet top-1 accuracy) under individual or composition of data augmentations, applied only to one branch. For all columns but the last, diagonal entries correspond to single transformation, and off-diagonals correspond to composition of two transformations (applied sequentially). The last column reflects the average over the row.

To understand the effects of individual data augmentations and the importance of augmentation composition, we investigate the performance of our framework when applying augmentations individually or in pairs. Since ImageNet images are of different sizes, we always apply crop and resize images (Krizhevsky et al., 2012; Szegedy et al., 2015), which makes it difficult to study other augmentations in the absence of cropping. To eliminate this confound, we consider an asymmetric data transformation setting for this ablation. Specifically, we always first randomly crop images and resize them to the same resolution, and we then apply the targeted transformation(s) *only* to one branch of the framework in Figure 2, while leaving the other branch as the identity (i.e. $t(\boldsymbol{x}_i) = \boldsymbol{x}_i$). Note that this asymmet-



(a) Without color distortion. (b) With color distortion.

Figure 6. Histograms of pixel intensities (over all channels) for different crops of two different images (i.e. two rows). The image for the first row is from Figure 4. All axes have the same range.

Methods	Color distortion strength 1/8	1/4	1/2	1	1 (+Blur)	AutoAug
SimCLR	59.6	61.0	62.6	63.2	64.5	61.1
Supervised	77.0	76.7	76.5	75.7	75.4	77.1

Table 1. Top-1 accuracy of unsupervised ResNet-50 using linear evaluation and supervised ResNet-50[5], under varied color distortion strength (see Appendix A) and other data transformations. Strength 1 (+Blur) is our default data augmentation policy.

ric data augmentation hurts the performance. Nonetheless, this setup should not substantively change the impact of individual data augmentations or their compositions.

Figure 5 shows linear evaluation results under individual and composition of transformations. We observe that *no single transformation suffices to learn good representations*, even though the model can almost perfectly identify the positive pairs in the contrastive task. When composing augmentations, the contrastive prediction task becomes harder, but the quality of representation improves dramatically. Appendix B.2 provides a further study on composing broader set of augmentations.

One composition of augmentations stands out: random cropping and random color distortion. We conjecture that one serious issue when using only random cropping as data augmentation is that most patches from an image share a similar color distribution. Figure 6 shows that color histograms alone suffice to distinguish images. Neural nets may exploit this shortcut to solve the predictive task. Therefore, it is critical to compose cropping with color distortion in order to learn generalizable features.

3.2. Contrastive learning needs stronger data augmentation than supervised learning

To further demonstrate the importance of the color augmentation, we adjust the strength of color augmentation as

[5]Supervised models are trained for 90 epochs; longer training improves performance of stronger augmentation by $\sim 0.5\%$.

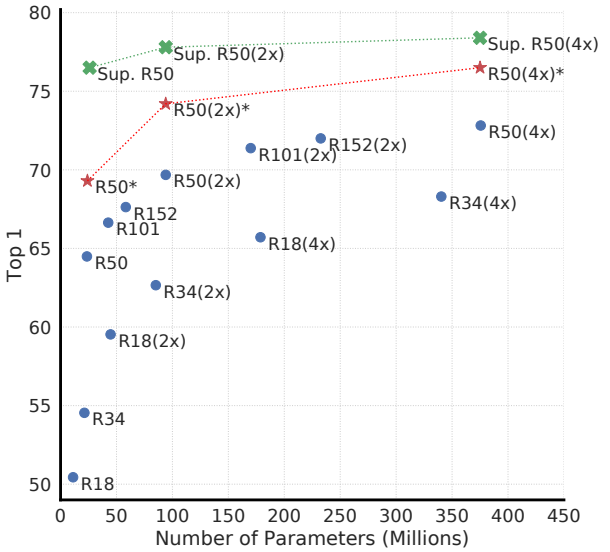


Figure 7. Linear evaluation of models with varied depth and width. Models in blue dots are ours trained for 100 epochs, models in red stars are ours trained for 1000 epochs, and models in green crosses are supervised ResNets trained for 90 epochs[7] (He et al., 2016).

shown in Table 1. Stronger color augmentation substantially improves the linear evaluation of the learned unsupervised models. In this context, AutoAugment (Cubuk et al., 2019), a sophisticated augmentation policy found using supervised learning, does not work better than simple cropping + (stronger) color distortion. When training supervised models with the same set of augmentations, we observe that stronger color augmentation does not improve or even hurts their performance. Thus, our experiments show that unsupervised contrastive learning benefits from stronger (color) data augmentation than supervised learning. Although previous work has reported that data augmentation is useful for self-supervised learning (Doersch et al., 2015; Bachman et al., 2019; Hénaff et al., 2019; Asano et al., 2019), we show that data augmentation that does not yield accuracy benefits for supervised learning can still help considerably with contrastive learning.

4. Architectures for Encoder and Head

4.1. Unsupervised contrastive learning benefits (more) from bigger models

Figure 7 shows, perhaps unsurprisingly, that increasing depth and width both improve performance. While similar findings hold for supervised learning (He et al., 2016), we find the gap between supervised models and linear classifiers trained on unsupervised models shrinks as the model size increases, suggesting that *unsupervised learning benefits more from bigger models than its supervised counterpart*.

[7]Training longer does not improve supervised ResNets (see Appendix B.3).

Name	Negative loss function	Gradient w.r.t. $\boldsymbol{u}$
NT-Xent	$\boldsymbol{u}^T\boldsymbol{v}^+/\tau - \log \sum_{\boldsymbol{v}\in\{\boldsymbol{v}^+,\boldsymbol{v}^-\}} \exp(\boldsymbol{u}^T\boldsymbol{v}/\tau)$	$(1 - \frac{\exp(\boldsymbol{u}^T\boldsymbol{v}^+/\tau)}{Z(\boldsymbol{u})})/\tau\boldsymbol{v}^+ - \sum_{\boldsymbol{v}^-} \frac{\exp(\boldsymbol{u}^T\boldsymbol{v}^-/\tau)}{Z(\boldsymbol{u})}/\tau\boldsymbol{v}^-$
NT-Logistic	$\log \sigma(\boldsymbol{u}^T\boldsymbol{v}^+/\tau) + \log \sigma(-\boldsymbol{u}^T\boldsymbol{v}^-/\tau)$	$(\sigma(-\boldsymbol{u}^T\boldsymbol{v}^+/\tau))/\tau\boldsymbol{v}^+ - \sigma(\boldsymbol{u}^T\boldsymbol{v}^-/\tau)/\tau\boldsymbol{v}^-$
Margin Triplet	$-\max(\boldsymbol{u}^T\boldsymbol{v}^- - \boldsymbol{u}^T\boldsymbol{v}^+ + m, 0)$	$\boldsymbol{v}^+ - \boldsymbol{v}^-$ if $\boldsymbol{u}^T\boldsymbol{v}^+ - \boldsymbol{u}^T\boldsymbol{v}^- < m$ else $\boldsymbol{0}$

Table 2. Negative loss functions and their gradients. All input vectors, i.e. $\boldsymbol{u}, \boldsymbol{v}^+, \boldsymbol{v}^-$, are ℓ_2 normalized. NT-Xent is an abbreviation for "Normalized Temperature-scaled Cross Entropy". Different loss functions impose different weightings of positive and negative examples.

