Understanding Negative Samples in Instance Discriminative Self-supervised Representation Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Instance discriminative self-supervised representation learning has been attracted attention thanks to its unsupervised nature and informative feature representation for downstream tasks. In practice, it commonly uses a larger number of negative samples than the number of supervised classes. However, there is an inconsistency in the existing analysis; theoretically, a large number of negative samples degrade classification performance on a downstream supervised task, while empirically, they improve the performance. We provide a novel framework to analyze this empirical result regarding negative samples using the coupon collector's problem. Our bound can implicitly incorporate the supervised loss of the downstream task in the self-supervised loss by increasing the number of negative samples. We confirm that our proposed analysis holds on real-world benchmark datasets.

1 Introduction

Self-supervised representation learning is a popular class of unsupervised representation learning algorithms in the domains of vision [Bachman et al., 2019, Chen et al., 2020a, He et al., 2020, Caron et al., 2020, Grill et al., 2020, Chen and He, 2021] and language [Mikolov et al., 2013, Devlin et al., 2019, Brown et al., 2020]. Generally, it trains a feature extractor by solving a pretext task constructed on a large unlabeled dataset. The learned feature extractor yields generic feature representations for other machine learning tasks such as classification. Recent self-supervised representation learning algorithms help a linear classifier to attain classification accuracy comparable to a supervised method from scratch, especially in a few amount of labeled data regime [Newell and Deng, 2020, Hénaff et al., 2020, Chen et al., 2020b]. For example, `SwAV` [Caron et al., 2020] with ResNet-50 has a top-1 validation accuracy of 75.3% on the ImageNet-1K classification [Deng et al., 2009] compared with 76.5% by using the fully supervised method.

InfoNCE [van den Oord et al., 2018, Eq. 4] or its modification is a de facto standard loss function used in many state-of-the-art self-supervised methods [Logeswaran and Lee, 2018, Bachman et al., 2019, He et al., 2020, Chen et al., 2020a, Hénaff et al., 2020, Caron et al., 2020]. Intuitively, the minimization of InfoNCE can be viewed as the minimization of cross-entropy loss on $K+1$ instance-wise classification, where K is the number of negative samples. Despite the empirical success of self-supervised learning, we still do not understand why the self-supervised learning algorithms with InfoNCE perform well for downstream tasks.

Arora et al. [2019] propose the first theoretical framework for contrastive unsupervised representation learning (CURL). However, there exists a gap between the theoretical analysis and empirical observation as in Figure 1. Precisely, we expect that a large number of negative samples degrade a supervised loss on a downstream task from the analysis by Arora et al. [2019]. In practice, however,

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

a large number of negative samples are commonly used in self-supervised representation learning algorithms [He et al., 2020, Chen et al., 2020a].

Contributions. We show difficulty to explain why large negative samples empirically improve supervised accuracy on the downstream task from the CURL framework when we use learned representations as feature vectors for the supervised classification in Section 3. To fill the gap, we propose a novel lower bound to theoretically explain this empirical observation regarding negative samples using the coupon collector's problem in Section 4.

2 InfoNCE-based Self-supervised Representations Learning

We focus on Chen et al. [2020a]'s self-supervised representation learning formulation, namely, `SimCLR`. Let $\mathcal{X}$ be an input space, e.g., $\mathcal{X} \subset \mathbb{R}^{\text{channel}\times\text{width}\times\text{height}}$ for color images. We can only access a unlabeled training dataset $\{\mathbf{x}_i\}_{i=1}^N$, where input $\mathbf{x} \in \mathcal{X}$. `SimCLR` learns feature extractor $\mathbf{f} : \mathcal{X} \to \mathbb{R}^h$ modeled by neural networks on the dataset, where h is the dimensionality of feature representation. Let $\mathbf{z} = \mathbf{f}(\mathbf{a}(\mathbf{x}))$ be a feature representation of $\mathbf{x}$ after applying data augmentation $\mathbf{a} : \mathcal{X} \to \mathcal{X}$. Data augmentation is a pre-defined stochastic function such as a composition of the horizontal flipping and cropping. Note that the output of $\mathbf{f}$ is normalized by its L2 norm. Let $\mathbf{z}^+ = \mathbf{f}(\mathbf{a}^+(\mathbf{x}))$ be a positive feature representation created from $\mathbf{x}$ with different data augmentation $\mathbf{a}^+(\cdot)$.

`SimCLR` minimizes the following InfoNCE-based loss with K negative samples for each pair of $(\mathbf{z}, \mathbf{z}^+)$:

$$\ell_{\text{Info}}(\mathbf{z}, \mathbf{Z}) \coloneqq -\ln \frac{\exp(\mathbf{z}\cdot\mathbf{z}^+/t)}{\sum_{\mathbf{z}_k\in\mathbf{Z}} \exp(\mathbf{z}\cdot\mathbf{z}_k/t)}, \tag{1}$$

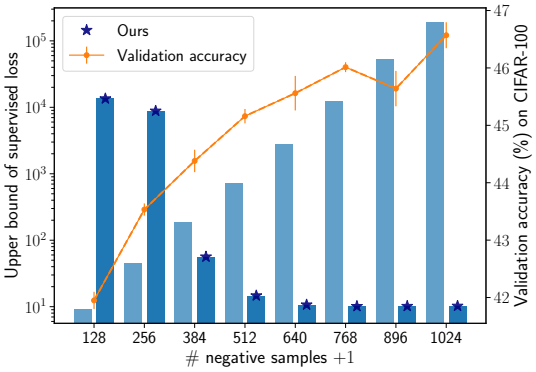

Figure 1: Upper bounds of supervised loss and validation accuracy on CIFAR-100. By increasing the number of negative samples, validation accuracy tends to improve. **Left bars**: The existing contrastive unsupervised representation learning bound (8) also increases when the number of negative samples increases because the bound of supervised loss explodes due to a collision term that is not related to classification loss (see Eq. (8) for the definition). **Right bars with ★**: On the other hand, the proposed counterpart (10) does not explode. Section 5 describes the details of this experiment.

where $\mathbf{Z} = \{\mathbf{z}^+, \mathbf{z}_1^-, \ldots, \mathbf{z}_K^-\}$ that is a set of positive representation $\mathbf{z}^+$ and K negative representations $\{\mathbf{z}_1^- \ldots, \mathbf{z}_K^-\}$ created from other samples, $(\cdot\cdot)$ is an inner product of two representations, and $t \in \mathbb{R}_+$ is a temperature parameter.[1] Intuitively, this loss function approximates an instance-wise classification loss by using K random negative samples [Wu et al., 2018]. From the definition of Eq. (1), we expect that $\mathbf{f}$ learns an invariant encoder with respect to data augmentation $\mathbf{a}$. After minimizing Eq. (1), $\mathbf{f}$ works as a feature extractor for downstream tasks such as classification.

