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 results.

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. 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).

---

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


1Code available at https://github.com/google-research/simclr.
A Simple Framework for Contrastive Learning of Visual Representations

- 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 2019). When fine-tuned with only 1% of the ImageNet labels, SimCLR achieves 85.8% top-5 accuracy, which is a 7% relative improvement over previous state-of-the-art (Hénaff et al., 2019). When fine-tuned with other natural image classification datasets, SimCLR performs on par with or better than 2019). When fine-tuned with only 1% of the ImageNet labels, SimCLR achieves 85.8% top-5 accuracy, which is a 7% relative improvement over previous state-of-the-art.

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 $x_i$ and $x_j$, 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 $h_i = f(x_i) = \text{ResNet}(\tilde{x}_i)$ where $h_i \in \mathbb{R}^d$ 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 $z_i = g(h_i) = W(1)\sigma(W(0)h_i)$ where $\sigma$ is a ReLU non-linearity. As shown in section 4, we find it beneficial to define the contrastive loss on $z_i$’s rather than $h_i$’s.
- A contrastive loss function defined for a contrastive prediction task. Given a set $\{x_k\}$ including a positive pair of examples $x_i$ and $x_j$, the contrastive prediction task aims to identify $\tilde{x}_j$ in $\{\tilde{x}_k\}_{k \neq i}$ for a given $x_i$.

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 $\text{sim}(u, v) = u^\top v / \|u\|\|v\|$ denote the dot product between $\ell_2$ normalized $u$ and $v$ (i.e. cosine similarity). Then the loss function for a positive pair of examples $(i, j)$ is defined as

$$\ell_{i,j} = - \log \frac{\exp(\text{sim}(z_i, z_j) / \tau)}{\sum_{k=1}^{2N} I_{[k \neq i]} \exp(\text{sim}(z_i, z_k) / \tau)}, \quad (1)$$

where $I_{[k \neq i]} \in \{0, 1\}$ is an indicator function evaluating to 1 iff $k \neq i$ and $\tau$ 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).

Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are applied. We use a MLP with one hidden layer to obtain $z_i = g(h_i) = W(1)\sigma(W(0)h_i)$ where $\sigma$ is a ReLU non-linearity. As shown in section 4, we find it beneficial to define the contrastive loss on $z_i$’s rather than $h_i$’s.
Algorithm 1 summarizes the proposed method.