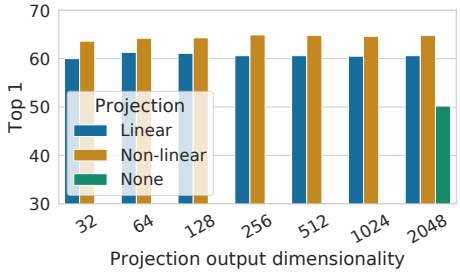


Figure 8. Linear evaluation of representations with different projection heads $g(\cdot)$ and various dimensions of $\boldsymbol{z} = g(\boldsymbol{h})$. The representation $\boldsymbol{h}$ (before projection) is 2048-dimensional here.

4.2. A nonlinear projection head improves the representation quality of the layer before it

We then study the importance of including a projection head, i.e. $g(\boldsymbol{h})$. Figure 8 shows linear evaluation results using three different architecture for the head: (1) identity mapping; (2) linear projection, as used by several previous approaches (Wu et al., 2018); and (3) the default nonlinear projection with one additional hidden layer (and ReLU activation), similar to Bachman et al. (2019). We observe that a nonlinear projection is better than a linear projection (+3%), and much better than no projection (>10%). When a projection head is used, similar results are observed regardless of output dimension. Furthermore, even when nonlinear projection is used, the layer before the projection head, $\boldsymbol{h}$, is still much better (>10%) than the layer after, $\boldsymbol{z} = g(\boldsymbol{h})$, which shows that *the hidden layer before the projection head is a better representation than the layer after*.

We conjecture that the importance of using the representation before the nonlinear projection is due to loss of information induced by the contrastive loss. In particular, $\boldsymbol{z} = g(\boldsymbol{h})$ is trained to be invariant to data transformation. Thus, g can remove information that may be useful for the downstream task, such as the color or orientation of objects. By leveraging the nonlinear transformation $g(\cdot)$, more information can be formed and maintained in $\boldsymbol{h}$. To verify this hypothesis, we conduct experiments that use either $\boldsymbol{h}$ or $g(\boldsymbol{h})$ to learn to predict the transformation applied during the pretraining. Here we set $g(h) = W^{(2)}\sigma(W^{(1)}h)$, with the same input and output dimensionality (i.e. 2048). Table 3 shows $\boldsymbol{h}$ contains much more information about the transformation applied, while $g(\boldsymbol{h})$ loses information. Further analysis can be found in Appendix B.4.

What to predict?	Random guess	Representation $\boldsymbol{h}$	Representation $g(\boldsymbol{h})$
Color vs grayscale	80	99.3	97.4
Rotation	25	67.6	25.6
Orig. vs corrupted	50	99.5	59.6
Orig. vs Sobel filtered	50	96.6	56.3

Table 3. Accuracy of training additional MLPs on different representations to predict the transformation applied. Other than crop and color augmentation, we additionally and independently add rotation (one of $\{0°, 90°, 180°, 270°\}$), Gaussian noise, and Sobel filtering transformation during the pretraining for the last three rows. Both $\boldsymbol{h}$ and $g(\boldsymbol{h})$ are of the same dimensionality, i.e. 2048.

5. Loss Functions and Batch Size

5.1. Normalized cross entropy loss with adjustable temperature works better than alternatives

We compare the NT-Xent loss against other commonly used contrastive loss functions, such as logistic loss (Mikolov et al., 2013), and margin loss (Schroff et al., 2015). Table 2 shows the objective function as well as the gradient to the input of the loss function. Looking at the gradient, we observe 1) ℓ_2 normalization (i.e. cosine similarity) along with temperature effectively weights different examples, and an appropriate temperature can help the model learn from hard negatives; and 2) unlike cross-entropy, other objective functions do not weigh the negatives by their relative hardness. As a result, one must apply semi-hard negative mining (Schroff et al., 2015) for these loss functions: instead of computing the gradient over all loss terms, one can compute the gradient using semi-hard negative terms (*i.e.*, those that are within the loss margin and closest in distance, but farther than positive examples).

To make the comparisons fair, we use the same ℓ_2 normalization for all loss functions, and we tune the hyperparameters, and report their best results.[8] Table 4 shows that, while (semi-hard) negative mining helps, the best result is still much worse than our default NT-Xent loss.

[8]Details can be found in Appendix B.10. For simplicity, we only consider the negatives from one augmentation view.

Margin	NT-Logi.	Margin (sh)	NT-Logi.(sh)	NT-Xent
50.9	51.6	57.5	57.9	63.9

Table 4. Linear evaluation (top-1) for models trained with different loss functions. "sh" means using semi-hard negative mining.

ℓ_2 norm?	τ	Entropy	Contrastive acc.	Top 1
Yes	0.05	1.0	90.5	59.7
	0.1	4.5	87.8	64.4
	0.5	8.2	68.2	60.7
	1	8.3	59.1	58.0
No	10	0.5	91.7	57.2
	100	0.5	92.1	57.0

Table 5. Linear evaluation for models trained with different choices of ℓ_2 norm and temperature τ for NT-Xent loss. The contrastive distribution is over 4096 examples.

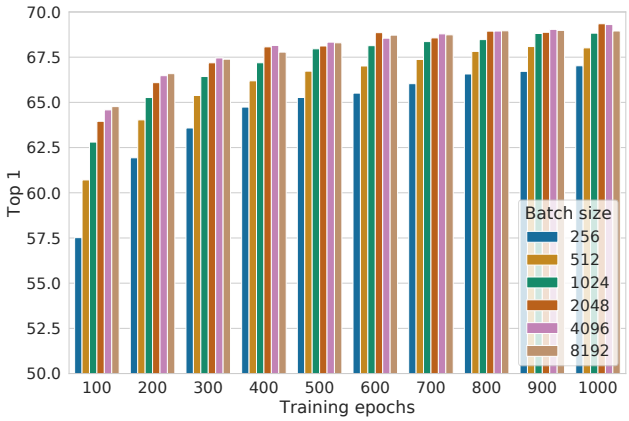


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

We next test the importance of the ℓ_2 normalization (i.e. cosine similarity vs dot product) and temperature τ in our default NT-Xent loss. Table 5 shows that without normalization and proper temperature scaling, performance is significantly worse. Without ℓ_2 normalization, the contrastive task accuracy is higher, but the resulting representation is worse under linear evaluation.

5.2. Contrastive learning benefits (more) from larger batch sizes and longer training

Figure 9 shows the impact of batch size when models are trained for different numbers of epochs. We find that, when the number of training epochs is small (e.g. 100 epochs), larger batch sizes have a significant advantage over the smaller ones. With more training steps/epochs, the gaps between different batch sizes decrease or disappear, provided the batches are randomly resampled. In contrast to supervised learning (Goyal et al., 2017), in contrastive learning, larger batch sizes provide more negative examples, facilitating convergence (i.e. taking fewer epochs and steps for a given accuracy). Training longer also provides more negative examples, improving the results. In Appendix B.1, results with even longer training steps are provided.

[10] A linear learning rate scaling is used here. Figure B.1 shows using a square root learning rate scaling can improve performance of ones with small batch sizes.

Method	Architecture	Param (M)	Top 1	Top 5
Methods using ResNet-50:				
Local Agg.	ResNet-50	24	60.2	-
MoCo	ResNet-50	24	60.6	-
PIRL	ResNet-50	24	63.6	-
CPC v2	ResNet-50	24	63.8	85.3
SimCLR (ours)	ResNet-50	24	**69.3**	**89.0**
Methods using other architectures:				
Rotation	RevNet-50 (4×)	86	55.4	-
BigBiGAN	RevNet-50 (4×)	86	61.3	81.9
AMDIM	Custom-ResNet	626	68.1	-
CMC	ResNet-50 (2×)	188	68.4	88.2
MoCo	ResNet-50 (4×)	375	68.6	-
CPC v2	ResNet-161 (∗)	305	71.5	90.1
SimCLR (ours)	ResNet-50 (2×)	94	74.2	92.0
SimCLR (ours)	ResNet-50 (4×)	375	**76.5**	**93.2**

Table 6. ImageNet accuracies of linear classifiers trained on representations learned with different self-supervised methods.