3 Extension of Contrastive Unsupervised Representation Learning Framework

We focus on the direct relationship between the self-supervised loss and a supervised loss to understand the role of the number of negative samples K. For a similar problem, the CURL framework [Arora et al., 2019] shows that the averaged supervised loss is bounded by the contrastive unsupervised loss. Thus, we extend the analysis of CURL to the self-supervised representation learning in Section 2 and point out that we have difficulty explaining the empirical observation as in Figure 1 from the existing analysis. Table 2 in Appendix A summarizes notations used in this paper for convenience.

[1] We perform the analysis with $t = 1$ for simplicity, but our analysis holds with any temperature.

3.1 CURL Formulation for `SimCLR`

By following the CURL analysis [Arora et al., 2019], we introduce the learning process in two steps: *self-supervised representation learning step* and *supervised learning step* (see Section 3.1.1 and Section 3.1.2, respectively).

3.1.1 Self-supervised representation learning step

The purpose of the self-supervised learning step is to learn a feature extractor $\mathbf{f}$ on an unlabeled dataset. During this step, we can only access input samples from $\mathcal{X}$ without any relationship between samples, unlike metric learning [Kulis, 2012] or similar unlabeled learning [Bao et al., 2018].

We formulate the data generation process for Eq. (1). The key idea of the CURL analysis is the existence of latent classes $\mathcal{C}$ that are associated with supervised classes.[2] Let ρ be a probability distribution over $\mathcal{C}$, and let $\mathbf{x}$ be an input sample drawn from a data distribution $\mathcal{D}_c$ conditioned on a latent class $c \in \mathcal{C}$. We draw two data augmentations $(\mathbf{a}, \mathbf{a}^+)$ from the distribution of data augmentation $\mathcal{A}$ and apply them independently to the sample. As a result, we have two augmented samples $(\mathbf{a}(\mathbf{x}), \mathbf{a}^+(\mathbf{x}))$ and call $\mathbf{a}^+(\mathbf{x})$ as a positive sample of $\mathbf{a}(\mathbf{x})$. Similarly, we draw K negative samples from $\mathcal{D}_{c_k^-}$ for each $c_k^- \in \{c_1^-, \dots, c_K^-\} \sim \rho^K$ and K data augmentations from $\mathcal{A}$, and then we apply them to negative samples. Definition 1 summarizes the data generation process.

Definition 1 (Data Generation Process in Self-supervised Representation Learning Step)**.**
1. Draw latent classes: $c, \{c_k^-\}_{k=1}^K \sim \rho^{K+1}$*;* *2. Draw input sample:* $\mathbf{x} \sim \mathcal{D}_c$*;*
3. Draw data augmentations: $(\mathbf{a}, \mathbf{a}^+) \sim \mathcal{A}^2$*;* *4. Apply data augmentations:* $\mathbf{a}(\mathbf{x}), \mathbf{a}^+(\mathbf{x})$*;*
5. Draw negative samples: $\{\mathbf{x}_k^-\}_{k=1}^K \sim \mathcal{D}_{c_k^-}^K$*;* *6. Draw data augmentations:* $\{\mathbf{a}_k^-\}_{k=1}^K \sim \mathcal{A}^K$*;*
7. Apply data augmentations: $\{\mathbf{a}_k^-(\mathbf{x}_k^-)\}_{k=1}^K$*.*

Note that the original data generation process of CURL samples a positive sample of $\mathbf{x}$ from $\mathcal{D}_c$ independently and does not use any data augmentations.

From the data generation process and InfoNCE loss (1), we give the following formal definition of self-supervised loss.

Definition 2 (Expected Self-supervised Loss)**.**

$$L_{\mathrm{Info}}(\mathbf{f}) \coloneqq \mathop{\mathbb{E}}_{c, \{c_k^-\}_{k=1}^K \sim \rho^{K+1}} \mathop{\mathbb{E}}_{\substack{\mathbf{x} \sim \mathcal{D}_c \\ (\mathbf{a}, \mathbf{a}^+) \sim \mathcal{A}^2}} \mathop{\mathbb{E}}_{\substack{\{\mathbf{x}_k^- \sim \mathcal{D}_{c_k^-}\}_{k=1}^K \\ \{\mathbf{a}_k^-\}_{k=1}^K \sim \mathcal{A}^K}} \ell_{\mathrm{Info}}(\mathbf{z}, \mathbf{Z}), \tag{2}$$

where recall that $\mathbf{z} = \mathbf{f}(\mathbf{a}(\mathbf{x}))$ *and* $\mathbf{Z} = \{\mathbf{f}(\mathbf{a}^+(\mathbf{x})), \mathbf{f}(\mathbf{a}_1^-(\mathbf{x}_1^-)), \dots, \mathbf{f}(\mathbf{a}_K^-(\mathbf{x}_K^-))\}$.

Since we cannot directly minimize Eq. (2), we minimize the empirical counterpart, $\widehat{L}_{\mathrm{Info}}(\mathbf{f})$, by sampling from the training dataset. After minimizing $\widehat{L}_{\mathrm{Info}}(\mathbf{f})$, learned $\widehat{\mathbf{f}}$ works as a feature extractor in the supervised learning step.

3.1.2 Supervised learning step

At the supervised learning step, we can observe label $y \in \mathcal{Y} = \{1, \dots, Y\}$ for each $\mathbf{x}$ that is used in the self-supervised learning step. Let $\mathcal{S}$ be a supervised data distribution over $\mathcal{X} \times \mathcal{Y}$, and $\mathbf{g} \circ \widehat{\mathbf{f}} : \mathcal{X} \to \mathbb{R}^Y$ be a classifier, where $\mathbf{g} : \mathbb{R}^h \to \mathbb{R}^Y$ and $\widehat{\mathbf{f}}$ is the frozen feature extractor. Given the supervised data distribution and classifier, our goal is to minimize the following supervised loss:

$$L_{\mathrm{sup}}(\mathbf{g} \circ \widehat{\mathbf{f}}) \coloneqq \mathop{\mathbb{E}}_{\substack{\mathbf{x}, y \sim \mathcal{S} \\ \mathbf{a} \sim \mathcal{A}}} -\ln \frac{\exp\left(\mathbf{g}_y(\widehat{\mathbf{f}}(\mathbf{a}(\mathbf{x})))\right)}{\sum_{j \in \mathcal{Y}} \exp\left(\mathbf{g}_j(\widehat{\mathbf{f}}(\mathbf{a}(\mathbf{x})))\right)}. \tag{3}$$