**Algorithm 1** SimCLR’s main learning algorithm.

```plaintext
input: batch size \( N \), constant \( \tau \), structure of \( f, g, T \).
for sampled minibatch \( \{x_k\}_{k=1}^{N} \) do
  for all \( k \in \{1, \ldots, N\} \) do
    draw two augmentation functions \( t \sim T, t' \sim T \)
    # the first augmentation
    \( x_{2k-1} = t(x_k) \)
    \( h_{2k-1} = f(x_{2k-1}) \)  # representation
    \( z_{2k-1} = g(h_{2k-1}) \)  # projection
  end for
  for all \( i \in \{1, \ldots, 2N\} \) and \( j \in \{1, \ldots, 2N\} \) do
    \( s_{i,j} = z_i^T z_j / (\|z_i\| \|z_j\|) \)  # pairwise similarity
  end for
  define \( \ell(i, j) \) as \( \ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \exp(s_{i,k}/\tau)} \)
  \( \mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)] \)
  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 (Hénaiff et al., 2019).

2With 128 TPU v3 cores, it takes \(~1.5\) hours to train our ResNet-50 with a batch size of 4096 for 100 epochs.

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).

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

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.,...
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 flip and resize), color distortion, and Gaussian blur. 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 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)

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. \( t(x_i) = x_i \)). Note that this asymmet-
A Simple Framework for Contrastive Learning of Visual Representations

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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Color distortion strength</th>
<th>AutoAug</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimCLR</td>
<td>1/8</td>
<td>77.0</td>
</tr>
<tr>
<td>Supervised</td>
<td>1/4</td>
<td>76.7</td>
</tr>
<tr>
<td></td>
<td>1/2</td>
<td>76.5</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>75.7</td>
</tr>
<tr>
<td></td>
<td>1 (+Blur)</td>
<td>75.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61.1</td>
</tr>
</tbody>
</table>

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

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 (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.

---

3Supervised models are trained for 90 epochs; longer training improves performance of stronger augmentation by \( \sim 0.5\% \).

7Training longer does not improve supervised ResNets (see Appendix B.3).
Negative loss functions and their gradients. All input vectors, i.e. \( \mathbf{u}, \mathbf{v}^+, \mathbf{v}^- \), are \( \ell_2 \) normalized. NT-Xent is an abbreviation for “Normalized Temperature-scaled Cross-Entropy”. Different loss functions impose different weightings of positive and negative examples.

### Table 2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Negative loss function</th>
<th>Gradient w.r.t. ( \mathbf{u} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT-Xent</td>
<td>( \mathbf{u}^T \mathbf{v}^+/\tau - \log \sum_{\mathbf{v} \in { \mathbf{v}^+, \mathbf{v}^- }} \exp(\mathbf{u}^T \mathbf{v}/\tau) )</td>
<td>( (1 - \frac{\exp(\mathbf{u}^T \mathbf{v}^+/\tau)}{Z(\mathbf{u})})/\tau \mathbf{v}^+ - \sum_{\mathbf{v}} \frac{\exp(\mathbf{u}^T \mathbf{w}^-/\tau)}{Z(\mathbf{u})} \mathbf{v}^- )</td>
</tr>
<tr>
<td>NT-Logistic</td>
<td>( \log \sigma(\mathbf{u}^T \mathbf{v}^+/\tau) + \log \sigma(\mathbf{u}^T \mathbf{v}^-/\tau) )</td>
<td>( (\sigma(\mathbf{u}^T \mathbf{v}^+/\tau)/\tau \mathbf{v}^+ - \sigma(\mathbf{u}^T \mathbf{v}^-/\tau)/\tau \mathbf{v}^- )</td>
</tr>
<tr>
<td>Margin Triplet</td>
<td>( -\max(\mathbf{u}^T \mathbf{v}^- - \mathbf{u}^T \mathbf{v}^+ + m, 0) )</td>
<td>( \mathbf{v}^+ - \mathbf{v}^- \text{ if } \mathbf{u}^T \mathbf{v}^+ - \mathbf{u}^T \mathbf{v}^- &lt; m \text{ else } 0 )</td>
</tr>
</tbody>
</table>

**Figure 8.** Linear evaluation of representations with different projection heads \( g(\cdot) \) and various dimensions of \( z = g(\mathbf{h}) \). The representation \( \mathbf{h} \) (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. \( g(\mathbf{h}) \). 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, \( \mathbf{h} \), is still much better (+10%) than the layer after, \( z = g(\mathbf{h}) \), 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 the loss of information induced by the contrastive loss. In particular, \( z = g(\mathbf{h}) \) 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 \( \mathbf{h} \). To verify this hypothesis, we conduct experiments that use either \( \mathbf{h} \) or \( g(\mathbf{h}) \) to learn to predict the transformation applied during the pretraining. Here we set \( g(\mathbf{h}) = W(2) \sigma(W(1) \mathbf{h}) \), with the same input and output dimensionality (i.e. 2048). Table 3 shows \( \mathbf{h} \) contains much more information about the transformation applied, while \( g(\mathbf{h}) \) loses information. Further analysis can be found in Appendix B.4.

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) \( \ell_2 \) normalization (i.e. cosine similarity) along with temperature effectively weighs 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 \( \ell_2 \) normalization for all loss functions, and we tune the hyperparameters, and report their best results. Table 4 shows that, while (semi-hard) negative mining helps, the best result is still much worse than our default NT-Xent loss.

---

8Details can be found in Appendix B.10. For simplicity, we only consider the negatives from one augmentation view.
We next test the importance of the $\ell_2$ normalization (i.e. cosine similarity vs dot product) and temperature $\tau$ in our default NT-Xent loss. Table 5 shows that without normalization and proper temperature scaling, performance is significantly worse. Without $\ell_2$ 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.

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 $1 \times$, $2 \times$, and $4 \times$). 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.,...
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×) 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.

### Linear evaluation:

<table>
<thead>
<tr>
<th>Model</th>
<th>Food</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
<th>Birdsnap</th>
<th>SUN397</th>
<th>Cars</th>
<th>Aircraft</th>
<th>VOC2007</th>
<th>DTD</th>
<th>Pets</th>
<th>Caltech-101</th>
<th>Flowers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimCLR (ours)</td>
<td><strong>76.9</strong></td>
<td><strong>95.3</strong></td>
<td>80.2</td>
<td>48.4</td>
<td><strong>65.9</strong></td>
<td>60.0</td>
<td>61.2</td>
<td><strong>84.2</strong></td>
<td><strong>78.9</strong></td>
<td><strong>92.3</strong></td>
<td><strong>94.1</strong></td>
<td><strong>94.2</strong></td>
</tr>
<tr>
<td>Supervised</td>
<td>75.2</td>
<td>95.7</td>
<td><strong>81.2</strong></td>
<td><strong>56.4</strong></td>
<td>64.9</td>
<td><strong>68.8</strong></td>
<td><strong>63.8</strong></td>
<td>83.8</td>
<td><strong>78.7</strong></td>
<td><strong>94.1</strong></td>
<td><strong>94.2</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Fine-tuned:

<table>
<thead>
<tr>
<th>Model</th>
<th>Food</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
<th>Birdsnap</th>
<th>SUN397</th>
<th>Cars</th>
<th>Aircraft</th>
<th>VOC2007</th>
<th>DTD</th>
<th>Pets</th>
<th>Caltech-101</th>
<th>Flowers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimCLR (ours)</td>
<td><strong>89.4</strong></td>
<td><strong>98.6</strong></td>
<td><strong>89.0</strong></td>
<td><strong>78.2</strong></td>
<td><strong>68.1</strong></td>
<td><strong>92.1</strong></td>
<td><strong>87.0</strong></td>
<td><strong>86.6</strong></td>
<td><strong>77.8</strong></td>
<td><strong>92.1</strong></td>
<td><strong>94.1</strong></td>
<td><strong>97.6</strong></td>
</tr>
<tr>
<td>Supervised</td>
<td>88.7</td>
<td>98.3</td>
<td><strong>88.7</strong></td>
<td><strong>77.8</strong></td>
<td>67.0</td>
<td>91.4</td>
<td><strong>88.0</strong></td>
<td>86.5</td>
<td><strong>87.8</strong></td>
<td><strong>93.2</strong></td>
<td><strong>94.2</strong></td>
<td><strong>98.0</strong></td>
</tr>
<tr>
<td>Random init</td>
<td>88.3</td>
<td>96.0</td>
<td>81.9</td>
<td><strong>77.0</strong></td>
<td>53.7</td>
<td>91.3</td>
<td>84.8</td>
<td>69.4</td>
<td>64.1</td>
<td>82.7</td>
<td>72.5</td>
<td>92.5</td>
</tr>
</tbody>
</table>

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×) 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×, 4×) 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×) 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.

11The 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


A Simple Framework for Contrastive Learning of Visual Representations