Method	Architecture	Label fraction 1%	Label fraction 10%
		Top 5	
Supervised baseline	ResNet-50	48.4	80.4
Methods using other label-propagation:			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	-	91.2
Methods using representation learning only:			
InstDisc	ResNet-50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161(∗)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet-50 (4×)	**85.8**	**92.6**

Table 7. ImageNet accuracy of models trained with few labels.

6. Comparison with State-of-the-art

In this subsection, similar to Kolesnikov et al. (2019); He et al. (2019), we use ResNet-50 in 3 different hidden layer widths (width multipliers of 1×, 2×, and 4×). For better convergence, our models here are trained for 1000 epochs.

Linear evaluation. Table 6 compares our results with previous approaches (Zhuang et al., 2019; He et al., 2019; Misra & van der Maaten, 2019; Hénaff et al., 2019; Kolesnikov et al., 2019; Donahue & Simonyan, 2019; Bachman et al.,

	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
SimCLR (ours)	**76.9**	**95.3**	80.2	48.4	**65.9**	60.0	61.2	**84.2**	**78.9**	89.2	**93.9**	**95.0**
Supervised	75.2	**95.7**	**81.2**	**56.4**	64.9	**68.8**	**63.8**	83.8	**78.7**	**92.3**	**94.1**	94.2
Fine-tuned:												
SimCLR (ours)	**89.4**	**98.6**	**89.0**	**78.2**	**68.1**	**92.1**	**87.0**	**86.6**	**77.8**	92.1	**94.1**	97.6
Supervised	88.7	98.3	**88.7**	**77.8**	67.0	91.4	**88.0**	86.5	**78.8**	**93.2**	**94.2**	**98.0**
Random init	88.3	96.0	81.9	**77.0**	53.7	91.3	84.8	69.4	64.1	82.7	72.5	92.5

Table 8. Comparison of transfer learning performance of our self-supervised approach with supervised baselines across 12 natural image classification datasets, for ResNet-50 $(4\times)$ models pretrained on ImageNet. Results not significantly worse than the best ($p > 0.05$, permutation test) are shown in bold. See Appendix B.8 for experimental details and results with standard ResNet-50.

2019; Tian et al., 2019) in the linear evaluation setting (see Appendix B.6). Table 1 shows more numerical comparisons among different methods. We are able to use standard networks to obtain substantially better results compared to previous methods that require specifically designed architectures. The best result obtained with our ResNet-50 $(4\times)$ can match the supervised pretrained ResNet-50.

Semi-supervised learning. We follow Zhai et al. (2019) and sample 1% or 10% of the labeled ILSVRC-12 training datasets in a class-balanced way (∼12.8 and ∼128 images per class respectively). [11] We simply fine-tune the whole base network on the labeled data without regularization (see Appendix B.5). Table 7 shows the comparisons of our results against recent methods (Zhai et al., 2019; Xie et al., 2019; Sohn et al., 2020; Wu et al., 2018; Donahue & Simonyan, 2019; Misra & van der Maaten, 2019; Hénaff et al., 2019). The supervised baseline from (Zhai et al., 2019) is strong due to intensive search of hyper-parameters (including augmentation). Again, our approach significantly improves over state-of-the-art with both 1% and 10% of the labels. Interestingly, fine-tuning our pretrained ResNet-50 $(2\times, 4\times)$ on *full* ImageNet are also significantly better then training from scratch (up to 2%, see Appendix B.2).

Transfer learning. We evaluate transfer learning performance across 12 natural image datasets in both linear evaluation (fixed feature extractor) and fine-tuning settings. Following Kornblith et al. (2019), we perform hyperparameter tuning for each model-dataset combination and select the best hyperparameters on a validation set. Table 8 shows results with the ResNet-50 $(4\times)$ model. When fine-tuned, our self-supervised model significantly outperforms the supervised baseline on 5 datasets, whereas the supervised baseline is superior on only 2 (i.e. Pets and Flowers). On the remaining 5 datasets, the models are statistically tied. Full experimental details as well as results with the standard ResNet-50 architecture are provided in Appendix B.8.

[11] The details of sampling and exact subsets can be found in https://www.tensorflow.org/datasets/catalog/imagenet2012_subset.

7. Related Work

The idea of making representations of an image agree with each other under small transformations dates back to Becker & Hinton (1992). We extend it by leveraging recent advances in data augmentation, network architecture and contrastive loss. A similar consistency idea, but for *class label prediction*, has been explored in other contexts such as semi-supervised learning (Xie et al., 2019; Berthelot et al., 2019).

Handcrafted pretext tasks. The recent renaissance of self-supervised learning began with artificially designed pretext tasks, such as relative patch prediction (Doersch et al., 2015), solving jigsaw puzzles (Noroozi & Favaro, 2016), colorization (Zhang et al., 2016) and rotation prediction (Gidaris et al., 2018; Chen et al., 2019). Although good results can be obtained with bigger networks and longer training (Kolesnikov et al., 2019), these pretext tasks rely on somewhat ad-hoc heuristics, which limits the generality of learned representations.

Contrastive visual representation learning. Dating back to Hadsell et al. (2006), these approaches learn representations by contrasting positive pairs against negative pairs. Along these lines, Dosovitskiy et al. (2014) proposes to treat each instance as a class represented by a feature vector (in a parametric form). Wu et al. (2018) proposes to use a memory bank to store the instance class representation vector, an approach adopted and extended in several recent papers (Zhuang et al., 2019; Tian et al., 2019; He et al., 2019; Misra & van der Maaten, 2019). Other work explores the use of in-batch samples for negative sampling instead of a memory bank (Doersch & Zisserman, 2017; Ye et al., 2019; Ji et al., 2019).

Recent literature has attempted to relate the success of their methods to maximization of mutual information between latent representations (Oord et al., 2018; Hénaff et al., 2019; Hjelm et al., 2018; Bachman et al., 2019). However, it is not clear if the success of contrastive approaches is determined by the mutual information, or by the specific form of the contrastive loss (Tschannen et al., 2019).

We note that almost all individual components of our framework have appeared in previous work, although the specific instantiations may be different. The superiority of our framework relative to previous work is not explained by any single design choice, but by their composition. We provide a comprehensive comparison of our design choices with those of previous work in Appendix C.

8. Conclusion

In this work, we present a simple framework and its instantiation for contrastive visual representation learning. We carefully study its components, and show the effects of different design choices. By combining our findings, we improve considerably over previous methods for self-supervised, semi-supervised, and transfer learning.

Our approach differs from standard supervised learning on ImageNet only in the choice of data augmentation, the use of a nonlinear head at the end of the network, and the loss function. The strength of this simple framework suggests that, despite a recent surge in interest, self-supervised learning remains undervalued.

Acknowledgements

We would like to thank Xiaohua Zhai, Rafael Müller and Yani Ioannou for their feedback on the draft. We are also grateful for general support from Google Research teams in Toronto and elsewhere.

References

Asano, Y. M., Rupprecht, C., and Vedaldi, A. A critical analysis of self-supervision, or what we can learn from a single image. *arXiv preprint arXiv:1904.13132*, 2019.

Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pp. 15509–15519, 2019.

Becker, S. and Hinton, G. E. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355 (6356):161–163, 1992.

Berg, T., Liu, J., Lee, S. W., Alexander, M. L., Jacobs, D. W., and Belhumeur, P. N. Birdsnap: Large-scale fine-grained visual categorization of birds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2019–2026. IEEE, 2014.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pp. 5050–5060, 2019.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Chen, T., Sun, Y., Shi, Y., and Hong, L. On sampling strategies for neural network-based collaborative filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 767–776, 2017.

Chen, T., Zhai, X., Ritter, M., Lucic, M., and Houlsby, N. Self-supervised gans via auxiliary rotation loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12154–12163, 2019.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3606–3613. IEEE, 2014.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 113–123, 2019.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Doersch, C. and Zisserman, A. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2051–2060, 2017.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430, 2015.

Donahue, J. and Simonyan, K. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, pp. 10541–10551, 2019.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, pp. 647–655, 2014.

Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in neural information processing systems*, pp. 766–774, 2014.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Generative-Model Based Vision*, 2004.

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Hénaff, O. J., Razavi, A., Doersch, C., Eslami, S., and Oord, A. v. d. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Howard, A. G. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Ji, X., Henriques, J. F., and Vedaldi, A. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9865–9874, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kolesnikov, A., Zhai, X., and Beyer, L. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1920–1929, 2019.

Kornblith, S., Shlens, J., and Le, Q. V. Do better ImageNet models transfer better? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2661–2671, 2019.

Krause, J., Deng, J., Stark, M., and Fei-Fei, L. Collecting a large-scale dataset of fine-grained cars. In *Second Workshop on Fine-Grained Visual Categorization*, 2013.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Maji, S., Kannala, J., Rahtu, E., Blaschko, M., and Vedaldi, A. Fine-grained visual classification of aircraft. Technical report, 2013.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Misra, I. and van der Maaten, L. Self-supervised learning of pretext-invariant representations. *arXiv preprint arXiv:1912.01991*, 2019.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pp. 722–729. IEEE, 2008.

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pp. 69–84. Springer, 2016.

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

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3498–3505. IEEE, 2012.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, pp. 1857–1865, 2016.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*, 2019.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3485–3492. IEEE, 2010.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*, 2019.

Ye, M., Zhang, X., Yuen, P. C., and Chang, S.-F. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6210–6219, 2019.

You, Y., Gitman, I., and Ginsburg, B. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. S4l: Self-supervised semi-supervised learning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Zhang, R., Isola, P., and Efros, A. A. Colorful image colorization. In *European conference on computer vision*, pp. 649–666. Springer, 2016.

Zhuang, C., Zhai, A. L., and Yamins, D. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6002–6012, 2019.

A. Data Augmentation Details

In our default pretraining setting (which is used to train our best models), we utilize random crop (with resize and random flip), random color distortion, and random Gaussian blur as the data augmentations. The details of these three augmentations are provided below.

Random crop and resize to 224x224 We use standard Inception-style random cropping (Szegedy et al., 2015). The crop of random size (uniform from 0.08 to 1.0 in area) of the original size and a random aspect ratio (default: of 3/4 to 4/3) of the original aspect ratio is made. This crop is finally resized to the original size. This has been implemented in Tensorflow as "slim.preprocessing.inception_preprocessing.distorted_bounding_box_crop", or in Pytorch as "torchvision.transforms.RandomResizedCrop". Additionally, the random crop (with resize) is always followed by a random horizontal/left-to-right flip with 50% probability. This is helpful but not essential. By removing this from our default augmentation policy, the top-1 linear evaluation drops from 64.5% to 63.4% for our ResNet-50 model trained in 100 epochs.

Color distortion Color distortion is composed by color jittering and color dropping. We find stronger color jittering usually helps, so we set a strength parameter.

A pseudo-code for color distortion using TensorFlow is as follows.

```
import tensorflow as tf
def color_distortion(image, s=1.0):
    # image is a tensor with value range in [0, 1].
    # s is the strength of color distortion.

    def color_jitter(x):
        # one can also shuffle the order of following augmentations
        # each time they are applied.
        x = tf.image.random_brightness(x, max_delta=0.8*s)
        x = tf.image.random_contrast(x, lower=1-0.8*s, upper=1+0.8*s)
        x = tf.image.random_saturation(x, lower=1-0.8*s, upper=1+0.8*s)
        x = tf.image.random_hue(x, max_delta=0.2*s)
        x = tf.clip_by_value(x, 0, 1)
        return x

    def color_drop(x):
        image = tf.image.rgb_to_grayscale(image)
        image = tf.tile(image, [1, 1, 3])

    # randomly apply transformation with probability p.
    image = random_apply(color_jitter, image, p=0.8)
    image = random_apply(color_drop, image, p=0.2)
    return image
```

A pseudo-code for color distortion using Pytorch is as follows [12].

```
from torchvision import transforms
def get_color_distortion(s=1.0):
    # s is the strength of color distortion.
    color_jitter = transforms.ColorJitter(0.8*s, 0.8*s, 0.8*s, 0.2*s)
    rnd_color_jitter = transforms.RandomApply([color_jitter], p=0.8)
    rnd_gray = transforms.RandomGrayscale(p=0.2)
    color_distort = transforms.Compose([
        rnd_color_jitter,
        rnd_gray])
```

[12] Our code and results are based on Tensorflow, the Pytorch code here is a reference.

```
    return color_distort
```

Gaussian blur This augmentation is in our default policy. We find it helpful, as it improves our ResNet-50 trained for 100 epochs from 63.2% to 64.5%. We blur the image 50% of the time using a Gaussian kernel. We randomly sample $\sigma \in [0.1, 2.0]$, and the kernel size is set to be 10% of the image height/width.

B. Additional Experimental Results

B.1. Batch Size and Training Steps

Figure B.1 shows the top-5 accuracy on linear evaluation when trained with different batch sizes and training epochs. The conclusion is very similar to top-1 accuracy shown before, except that the differences between different batch sizes and training steps seems slightly smaller here.

In both Figure 9 and Figure B.1, we use a linear scaling of learning rate similar to (Goyal et al., 2017) when training with different batch sizes. Although linear learning rate scaling is popular with SGD/Momentum optimizer, we find a square root learning rate scaling is more desirable with LARS optimizer. With square root learning rate scaling, we have $\text{LearningRate} = 0.075 \times \sqrt{\text{BatchSize}}$, instead of $\text{LearningRate} = 0.3 \times \text{BatchSize}/256$ in the linear scaling case, but the learning rate is the same under both scaling methods when batch size of 4096 (our default batch size). A comparison is presented in Table B.1, where we observe that square root learning rate scaling improves the performance for models trained with small batch sizes and in smaller number of epochs.

Batch size \ Epochs	100	200	400	800
256	57.5 / **62.8**	61.9 / **64.3**	64.7 / **65.7**	66.6 / 66.5
512	60.7 / **63.8**	64.0 / **65.6**	66.2 / 66.7	67.8 / 67.4
1024	62.8 / **64.3**	65.3 / **66.1**	67.2 / 67.2	68.5 / 68.3
2048	64.0 / **64.7**	66.1 / **66.8**	68.1 / 67.9	68.9 / 68.8
4096	64.6 / 64.5	66.5 / 66.8	68.2 / 68.0	68.9 / 69.1
8192	64.8 / 64.8	66.6 / 67.0	67.8 / 68.3	69.0 / 69.1

Table B.1. Linear evaluation (top-1) under different batch sizes and training epochs. On the left side of slash sign are models trained with linear LR scaling, and on the right are models trained with square root LR scaling. The result is bolded if it is more than 0.5% better. Square root LR scaling works better for smaller batch size trained in fewer epochs (with LARS optimizer).