By following the CURL analysis, we introduce a mean classifier as a simple instance of $\mathbf{g}$ because its loss is an upper bound of Eq. (3). The mean classifier is the linear classifier whose weight of

[2] Technically, we do not need c to draw $\mathbf{x}$ for self-supervised representation learning. However, we explicitly include it in the data generation process to understand the relationship with a supervised task in Section 3.2.

label y is computed by averaging representations: $\boldsymbol{\mu}_y = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_y} \mathbb{E}_{\mathbf{a}\sim\mathcal{A}} \mathbf{f}(\mathbf{a}(\mathbf{x}))$, where $\mathcal{D}_y$ is a data distribution conditioned on the supervise label y. We introduce the definition of the mean classifier's supervised loss as follows:

Definition 3 (Mean Classifier's Supervised Loss)**.**

$$L^{\mu}_{\mathrm{sup}}(\widehat{\mathbf{f}}) \coloneqq \mathop{\mathbb{E}}_{\substack{\mathbf{x},y\sim\mathcal{S}\\ \mathbf{a}\sim\mathcal{A}}} -\ln \frac{\exp\left(\widehat{\mathbf{f}}(\mathbf{a}(\mathbf{x}))\cdot\boldsymbol{\mu}_y\right)}{\sum_{j\in\mathcal{Y}}\exp\left(\widehat{\mathbf{f}}(\mathbf{a}(\mathbf{x}))\cdot\boldsymbol{\mu}_j\right)}. \tag{4}$$

We also introduce a sub-class loss function.[3] Let $\mathcal{Y}_{\mathrm{sub}}$ be a subset of $\mathcal{Y}$ and $\mathcal{S}_{\mathrm{sub}}$ be a data distribution over $\mathcal{X}\times\mathcal{Y}_{\mathrm{sub}}$, then we define the sub-class losses of classifier $\mathbf{g}$ and mean classifier:

Definition 4 (Supervised Sub-class Losses of Classifier $\mathbf{g}$ and Mean Classifier with $\widehat{\mathbf{f}}$)**.**

$$L_{\mathrm{sub}}(\mathbf{g}\circ\widehat{\mathbf{f}},\mathcal{Y}_{\mathrm{sub}}) \coloneqq \mathop{\mathbb{E}}_{\substack{\mathbf{x},y\sim\mathcal{S}_{\mathrm{sub}}\\ \mathbf{a}\sim\mathcal{A}}} -\ln \frac{\exp\left(\mathbf{g}_y(\widehat{\mathbf{f}}(\mathbf{a}(\mathbf{x})))\right)}{\sum_{j\in\mathcal{Y}_{\mathrm{sub}}}\left(\mathbf{g}_j(\widehat{\mathbf{f}}(\mathbf{a}(\mathbf{x})))\right)}, \tag{5}$$

$$L^{\mu}_{\mathrm{sub}}(\widehat{\mathbf{f}},\mathcal{Y}_{\mathrm{sub}}) \coloneqq \mathop{\mathbb{E}}_{\substack{\mathbf{x},y\sim\mathcal{S}_{\mathrm{sub}}\\ \mathbf{a}\sim\mathcal{A}}} -\ln \frac{\exp\left(\widehat{\mathbf{f}}(\mathbf{a}(\mathbf{x}))\cdot\boldsymbol{\mu}_y\right)}{\sum_{j\in\mathcal{Y}_{\mathrm{sub}}}\exp\left(\widehat{\mathbf{f}}(\mathbf{a}(\mathbf{x}))\cdot\boldsymbol{\mu}_j\right)}. \tag{6}$$

The purpose of unsupervised representation learning [Bengio et al., 2013] is to learn generic feature representation rather than improve the accuracy of the classifier on the same dataset. However, we believe that such a feature extractor tends to transfer well to another task. Indeed, Kornblith et al. [2019] empirically show a strong correlation between ImageNet's accuracy and transfer accuracy.

3.2 Theoretical Analysis based on CURL

We show that InfoNCE loss (2) is an upper bound of the expected sub-class loss of the mean classifier (6).

Step 1. Introduce a lower bound We denote $\boldsymbol{\mu}(\mathbf{x}) = \mathbb{E}_{\mathbf{a}\sim\mathcal{A}}\mathbf{f}(\mathbf{a}(\mathbf{x}))$ and derive a lower bound of unsupervised loss $L_{\mathrm{Info}}(\mathbf{f})$.

$$\begin{aligned}
L_{\mathrm{Info}}(\mathbf{f}) &\geq \mathop{\mathbb{E}}_{c,\{c_k^-\}_{k=1}^K\sim\rho^{K+1}} \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim\mathcal{D}_c\\ \mathbf{a}\sim\mathcal{A}}} \mathop{\mathbb{E}}_{\{\mathbf{x}_k^-\sim\mathcal{D}_{c_k^-}\}_{k=1}^K} \ell_{\mathrm{Info}}\left(\mathbf{f}(\mathbf{a}(\mathbf{x})),\left\{\boldsymbol{\mu}(\mathbf{x}),\boldsymbol{\mu}(\mathbf{x}_1^-),\ldots,\boldsymbol{\mu}(\mathbf{x}_K^-)\right\}\right)\\
&\geq \mathop{\mathbb{E}}_{c,\{c_k^-\}_{k=1}^K\sim\rho^{K+1}} \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim\mathcal{D}_c\\ \mathbf{a}\sim\mathcal{A}}} \ell_{\mathrm{Info}}\left(\mathbf{f}(\mathbf{a}(\mathbf{x})),\left\{\boldsymbol{\mu}(\mathbf{x}),\boldsymbol{\mu}_{c_1^-},\ldots,\boldsymbol{\mu}_{c_K^-}\right\}\right)\\
&\geq \mathop{\mathbb{E}}_{c,\{c_k^-\}_{k=1}^K\sim\rho^{K+1}} \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim\mathcal{D}_c\\ \mathbf{a}\sim\mathcal{A}}} \ell_{\mathrm{Info}}\left(\mathbf{f}(\mathbf{a}(\mathbf{x})),\left\{\boldsymbol{\mu}_c,\boldsymbol{\mu}_{c_1^-},\ldots,\boldsymbol{\mu}_{c_K^-}\right\}\right) + d(\mathbf{f}),
\end{aligned} \tag{7}$$

$$\text{where } d(\mathbf{f}) = \frac{1}{t}\mathop{\mathbb{E}}_{c\sim\rho}\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}_c}\left\{\mathop{\mathbb{E}}_{\mathbf{a}\sim\mathcal{A}}\left[\mathbf{f}(\mathbf{a}(\mathbf{x}))\right]\cdot\left[\mathop{\mathbb{E}}_{\substack{\mathbf{x}^+\sim\mathcal{D}_c\\ \mathbf{a}_1^+\sim\mathcal{A}}}\left[\mathbf{f}(\mathbf{a}_1^+(\mathbf{x}^+))\right] - \mathop{\mathbb{E}}_{\mathbf{a}_2^+\sim\mathcal{A}}\left[\mathbf{f}(\mathbf{a}_2^+(\mathbf{x}))\right]\right]\right\}.$$

The first and second inequalities are done by using Jensen's inequality for convex function. The proof of the third inequality is shown in Appendix B.1.

Remark 5 (Effect of gap term $d(\mathbf{f})$)**.** *We confirm that $d(\mathbf{f})$ is an almost constant among different K in practice (see Table 1). Therefore we focus on the first term in Eq.* (7) *in the following analysis.*

Step 2. Decomposition into the expected sub-class loss We convert the first term of Eq. (7) into the expected sub-class loss explicitly. By following Arora et al. [2019, Theorem B.1], we introduce collision probability: $\tau_K = \mathbb{P}(\mathrm{Col}(c,\{c_k^-\}_{k=1}^K)\neq 0)$, where $\mathrm{Col}(c,\{c_k^-\}_{k=1}^K) = \sum_{k=1}^K \mathbb{I}[c=c_k^-]$ and

[3] Arora et al. [2019] refer to this loss function as *averaged supervised loss*.

$\mathbb{I}[\cdot]$ is the indicator function. We omit the arguments of Col for simplicity. Let $\mathcal{C}_{\mathrm{sub}}(\{c, c_1^-, \dots, c_K^-\})$ be a function to remove duplicated latent classes given latent classes. We omit the arguments of $\mathcal{C}_{\mathrm{sub}}$ as well. The result is our extension of Arora et al. [2019, Lemma 4.3] for self-supervised representation learning as the following proposition:

Proposition 6 (CURL Lower Bound of Self-supervised Loss)**.** *For all feature extractor* $\mathbf{f}$*,*

$$L_{\mathrm{Info}}(\mathbf{f}) \geq (1-\tau_K) \mathop{\mathbb{E}}_{c,\{c_k^-\}_{k=1}^K \sim \rho^{K+1}} [\underbrace{L_{\mathrm{sub}}^{\mu}(\mathbf{f}, \mathcal{C}_{\mathrm{sub}})}_{\textit{sub-class loss}} \mid \mathrm{Col} = 0] + \tau_K \mathop{\mathbb{E}}_{c,\{c_k^-\}_{k=1}^K \sim \rho^{K+1}} [\underbrace{\ln(\mathrm{Col}+1)}_{\textit{collision}} \mid \mathrm{Col} \neq 0] + d(\mathbf{f}). \tag{8}$$

The proof is found in Appendix B.2.

Arora et al. [2019] also show Rademacher complexity-based generalization error bound for the mean classifier. Since we focus on the behavior of the self-supervised loss rather than a generalization error bound when K increases, we do not perform further analysis as was done in Arora et al. [2019]. Nevertheless, we believe that constructing a generalization error bound by following either Arora et al. [2019, Theorem 4.1] or Nozawa et al. [2020, Theorem 7] is worth interesting future work.

3.3 Limitations of Eq. (8)

The bound (8) tells us that K gives a trade-off between the sub-class loss and collision term via τ_K. Recall that our final goal is to minimize the supervised loss (4) rather than expected sub-class loss (6). To incorporate the supervised loss (4) into the lower bound (8), we need K large enough to satisfy $\mathcal{C}_{\mathrm{sub}} \supseteq \mathcal{Y}$. However, such a large K makes the lower bound meaningless.

We can easily observe that the lower bound (8) converges to the collision term by increasing K since the collision probability τ_K converges to 1. As a result, the sub-class loss rarely contributes to the lower bound. Let us consider the bound on CIFAR-10, where the number of supervised classes is 10, and latent classes are the same as the supervised classes. When $\tau_{K=32} \approx 0.967$, i.e., the only 3.3% training samples contribute to the expected sub-class loss, the others fall into the collision term. Indeed, Arora et al. [2019] show that small latent classes or large negative samples degrade classification performance of the expected sub-class classification task. However, even much larger negative samples, $K+1 = 512$, yield the best performance of a linear classifier with self-supervised representation on CIFAR-10 [Chen et al., 2020a, B.9].

4 Proposed Lower Bound for Instance-wise Self-supervised Representation Learning

To fill the gap between the theoretical bound (8) and empirical observation from recent work, we propose another lower bound for self-supervised representation learning. The key idea of our bound is to replace τ with a different probability since τ can quickly increase depending on K. The idea is motivated by focusing on the supervised loss rather than the expected sub-class loss.

4.1 Proposed Lower Bound

Let υ_K be a probability that sampled K latent classes contain all latent classes: $\{c_k\}_{k=1}^K \supseteq \mathcal{C}$. This probability appears in the coupon collector's problem of probability theory (e.g., Durrett [2019, Example 2.2.7]). If the latent class probability is uniform: $\forall c, \rho(c) = 1/|\mathcal{C}|$, then we can calculate the probability explicitly as follows.

Definition 7 (Probability to Draw All Latent Classes)**.** *Assume that* ρ *is a uniform distribution over latent classes* $\mathcal{C}$*. The probability that* K *latent classes drawn from* ρ *contain all latent classes is defined as*

$$\upsilon_K \coloneqq \sum_{n=1}^{K} \sum_{m=0}^{|\mathcal{C}|-1} \binom{|\mathcal{C}|-1}{m} (-1)^m \left(1 - \frac{m+1}{|\mathcal{C}|}\right)^{n-1}, \tag{9}$$

where the first summation is a probability that n drawn latent samples contain all latent classes [Nakata and Kubo, 2006, Eq. 2].[4]

By replacing τ with υ, we obtain our lower bound of InfoNCE loss:

Theorem 8 (Proposed Lower Bound of Self-supervised Loss)**.** *For all feature extractor* $\mathbf{f}$*,*

$$\begin{aligned} L_{\mathrm{Info}}(\mathbf{f}) \geq \frac{1}{2}\Big\{ \upsilon_{K+1} & \mathop{\mathbb{E}}_{c,\{c_k^-\}_{k=1}^K \sim \rho^{K+1}} [\underbrace{L_{\mathrm{sub}}^{\mu}(\mathbf{f},\mathcal{C})}_{\textit{sup. loss}} \mid \mathcal{C}_{\mathrm{sub}} = \mathcal{C}] \\ +(1-\upsilon_{K+1}) & \mathop{\mathbb{E}}_{c,\{c_k^-\}_{k=1}^K \sim \rho^{K+1}} [\underbrace{L_{\mathrm{sub}}^{\mu}(\mathbf{f},\mathcal{C}_{\mathrm{sub}})}_{\textit{sub-class loss}} \mid \mathcal{C}_{\mathrm{sub}} \neq \mathcal{C}] \\ & + \mathop{\mathbb{E}}_{c,\{c_k^-\}_{k=1}^K \sim \rho^{K+1}} \underbrace{\ln(\mathrm{Col}+1)}_{\textit{collision}} \Big\} + d(\mathbf{f}). \end{aligned} \tag{10}$$

The proof is found in Appendix B.3.