We also train with larger batch size (up to 32K) and longer (up to 3200 epochs), with the square root learning rate scaling. A shown in Figure B.2, *the performance seems to saturate with a batch size of 8192, while training longer can still significantly improve the performance*.

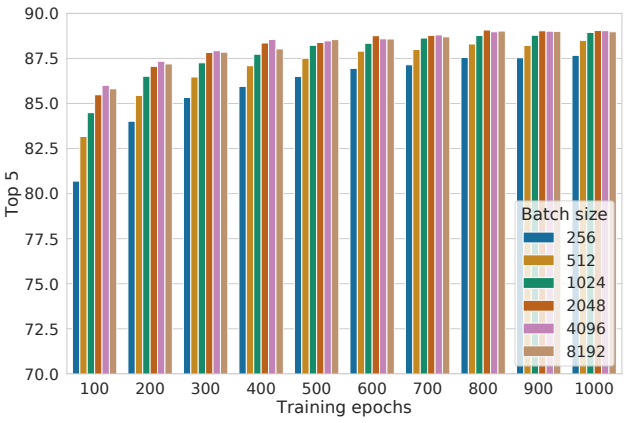


Figure B.1. Linear evaluation (top-5) of ResNet-50 trained with different batch sizes and epochs. Each bar is a single run from scratch. See Figure 9 for top-1 accuracy.

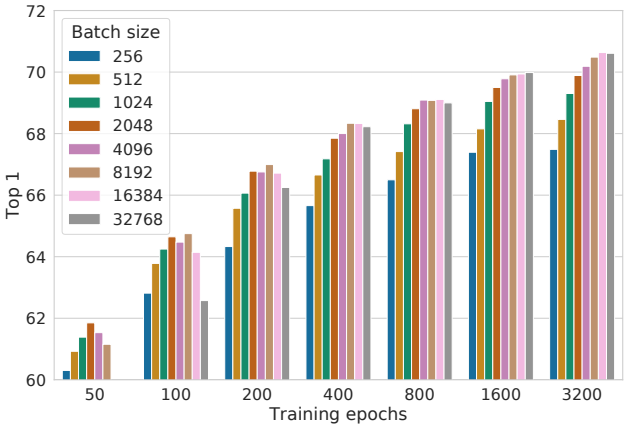


Figure B.2. Linear evaluation (top-1) of ResNet-50 trained with different batch sizes and *longer* epochs. Here a *square root* learning rate, instead of a linear one, is utilized.

B.2. Broader composition of data augmentations further improves performance

Our best results in the main text (Table 6 and 7) can be further improved when expanding the default augmentation policy to include the following: (1) Sobel filtering, (2) additional color distortion (equalize, solarize), and (3) motion blur. For linear evaluation protocol, the ResNet-50 models ($1\times, 2\times, 4\times$) trained with broader data augmentations achieve 70.0 (+0.7), 74.4 (+0.2), 76.8 (+0.3), respectively.

Table B.2 shows ImageNet accuracy obtained by fine-tuning the SimCLR model (see Appendix B.5 for the details of fine-tuning procedure). Interestingly, when fine-tuned on full (100%) ImageNet training set, our ResNet ($4\times$) model achieves 80.4% top-1 / 95.4% top-5 [13], which is significantly better than that (78.4% top-1 / 94.2% top-5) of training from scratch using the same set of augmentations (i.e. random crop and horizontal flip). For ResNet-50 ($2\times$), fine-tuning our pre-trained ResNet-50 ($2\times$) is also better than training from scratch (77.8% top-1 / 93.9% top-5). There is no improvement from fine-tuning for ResNet-50.

Architecture	Label fraction					
	1%		10%		100%	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
ResNet-50	49.4	76.6	66.1	88.1	76.0	93.1
ResNet-50 ($2\times$)	59.4	83.7	71.8	91.2	79.1	94.8
ResNet-50 ($4\times$)	64.1	86.6	74.8	92.8	80.4	95.4

Table B.2. Classification accuracy obtained by fine-tuning the SimCLR (which is pretrained with broader data augmentations) on 1%, 10% and full of ImageNet. As a reference, our ResNet-50 ($4\times$) trained from scratch on 100% labels achieves 78.4% top-1 / 94.2% top-5.

B.3. Effects of Longer Training for Supervised Models

Here we perform experiments to see how training steps and stronger data augmentation affect supervised training. We test ResNet-50 and ResNet-50 ($4\times$) under the same set of data augmentations (random crops, color distortion, 50% Gaussian blur) as used in our unsupervised models. Figure B.3 shows the top-1 accuracy. We observe that there is no significant benefit from training supervised models longer on ImageNet. Stronger data augmentation slightly improves the accuracy of ResNet-50 ($4\times$) but does not help on ResNet-50. When stronger data augmentation is applied, ResNet-50 generally requires longer training (e.g. 500 epochs [14]) to obtain the optimal result, while ResNet-50 ($4\times$) does not benefit from longer training.

Model	Training epochs	Top 1		
		Crop	+Color	+Color+Blur
ResNet-50	90	76.5	75.6	75.3
	500	76.2	76.5	76.7
	1000	75.8	75.2	76.4
ResNet-50 ($4\times$)	90	78.4	78.9	78.7
	500	78.3	78.4	78.5
	1000	77.9	78.2	78.3

Table B.3. Top-1 accuracy of supervised models trained longer under various data augmentation procedures (from the same set of data augmentations for contrastive learning).

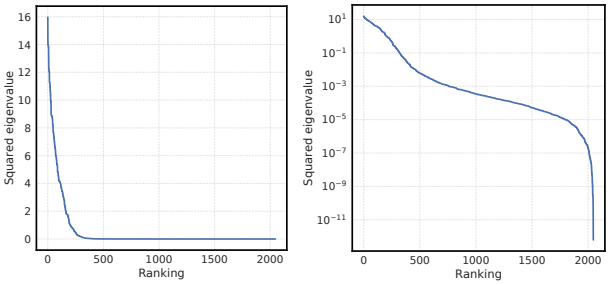
B.4. Understanding The Non-Linear Projection Head

Figure B.3 shows the eigenvalue distribution of linear projection matrix $W \in R^{2048\times2048}$ used to compute $\boldsymbol{z} = W\boldsymbol{h}$. This matrix has relatively few large eigenvalues, indicating that it is approximately low-rank.

Figure B.4 shows t-SNE (Maaten & Hinton, 2008) visualizations of $\boldsymbol{h}$ and $\boldsymbol{z} = g(\boldsymbol{h})$ for randomly selected 10 classes by our best ResNet-50 (top-1 linear evaluation 69.3%). Classes represented by $\boldsymbol{h}$ are better separated compared to $\boldsymbol{z}$.

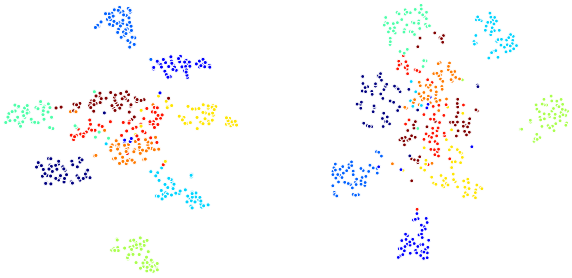
[13] It is 80.1% top-1 / 95.2% top-5 without broader augmentations for pretraining SimCLR.