Theorem 8 tells us that probability υ_{K+1} converges to 1 by increasing the number of negative samples K; as a result, the self-supervised loss is more likely to contain the supervised loss and the collision term. The sub-class loss contributes to the self-supervised loss when K is small as in Eq. (8). Let us consider the example value of υ with the same setting discussed in Section 3.3. When $\upsilon_{K=32} \approx 0.719$, i.e., 71.9% training samples contribute the supervised loss.

4.2 Increasing K does not Spread Normalized Features within Same Latent Class

We argue that the feature representations do not have a large within-class variance on the feature space by increasing K in practice. To do so, we show the upper bound of the collision term in the lower bounds to understand the effect of large negative samples.

Corollary 9 (Upper Bound of Collision Term)**.** *Given a latent class c, K negative classes $\{c_k^-\}_{c=1}^K$, and feature extractor* $\mathbf{f}$*,*

$$\ln\left(\mathrm{Col}\left(c,\{c_k^-\}_{k=1}^K\right)+1\right) \leq \alpha + \beta \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim\mathcal{D}_c \\ (\mathbf{a},\mathbf{a}^+)\sim\mathcal{A}^2}} \mathop{\mathbb{E}}_{\substack{\mathbf{x}'\sim\mathcal{D}_c \\ \mathbf{a}'\sim\mathcal{A}}} \left|\mathbf{f}(\mathbf{a}(\mathbf{x})) \cdot \left[\mathbf{f}(\mathbf{a}'(\mathbf{x}')) - \mathbf{f}(\mathbf{a}^+(\mathbf{x}))\right]\right|, \tag{11}$$

where α and β are non-negative constants depending on the number of duplicated latent classes.

The proof is found in Appendix B.4. A similar bound is shown by Arora et al. [2019, Lemma 4.4].

Intuitively, we expect that two feature representations in the same latent class tend to be dissimilar by increasing K. However, Eq. (11) converges even if K is small in practice. Let us consider the condition when this upper bound achieves the maximum. Since $\mathbf{f}(\mathbf{a}(\mathbf{x}))$ and $\mathbf{f}(\mathbf{a}^+(\mathbf{x}))$ are computed from the same input sample with different data augmentations, their inner product tends to be 1; thus, $\mathbf{f}(\mathbf{a}'(\mathbf{x}'))$ is located in the opposite direction of $\mathbf{f}(\mathbf{a}(\mathbf{x}))$. But, it is not possible to learn such representations because of the definition of InfoNCE with normalized representation and the dimensionality of feature space: Equilateral dimension. We confirm that Eq. (11) without α and β does not increase by increasing K (see Table 1 and Appendix D for more analysis).

4.3 Small K can Give Consistent Loss Function for Supervised Task

As we shown in Theorem 8, L_{Info} with large K can be viewed as an upper bound of L_{sup}^{μ}. However, L_{Info} with smaller K can still yield good feature representations for downstream tasks on ImageNet-1K as reported by Chen et al. [2020a]. Arora et al. [2019] also reported similar results on CIFAR-100 with contrastive losses.[5] To shed light on this smaller K regime, we focus on the class distributions of both datasets, ImageNet-1K and CIFAR-100, which are almost uniform.

Proposition 10 (Optimality of L_{sub})**.** *Suppose $\mathcal{C} \supseteq \mathcal{Y}$ and ρ is uniform: $\forall c \in \mathcal{C}, \rho(c) = 1/|\mathcal{C}|$. Suppose $\mathbf{q} : \mathcal{X} \to \left[q_1, \ldots, q_{|\mathcal{C}|}\right]^\top \in \mathbb{R}^{|\mathcal{C}|}$. Optimal $\mathbf{q}^*$ is a constant function that outputs a vector*

[4] We report the expected $K+1$ to draw all supervised labels for CIFAR-10, CIFAR-100, and ImageNet-1K in Appendix C.

[5] Arora et al. [2019, Table D.1] mentioned that this phenomenon is not covered by the CURL framework.

with the same value if and only if it minimizes $\mathbb{E}_{c,\{c_k^-\}_{k=1}^K \sim \rho^{K+1}} L_{\mathrm{sub}}(\mathbf{q}^*, \mathcal{C}_{\mathrm{sub}})$*. The optimal* $\mathbf{q}^*$ *is also the minimizer of* L_{sup}*.*

The proof is found in Appendix B.5 inspired by Titsias [2016, Proposition 2].

Proposition 10 *does not* argue that self-supervised representation learning algorithms fail to learn feature representations on a dataset with a non-uniform class distribution. This is because we perform a supervised algorithm on a downstream dataset after representation learning in general.

4.4 Relation to Clustering-based Self-supervised Representation Learning

During the minimization of L_{Info}, we cannot minimize L_{sup}^{μ} in Eq. (10) directly since we cannot access supervised and latent classes. Interestingly, we find a similar formulation in clustering-based self-supervised representation learning algorithms.

Remark 11. *Clustering-based self-supervised representation learning algorithms, such as* `DeepCluster` *[Caron et al., 2018],* `SwAV` *[Caron et al., 2020], and* `PCL` *[Li et al., 2021], use prototype representations instead of* $\mathbf{z}^+, \{\mathbf{z}_k^-\}_{k=1}^K$ *by applying unsupervised clustering on feature representations. This procedure is justified as the approximation of latent class's mean representation* $\boldsymbol{\mu}_c$ *with a prototype representation to minimize the supervised loss in Eq.* (10) *rather than Eq.* (2)*, as a result, the mini-batch size does not depend on the number of negative samples.*

This replacement supports the empirical observation in Caron et al. [2020, Section 4.3], where the authors reported `SwAV` maintained top-1 accuracy on ImageNet-1K with a small mini-batch size, 200, compared to a large mini-batch, 4 096. On the other hand, `SimCLR` [Chen et al., 2020a] did not.

5 Experiments

We numerically confirm our theoretical findings by using `SimCLR` [Chen et al., 2020a] on the image classification tasks. Appendix D contains some analysis that has been omitted in this section due to the lack of space. We used datasets with a relatively small number of classes to compare bounds. Our experimental codes are available in Supplementary Material.[6]

Datasets and Data Augmentations We used the CIFAR-10 and CIFAR-100 [Krizhevsky, 2009] image classification datasets with the original 50 000 training samples for both self-supervised and supervised training and the original 10 000 validation samples for the evaluation of supervised learning. We used the same data augmentations in the CIFAR-10 experiment by Chen et al. [2020a]: random resize cropping with the original image size, horizontal flipping with probability 0.5, color jitter with a strength parameter of 0.5 with probability 0.8, and grey scaling with probability 0.2.

Self-supervised Learning We mainly followed the experimental setting provided by Chen et al. [2020a] and its implementation.[7] We used ResNet-18 [He et al., 2016] as a feature encoder without the last fully connected layer. We replaced the first convolution layer with the convolutional layer with 64 output channels, the stride size of 1, the kernel size of 3, and the padding size of 3. We removed the first max-pooling from the encoder, and we added a non-linear projection head to the end of the encoder. The projection head consisted of the fully connected layer with 512 units, batch-normalization [Ioffe and Szegedy, 2015], ReLU activation function, and another fully connected layer with 128 units and without bias.

We trained the encoder by using PyTorch [Paszke et al., 2019]'s distributed data-parallel training on four GPUs, which are NVIDIA Tesla P100 on an internal cluster. For distributed training, we replaced all batch-normalization with synchronized batch-normalization. We used stochastic gradient descent with momentum factor of 0.9 on 500 epochs. We used LARC [You et al., 2017][8], and its global learning rate was updated by using linear warmup at each step during the first 10 epochs, then updated by using cosine annealing without restart [Loshchilov and Hutter, 2017] at each step

[6] We used Hydra [Yadan, 2019], GNU Parallel [Tange, 2020], Scikit-learn [Pedregosa et al., 2011], Pandas [Reback et al., 2020], Matplotlib [Hunter, 2007], and seaborn [Waskom, 2021] in our experiments.

[7] `https://github.com/google-research/simclr`

[8] `https://github.com/NVIDIA/apex`

Table 1: The bound values on CIFAR-10/100 experiments with different $K+1$. CURL bound and its quantities are shown with †. The proposed ones are shown without †. Since the proposed collision values are half of †Collision, they are omitted. The reported values contain their coefficient except for Collision bound.

		CIFAR-10				CIFAR-100			
$K+1$		32	128	256	512	128	256	512	1024
μ acc		72.75	77.22	78.60	80.12	32.67	34.25	35.90	37.44
Linear acc		77.13	81.33	82.85	84.13	41.95	43.53	45.16	46.57
L_{Info}	Eq. (2)	2.02	3.29	3.96	4.64	3.32	3.98	4.66	5.34
$d(\mathbf{f})$	Eq. (7)	-1.16	-1.18	-1.18	-1.19	-0.99	-0.98	-0.97	-0.95
$^{\dagger}L_{\mathrm{Info}}$ bound	Eq. (8)	0.23	1.41	2.08	2.75	0.72	0.46	0.78	1.42
†Collision		1.32	2.58	3.26	3.94	0.69	1.15	1.73	2.37
$^{\dagger}L^{\mu}_{\mathrm{sup}}$		0.05	0.00	0.00	0.00	0.00	0.00	0.01	0.00
$^{\dagger}L^{\mu}_{\mathrm{sub}}$		0.01	0.00	0.00	0.00	1.03	0.30	0.01	0.00
L_{Info} bound	Eq. (10)	0.39	1.02	1.35	1.69	1.18	1.53	1.86	2.19
L^{μ}_{sup}		0.63	0.91	0.90	0.90	0.00	0.00	1.17	1.94
L^{μ}_{sub}		0.26	0.00	0.00	0.00	1.82	1.93	0.79	0.01
Collision bound	Eq. (11)	0.60	0.61	0.62	0.62	0.52	0.52	0.51	0.51

until the end. We initialized the learning rate with $(K+1)/256$. We applied weight decay of 10^{-4} to all weights except for parameters of all synchronized batch-normalization and bias terms. The temperature parameter was set to $t = 0.5$.

Linear Evaluation We report the validation accuracy of two linear classifiers: mean classifier and linear classifier. We constructed a mean classifier by averaging the feature representations of $\mathbf{z}$ per supervised class. We applied one data augmentation to each training sample, where the data augmentation was the same as in that the self-supervised learning step. For linear classifier $\mathbf{g}$, we optimized $\mathbf{g}$ by using stochastic gradient descent with Nesterov's momentum [Sutskever et al., 2013] whose factor is 0.9 with 100 epochs. Similar to self-supervised training, we trained the classifier by using distributed data-parallel training on the four GPUs with 512 mini-batches on each GPU. The initial learning rate was 0.3, which was updated by using cosine annealing without restart [Loshchilov and Hutter, 2017] until the end.

Bound Evaluation We compared the extension bound of CURL (8) and our bound (10) by varying the number of negative samples K. We selected $K+1 \in \{32, 64, 128, 256, 512\}$ for CIFAR-10 and $K+1 \in \{128, 256, 384, 512, 640, 786, 896, 1024\}$ for CIFAR-100. After self-supervised learning, we approximated $\boldsymbol{\mu}_c$ by averaging 10 sampled data augmentations per sample on the training dataset and evaluated Equations (8) and (10) with the same negative sample size K as in self-supervised training. We reported the averaged values over validation samples with 10 epochs: $(\lfloor 10\,000/(K+1) \rfloor \times (K+1) \times \mathrm{epoch})$ pairs of $(\mathbf{z}, \mathbf{Z})$. Note that we used a theoretical value of υ defined by Eq. (9) to avoid dividing by zero if a realized υ value is 0. We also reported the upper bound of collision (11) without constants α, β referred to as "Collision Bound" on the training data. See Appendix D for details.

5.1 Experimental Results

Table 1 show the bound values on CIFAR-10 and CIFAR-100. We only showed a part of values among different numbers of negative samples due to the page limitation.[9] We reported mean and linear classifiers' validation accuracy as "μ acc" and "Linear acc", respectively. Since the CURL bound (8) does not contain $^{\dagger}L^{\mu}_{\mathrm{sup}}$ explicitly, we subtracted $^{\dagger}L^{\mu}_{\mathrm{sup}}$ from L^{μ}_{sub} in Eq. (8) and reported $^{\dagger}L^{\mu}_{\mathrm{sup}}$ and subtracted L^{μ}_{sub} as $^{\dagger}L^{\mu}_{\mathrm{sub}}$ for the comparison. We confirmed that CURL bounds converged to †Collision with relatively small K. On the other hand, proposed bound values had still a large proportion of supervised loss L^{μ}_{sup} with larger K. Figure 1 shows the linear accuracy and *upper* bounds of supervised loss L_{sup} by rearrange Equations (8) and (10). The reported values were averaged over three training runs of both self-supervised and supervised steps with different random seeds. The error bars represented the standard deviation.