[14] With AutoAugment (Cubuk et al., 2019), optimal test accuracy can be achieved between 900 and 500 epochs.



(a) Y-axis in uniform scale. (b) Y-axis in log scale.

Figure B.3. Squared real eigenvalue distribution of linear projection matrix $W \in R^{2048\times2048}$ used to compute $g(\boldsymbol{h}) = W\boldsymbol{h}$.



(a) $\boldsymbol{h}$ (b) $\boldsymbol{z} = g(\boldsymbol{h})$

Figure B.4. t-SNE visualizations of hidden vectors of images from a randomly selected 10 classes in the validation set.

B.5. Semi-supervised Learning via Fine-Tuning

Fine-tuning Procedure We fine-tune using the Nesterov momentum optimizer with a batch size of 4096, momentum of 0.9, and a learning rate of 0.8 (following $\text{LearningRate} = 0.05 \times \text{BatchSize}/256$) without warmup. Only random cropping (with random left-to-right flipping and resizing to 224x224) is used for preprocessing. We do not use any regularization (including weight decay). For 1% labeled data we fine-tune for 60 epochs, and for 10% labeled data we fine-tune for 30 epochs. For the inference, we resize the given image to 256x256, and take a single center crop of 224x224.

Table B.4 shows the comparisons of top-1 accuracy for different methods for semi-supervised learning. Our models significantly improve state-of-the-art.

Method	Architecture	Label fraction (Top 1) 1%	10%
Supervised baseline	ResNet-50	25.4	56.4
Methods using label-propagation:			
UDA (w. RandAug)	ResNet-50	-	68.8
FixMatch (w. RandAug)	ResNet-50	-	71.5
S4L (Rot+VAT+Ent. Min.)	ResNet-50 (4×)	-	73.2
Methods using self-supervised representation learning only:			
CPC v2	ResNet-161(∗)	52.7	73.1
SimCLR (ours)	ResNet-50	48.3	65.6
SimCLR (ours)	ResNet-50 (2×)	58.5	71.7
SimCLR (ours)	ResNet-50 (4×)	**63.0**	**74.4**

Table B.4. ImageNet top-1 accuracy of models trained with few labels. See Table 7 for top-5 accuracy.

B.6. Linear Evaluation

For linear evaluation, we follow similar procedure as fine-tuning (described in Appendix B.5), except that a larger learning rate of 1.6 (following $\text{LearningRate} = 0.1 \times \text{BatchSize}/256$) and longer training of 90 epochs. Alternatively, using LARS optimizer with the pretraining hyper-parameters also yield similar results. Furthermore, we find that attaching the linear classifier on top of the base encoder (with a stop_gradient on the input to linear classifier to prevent the label information from influencing the encoder) and train them simultaneously during the pretraining achieves similar performance.

B.7. Correlation Between Linear Evaluation and Fine-Tuning

Here we study the correlation between linear evaluation and fine-tuning under different settings of training step and network architecture.

Figure B.5 shows linear evaluation versus fine-tuning when training epochs of a ResNet-50 (using batch size of 4096) are varied from 50 to 3200 as in Figure B.2. While they are almost linearly correlated, it seems fine-tuning on a small fraction

of labels benefits more from training longer.

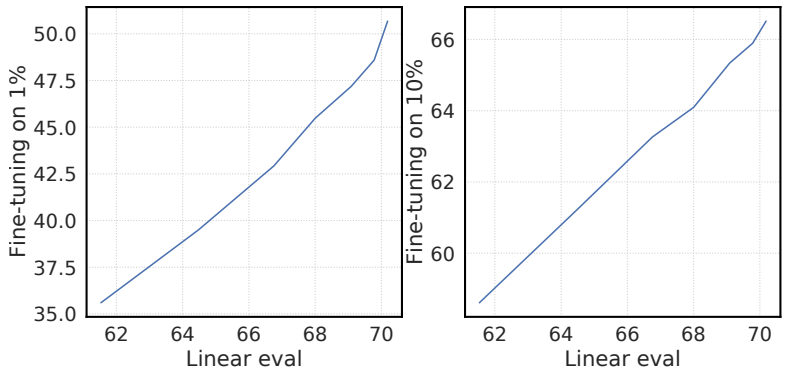


Figure B.5. Top-1 accuracy of models trained in different epochs (from Figure B.2), under linear evaluation and fine-tuning.

Figure B.6 shows shows linear evaluation versus fine-tuning for different architectures of choice.

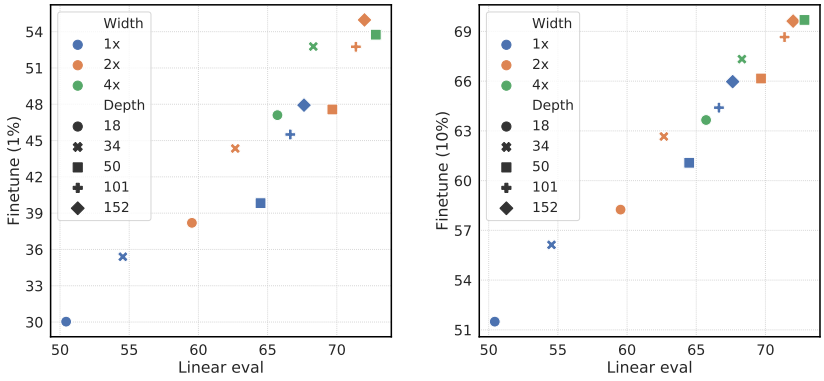


Figure B.6. Top-1 accuracy of different architectures under linear evaluation and fine-tuning.

B.8. Transfer Learning

We evaluated the performance of our self-supervised representation for transfer learning in two settings: linear evaluation, where a logistic regression classifier is trained to classify a new dataset based on the self-supervised representation learned on ImageNet, and fine-tuning, where we allow all weights to vary during training. In both cases, we follow the approach described by Kornblith et al. (2019), although our preprocessing differs slightly.

B.8.1. Methods

Datasets We investigated transfer learning performance on the Food-101 dataset (Bossard et al., 2014), CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009), Birdsnap (Berg et al., 2014), the SUN397 scene dataset (Xiao et al., 2010), Stanford Cars (Krause et al., 2013), FGVC Aircraft (Maji et al., 2013), the PASCAL VOC 2007 classification task (Everingham et al., 2010), the Describable Textures Dataset (DTD) (Cimpoi et al., 2014), Oxford-IIIT Pets (Parkhi et al., 2012), Caltech-101 (Fei-Fei et al., 2004), and Oxford 102 Flowers (Nilsback & Zisserman, 2008). We follow the evaluation protocols in the papers introducing these datasets, *i.e.*, we report top-1 accuracy for Food-101, CIFAR-10, CIFAR-100, Birdsnap, SUN397, Stanford Cars, and DTD; mean per-class accuracy for FGVC Aircraft, Oxford-IIIT Pets, Caltech-101, and Oxford 102 Flowers; and the 11-point mAP metric as defined in Everingham et al. (2010) for PASCAL VOC 2007. For DTD and SUN397, the dataset creators defined multiple train/test splits; we report results only for the first split. Caltech-101 defines no train/test split, so we randomly chose 30 images per class and test on the remainder, for fair comparison with previous work (Donahue et al., 2014; Simonyan & Zisserman, 2014).