[9] Tables 4 and 5 in Appendix D provides the comprehensive results.

6 Related Work

6.1 Self-supervised Representation Learning

Self-supervised learning tries to learn an encoder that extracts generic feature representations from an unlabeled dataset. Self-supervised learning algorithms solve a pretext task that does not require any supervision and can be easily constructed on the dataset, such as denoising [Vincent et al., 2008], colorization [Zhang et al., 2016, Larsson et al., 2016] solving jigsaw puzzles [Noroozi and Favaro, 2016], inpainting blank pixels [Pathak et al., 2016], reconstructing missing channels [Zhang et al., 2017], predicting rotation [Gidaris et al., 2018], and adversarial generative models [Donahue and Simonyan, 2019] for vision; predicting neighbor words [Mikolov et al., 2013], generating neighbor sentences [Kiros et al., 2015], and solving masked language model [Devlin et al., 2019] for language. See also recent review articles [Le-Khac et al., 2020, Schmarje et al., 2021].

Recent self-supervised learning algorithms for the vision domain mainly solve an instance discrimination task. `Exemplar-CNN` [Dosovitskiy et al., 2014] is one of the earlier algorithms that can obtain generic feature representations of images by using convolutional neural networks and data augmentation. After van den Oord et al. [2018] proposed InfoNCE loss function, many state-of-the-art self-supervised algorithms minimize InfoNCE-based loss function, e.g., `DeepInfoMax` [Hjelm et al., 2019], `AMDIM` [Bachman et al., 2019], `SimCLR` [Chen et al., 2020a], `CPCv2` [Hénaff et al., 2020], `MoCo` [He et al., 2020], and `SwAV` [Caron et al., 2020].

6.2 Theoretical Perspective

InfoNCE [van den Oord et al., 2018] was initially proposed as a lower bound of intractable mutual information between feature representations by using noise-contrastive estimation (NCE) [Gutmann and Hyvärinen, 2012]. Optimizing InfoNCE can be considered as maximizing the InfoMax principle [Linsker, 1988]. The history of InfoMax-based self-supervised representation learning dates back to more than 30 years ago [Hinton and Becker, 1990]. However, this interpretation does not directly explain the generalization for a downstream task. Indeed, Tschannen et al. [2020] empirically showed that the performances of classification on downstream tasks and mutual information estimation are uncorrelated with each other. Kolesnikov et al. [2019] also reported a similar relationship between the performances of linear classification on downstream tasks and of pretext tasks. McAllester and Stratos [2020] theoretically showed the limitation of maximizing the lower bounds of mutual information – accurate approximation requires an exponential sample size.

As the most related work, Arora et al. [2019] provide the first theoretical analyses to explain the generalization of the CURL. It is worth noting that our analysis focuses on the different representation learning setting. Bansal et al. [2021] decomposed the generalization error gap of a linear classifier given feature representations. Wang and Isola [2020] decomposed the self-supervised loss into alignment and uniformity and showed properties of both metrics. Tosh et al. [2021] also analyzed self-supervised loss for augmented samples, but they only focused on one negative sample setting. Recently, Wei et al. [2021] proposed learning theoretical analysis by introducing "expansion" assumption for self-training where (pseudo) labels are generated from the previously learned model. Learning theory-based analyses were also proposed for other types of self-supervised learning problem such as reconstruction task [Garg and Liang, 2020, Lee et al., 2020] and language modeling [Saunshi et al., 2021].

7 Conclusion

We applied the CURL framework to the recent self-supervised representation learning formulation. We pointed out that the existing framework has difficulty explaining why large negative samples in self-supervised learning improve classification accuracy on a downstream supervised task as in Figure 1. We proposed a novel framework using the coupon collector's problem to explain the phenomenon and confirmed our analysis on the CIFAR-10/100 datasets.

Limitations We did not discuss the properties of data augmentation explicitly in our framework. Practically, self-supervised representation learning algorithms discard the projection head after self-supervised learning, but our analysis does not cover this procedure. We believe that extensions to cover these settings are fruitful explorations of future work.

References

S. Arora, H. Khandeparkar, M. Khodak, O. Plevrakis, and N. Saunshi. A Theoretical Analysis of Contrastive Unsupervised Representation Learning. In *ICML*, pages 5628–5637, 2019.

P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning Representations by Maximizing Mutual Information Across Views. In *NeurIPS*, pages 15535–15545, 2019.

Y. Bansal, G. Kaplun, and B. Barak. For Self-Supervised Learning, Rationality Implies Generalization, Provably. In *ICLR*, 2021.

H. Bao, G. Niu, and M. Sugiyama. Classification from Pairwise Similarity and Unlabeled Data. In *ICML*, pages 452–461, 2018.

Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. Mccandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. In *NeurIPS*, pages 1877–1901, 2020.

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep Clustering for Unsupervised Learning of Visual Features. In *ECCV*, pages 139–156, 2018.

M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *NeurIPS*, pages 9912–9924, 2020.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, pages 1597–1607, 2020a.

T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton. Big Self-Supervised Models are Strong Semi-Supervised Learners. In *NeurIPS*, 2020b.

X. Chen and K. He. Exploring Simple Siamese Representation Learning. In *CVPR*, 2021.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, pages 248–255, 2009.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

J. Donahue and K. Simonyan. Large Scale Adversarial Representation Learning. In *NeurIPS*, pages 10542–10552, 2019.

A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative Unsupervised Feature Learning with Convolutional Neural Networks. In *NeurIPS*, pages 766–774, 2014.

R. Durrett. *Probability: Theory and Examples*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 5 edition, 2019.

P. Flajolet, D. Gardy, and L. Thimonier. Birthday Paradox, Coupon Collectors, Caching Algorithms and Self-organizing Search. *Discrete Applied Mathematics*, 39(3):207–229, 1992.

S. Garg and Y. Liang. Functional Regularization for Representation Learning: A Unified Theoretical Perspective. In *NeurIPS*, pages 17187–17199, 2020.

S. Gidaris, P. Singh, and N. Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. In *ICLR*, 2018.

J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap Your Own Latent A New Approach to Self-Supervised Learning. In *NeurIPS*, pages 21271–21284, 2020.

M. U. Gutmann and A. Hyvärinen. Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *Journal of Machine Learning Research*, 13:307–361, 2012.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*, pages 9726–9735, 2020.

O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. Van den oord. Data-Efficient Image Recognition with Contrastive Predictive Coding. In *ICML*, pages 4182–4192, 2020.

G. E. Hinton and S. Becker. An Unsupervised Learning Procedure that Discovers Surfaces in Random-dot Stereograms. In *IJCNN*, pages 218–222, 1990.

R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning Deep Representations by Mutual Information. In *ICLR*, 2019.

J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, pages 448–456, 2015.

R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-Thought Vectors. In *NeurIPS*, pages 3294–3302, 2015.

A. Kolesnikov, X. Zhai, and L. Beyer. Revisiting Self-Supervised Visual Representation Learning. In *CVPR*, pages 1920–1929, 2019.

S. Kornblith, J. Shlens, and Q. V. Le. Do Better ImageNet Models Transfer Better? In *CVPR*, pages 2661–2671, 2019.

A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.

B. Kulis. Metric Learning: A Survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2012.

G. Larsson, M. Maire, and G. Shakhnarovich. Learning Representations for Automatic Colorization. In *ECCV*, pages 577–593, 2016.

P. H. Le-Khac, G. Healy, and A. F. Smeaton. Contrastive Representation Learning: A Framework and Review. *IEEE Access*, 8:193907–193934, 2020.

J. D. Lee, Q. Lei, N. Saunshi, and J. Zhuo. Predicting What You Already Know Helps: Provable Self-Supervised Learning. *arXiv:2008.01064v1 [cs.LG]*, 2020.

J. Li, P. Zhou, C. Xiong, R. Socher, and S. C. Hoi. Prototypical Contrastive Learning of Unsupervised Representations. In *ICLR*, 2021.

R. Linsker. Self-Organization in a Perceptual Network. *Computer*, 21(3):105–117, 1988.

L. Logeswaran and H. Lee. An Efficient Framework for Learning Sentence Representations. In *ICLR*, 2018.

I. Loshchilov and F. Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR*, 2017.

D. McAllester and K. Stratos. Formal Limitations on the Measurement of Mutual Information. In *AISTATS*, pages 875–884, 2020.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *NeurIPS*, pages 3111–3119, 2013.

T. Nakata and I. Kubo. A Coupon Collector's Problem with Bonuses. In *Fourth Colloquium on Mathematics and Computer Science*, pages 215–224, 2006.

A. Newell and J. Deng. How Useful is Self-Supervised Pretraining for Visual Tasks? In *CVPR*, pages 7345–7354, 2020.

M. Noroozi and P. Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *ECCV*, pages 69–84, 2016.

K. Nozawa, P. Germain, and B. Guedj. PAC-Bayesian Contrastive Unsupervised Representation Learning. In *UAI*, pages 21–30, 2020.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, pages 8024–8035, 2019.

D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context Encoders: Feature Learning by Inpainting. In *CVPR*, pages 2536–2544, 2016.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

J. Reback, W. McKinney, jbrockmendel, J. V. den Bossche, T. Augspurger, P. Cloud, gfyoung, Sinhrks, A. Klein, M. Roeschke, S. Hawkins, J. Tratner, C. She, W. Ayd, T. Petersen, M. Garcia, J. Schendel, A. Hayden, MomIsBestFriend, V. Jancauskas, P. Battiston, S. Seabold, chris-b1, h-vetinari, S. Hoyer, W. Overmeire, alimcmaster1, K. Dong, C. Whelan, and M. Mehyar. pandas-dev/pandas: Pandas 1.0.3, Mar. 2020. URL `https://doi.org/10.5281/zenodo.3715232`.

N. Saunshi, S. Malladi, and S. Arora. A Mathematical Exploration of Why Language Models Help Solve Downstream Tasks. In *ICLR*, 2021.

L. Schmarje, M. Santarossa, S.-M. Schröder, and R. Koch. A Survey on Semi-, Self-and Unsupervised Learning in Image Classification. *IEEE Access*, 2021.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the Importance of Initialization and Momentum in Deep Learning. In *ICML*, pages 1139–1147, 2013.

O. Tange. GNU Parallel, Nov. 2020. URL `https://doi.org/10.5281/zenodo.4284075`.

M. K. Titsias. One-vs-Each Approximation to Softmax for Scalable Estimation of Probabilities. In *NeurIPS*, pages 4168–4176, 2016.

C. Tosh, A. Krishnamurthy, and D. Hsu. Contrastive Learning, Multi-view Redundancy, and Linear Models. In *ALT*, pages 1179–1206, 2021.

M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic. On Mutual Information Maximization for Representation Learning. In *ICLR*, 2020.

A. van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv:1807.03748v2 [cs.LG]*, 2018.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *ICML*, pages 1096–1103, 2008.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

T. Wang and P. Isola. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *ICML*, pages 9929–9939, 2020.

M. L. Waskom. seaborn: Statistical Data Visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL `https://doi.org/10.21105/joss.03021`.

C. Wei, K. Shen, Y. Chen, and T. Ma. Theoretical Analysis of Self-Training with Deep Networks on Unlabeled Data. In *ICLR*, 2021.

Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In *CVPR*, pages 3733–3742, 2018.

O. Yadan. Hydra - A Framework for Elegantly Configuring Complex Applications. GitHub, 2019. URL `https://github.com/facebookresearch/hydra`.

Y. You, I. Gitman, and B. Ginsburg. Large Batch Training of Convolutional Networks. *arXiv:1708.03888v3 [cs.CV]*, 2017.

R. Zhang, P. Isola, and A. A. Efros. Colorful Image Colorization. In *ECCV*, pages 649–666, 2016.

R. Zhang, P. Isola, and A. A. Efros. Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction. In *CVPR*, pages 1058–1067, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.1.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The experimental code and instructions are found in supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5, where we used the original training and test splits for both CIFAR-10 and CIFAR-100 datasets. The hyper-parameters were chosen by following the experiments in Chen et al. [2020a].
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The validation accuracy values on Figure 1 are shown with their standard deviation values.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] But Section 5 mentioned the type of GPUs was NVIDIA Tesla P100 on an internal cluster.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5, where we cited the creators of datasets and the libraries used in our experiments.
   (b) Did you mention the license of the assets? [Yes] The license of our experimental codes is the MIT license and it is as the anonymized license file in the supplimental material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We included the experimental codes in the supplimental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We used CIFAR-10/100 datasets in our experiments.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We used CIFAR-10/100 datasets in our experiments.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]