We used the validation sets specified by the dataset creators to select hyperparameters for FGVC Aircraft, PASCAL VOC

2007, DTD, and Oxford 102 Flowers. For other datasets, we held out a subset of the training set for validation while performing hyperparameter tuning. After selecting the optimal hyperparameters on the validation set, we retrained the model using the selected parameters using all training and validation images. We report accuracy on the test set.

Transfer Learning via a Linear Classifier We trained an ℓ_2-regularized multinomial logistic regression classifier on features extracted from the frozen pretrained network. We used L-BFGS to optimize the softmax cross-entropy objective and we did not apply data augmentation. As preprocessing, all images were resized to 224 pixels along the shorter side using bicubic resampling, after which we took a 224×224 center crop. We selected the ℓ_2 regularization parameter from a range of 45 logarithmically spaced values between 10^{-6} and 10^{5}.

Transfer Learning via Fine-Tuning We fine-tuned the entire network using the weights of the pretrained network as initialization. We trained for 20,000 steps at a batch size of 256 using SGD with Nesterov momentum with a momentum parameter of 0.9. We set the momentum parameter for the batch normalization statistics to $\max(1 - 10/s, 0.9)$ where s is the number of steps per epoch. As data augmentation during fine-tuning, we performed only random crops with resize and flips; in contrast to pretraining, we did not perform color augmentation or blurring. At test time, we resized images to 256 pixels along the shorter side and took a 224×224 center crop. (Additional accuracy improvements may be possible with further optimization of data augmentation, particularly on the CIFAR-10 and CIFAR-100 datasets.) We selected the learning rate and weight decay, with a grid of 7 logarithmically spaced learning rates between 0.0001 and 0.1 and 7 logarithmically spaced values of weight decay between 10^{-6} and 10^{-3}, as well as no weight decay. We divide these values of weight decay by the learning rate.

Training from Random Initialization We trained the network from random initialization using the same procedure as for fine-tuning, but for longer, and with an altered hyperparameter grid. We chose hyperparameters from a grid of 7 logarithmically spaced learning rates between 0.001 and 1.0 and 8 logarithmically spaced values of weight decay between 10^{-5} and $10^{-1.5}$. Importantly, our random initialization baselines are trained for 40,000 steps, which is sufficiently long to achieve near-maximal accuracy, as demonstrated in Figure 8 of Kornblith et al. (2019).

On Birdsnap, there are no statistically significant differences among methods, and on Food-101, Stanford Cars, and FGVC Aircraft datasets, fine-tuning provides only a small advantage over training from random initialization. However, on the remaining 8 datasets, pretraining has clear advantages.

Supervised Baselines We compare against architecturally identical ResNet models trained on ImageNet with standard cross-entropy loss. These models are trained with the same data augmentation as our self-supervised models (crops, strong color augmentation, and blur) and are also trained for 1000 epochs. We found that, although stronger data augmentation and longer training time do not benefit accuracy on ImageNet, these models performed significantly better than a supervised baseline trained for 90 epochs and ordinary data augmentation for linear evaluation on a subset of transfer datasets. The supervised ResNet-50 baseline achieves 76.3% top-1 accuracy on ImageNet, vs. 69.3% for the self-supervised counterpart, while the ResNet-50 ($4\times$) baseline achieves 78.3%, vs. 76.5% for the self-supervised model.

Statistical Significance Testing We test for the significance of differences between model with a permutation test. Given predictions of two models, we generate 100,000 samples from the null distribution by randomly exchanging predictions for each example and computing the difference in accuracy after performing this randomization. We then compute the percentage of samples from the null distribution that are more extreme than the observed difference in predictions. For top-1 accuracy, this procedure yields the same result as the exact McNemar test. The assumption of exchangeability under the null hypothesis is also valid for mean per-class accuracy, but not when computing average precision curves. Thus, we perform significance testing for a difference in accuracy on VOC 2007 rather than a difference in mAP. A caveat of this procedure is that it does not consider run-to-run variability when training the models, only variability arising from using a finite sample of images for evaluation.

B.8.2. Results with Standard ResNet

The ResNet-50 ($4\times$) results shown in Table 8 of the text show no clear advantage to the supervised or self-supervised models. With the narrower ResNet-50 architecture, however, supervised learning maintains a clear advantage over self-supervised learning. The supervised ResNet-50 model outperforms the self-supervised model on all datasets with linear evaluation, and most (10 of 12) datasets with fine-tuning. The weaker performance of the ResNet model compared to the ResNet ($4\times$)

	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
SimCLR (ours)	68.4	90.6	71.6	37.4	58.8	50.3	50.3	80.5	**74.5**	83.6	90.3	91.2
Supervised	**72.3**	**93.6**	**78.3**	**53.7**	**61.9**	**66.7**	**61.0**	**82.8**	74.9	**91.5**	**94.5**	**94.7**
Fine-tuned:												
SimCLR (ours)	**88.2**	**97.7**	**85.9**	**75.9**	63.5	91.3	**88.1**	84.1	**73.2**	89.2	92.1	97.0
Supervised	**88.3**	**97.5**	**86.4**	**75.8**	**64.3**	**92.1**	86.0	**85.0**	**74.6**	**92.1**	**93.3**	**97.6**
Random init	86.9	95.9	80.2	**76.1**	53.6	91.4	85.9	67.3	64.8	81.5	72.6	92.0

Table B.5. Comparison of transfer learning performance of our self-supervised approach with supervised baselines across 12 natural image datasets, using ImageNet-pretrained ResNet models. See also Figure 8 for results with the ResNet $(4\times)$ architecture.

model may relate to the accuracy gap between the supervised and self-supervised models on ImageNet. The self-supervised ResNet gets 69.3% top-1 accuracy, 6.8% worse than the supervised model in absolute terms, whereas the self-supervised ResNet $(4\times)$ model gets 76.5%, which is only 1.8% worse than the supervised model.

B.9. CIFAR-10

While we focus on using ImageNet as the main dataset for pretraining our unsupervised model, our method also works with other datasets. We demonstrate it by testing on CIFAR-10 as follows.

Setup As our goal is not to optimize CIFAR-10 performance, but rather to provide further confirmation of our observations on ImageNet, we use the same architecture (ResNet-50) for CIFAR-10 experiments. Because CIFAR-10 images are much smaller than ImageNet images, we replace the first 7x7 Conv of stride 2 with 3x3 Conv of stride 1, and also remove the first max pooling operation. For data augmentation, we use the same Inception crop (flip and resize to 32x32) as ImageNet,[15] and color distortion (strength=0.5), leaving out Gaussian blur. We pretrain with learning rate in $\{0.5, 1.0, 1.5\}$, temperature in $\{0.1, 0.5, 1.0\}$, and batch size in $\{256, 512, 1024, 2048, 4096\}$. The rest of the settings (including optimizer, weight decay, etc.) are the same as our ImageNet training.

Our best model trained with batch size 1024 can achieve a linear evaluation accuracy of 94.0%, compared to 95.1% from the supervised baseline using the same architecture and batch size. The best self-supervised model that reports linear evaluation result on CIFAR-10 is AMDIM (Bachman et al., 2019), which achieves 91.2% with a model $25\times$ larger than ours. We note that our model can be improved by incorporating extra data augmentations as well as using a more suitable base network.

Performance under different batch sizes and training steps Figure B.7 shows the linear evaluation performance under different batch sizes and training steps. The results are consistent with our observations on ImageNet, although the largest batch size of 4096 seems to cause a small degradation in performance on CIFAR-10.

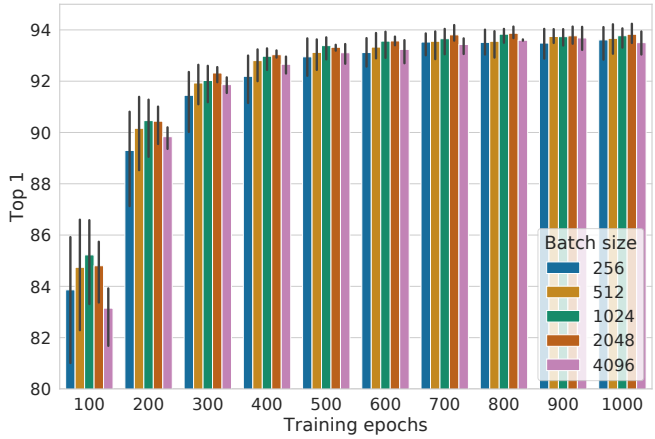
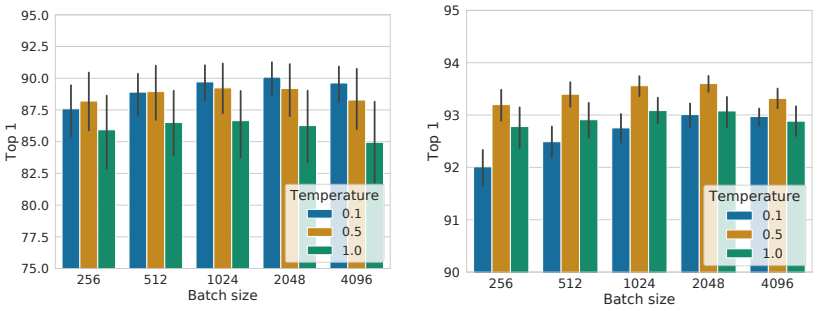


Figure B.7. Linear evaluation of ResNet-50 (with adjusted stem) trained with different batch size and epochs on CIFAR-10 dataset. Each bar is averaged over 3 runs with different learning rates (0.5, 1.0, 1.5) and temperature $\tau = 0.5$. Error bar denotes standard deviation.

[15] It is worth noting that, although CIFAR-10 images are much smaller than ImageNet images and image size does not differ among examples, cropping with resizing is still a very effective augmentation for contrastive learning.

Optimal temperature under different batch sizes Figure B.8 shows the linear evaluation of model trained with three different temperatures under various batch sizes. We find that when training to convergence (e.g. training epochs > 300), the optimal temperature in $\{0.1, 0.5, 1.0\}$ is 0.5 and seems consistent regardless of the batch sizes. However, the performance with $\tau = 0.1$ improves as batch size increases, which may suggest a small shift of optimal temperature towards 0.1.



(a) Training epochs ≤ 300 (b) Training epochs > 300

Figure B.8. Linear evaluation of the model (ResNet-50) trained with three temperatures on different batch sizes on CIFAR-10. Each bar is averaged over multiple runs with different learning rates and total train epochs. Error bar denotes standard deviation.

B.10. Tuning For Other Loss Functions

The learning rate that works best for NT-Xent loss may not be a good learning rate for other loss functions. To ensure a fair comparison, we also tune hyperparameters for both margin loss and logistic loss. Specifically, we tune learning rate in $\{0.01, 0.1, 0.3, 0.5, 1.0\}$ for both loss functions. We further tune the margin in $\{0, 0.4, 0.8, 1.6\}$ for margin loss, the temperature in $\{0.1, 0.2, 0.5, 1.0\}$ for logistic loss. For simplicity, we only consider the negatives from one augmentation view (instead of both sides), which slightly impairs performance but ensures fair comparison.

C. Further Comparison to Related Methods

As we have noted in the main text, most individual components of SimCLR have appeared in previous work, and the improved performance is a result of a combination of these design choices. Table C.1 provides a high-level comparison of the design choices of our method with those of previous methods. Compared with previous work, our design choices are generally simpler.

Model	Data Augmentation	Base Encoder	Projection Head	Loss	Batch Size	Train Epochs
CPC v2	Custom	ResNet-161 (modified)	PixelCNN	Xent	$512^{\#}$	$\sim$200
AMDIM	Fast AutoAug.	Custom ResNet	Non-linear MLP	Xent w/ clip,reg	$1008^{\#}$	150
CMC	Fast AutoAug.	ResNet-50 (2×, L+ab)	Linear layer	Xent w/ ℓ_2, τ	156^{*}	280
MoCo	Crop+color	ResNet-50 (4×)	Linear layer	Xent w/ ℓ_2, τ	256^{*}	200
PIRL	Crop+color	ResNet-50 (2×)	Linear layer	Xent w/ ℓ_2, τ	1024^{*}	800
SimCLR	Crop+color+blur	ResNet-50 (4×)	Non-linear MLP	Xent w/ ℓ_2, τ	4096	1000

Table C.1. A high-level comparison of design choices and training setup (for best result on ImageNet) for each method. Note that descriptions provided here are general; even when they match for two methods, formulations and implementations may differ (e.g. for color augmentation). Refer to the original papers for more details. $^{\#}$Examples are split into multiple patches, which enlarges the effective batch size. *A memory bank is employed.

In below, we provide an in-depth comparison of our method to the recently proposed contrastive representation learning methods:

- DIM/AMDIM (Hjelm et al., 2018; Bachman et al., 2019) achieve global-to-local/local-to-neighbor prediction by predicting the middle layer of ConvNet. The ConvNet is a ResNet that has bewen modified to place significant constraints on the receptive fields of the network (e.g. replacing many 3x3 Convs with 1x1 Convs). In our framework, we decouple the prediction task and encoder architecture, by random cropping (with resizing) and using the final

representations of two augmented views for prediction, so we can use standard and more powerful ResNets. Our NT-Xent loss function leverages normalization and temperature to restrict the range of similarity scores, whereas they use a tanh function with regularization. We use a simpler data augmentation policy, while they use FastAutoAugment for their best result.

- CPC v1 and v2 (Oord et al., 2018; Hénaff et al., 2019) define the context prediction task using a deterministic strategy to split examples into patches, and a context aggregation network (a PixelCNN) to aggregate these patches. The base encoder network sees only patches, which are considerably smaller than the original image. We decouple the prediction task and the encoder architecture, so we do not require a context aggregation network, and our encoder can look at the images of wider spectrum of resolutions. In addition, we use the NT-Xent loss function, which leverages normalization and temperature, whereas they use an unnormalized cross-entropy-based objective. We use simpler data augmentation.

- InstDisc, MoCo, PIRL (Wu et al., 2018; He et al., 2019; Misra & van der Maaten, 2019) generalize the Exemplar approach originally proposed by Dosovitskiy et al. (2014) and leverage an explicit memory bank. We do not use a memory bank; we find that, with a larger batch size, in-batch negative example sampling suffices. We also utilize a nonlinear projection head, and use the representation before the projection head. Although we use similar types of augmentations (e.g., random crop and color distortion), we expect specific parameters may be different.

- CMC (Tian et al., 2019) uses a separated network for each view, while we simply use a single network shared for all randomly augmented views. The data augmentation, projection head and loss function are also different. We use larger batch size instead of a memory bank.

- Whereas Ye et al. (2019) maximize similarity between augmented and unaugmented copies of the same image, we apply data augmentation symmetrically to both branches of our framework (Figure 2). We also apply a nonlinear projection on the output of base feature network, and use the representation before projection network, whereas Ye et al. (2019) use the linearly projected final hidden vector as the representation. When training with large batch sizes using multiple accelerators, we use global BN to avoid shortcuts that can greatly decrease representation quality.