Understanding Negative Samples in Instance Discriminative Self-supervised Representation Learning

Kento Nozawa
The University of Tokyo & RIKEN AIP
nzw@g.ecc.u-tokyo.ac.jp

Issei Sato
The University of Tokyo
sato@g.ecc.u-tokyo.ac.jp

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., 2020, Chen et al., 2018a, 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,
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 \( \{x_i\}_{i=1}^N \), where input \( x \in \mathcal{X} \). SimCLR learns feature extractor \( f : \mathcal{X} \rightarrow \mathbb{R}^h \) modeled by neural networks on the dataset, where \( h \) is the dimensionality of feature representation. Let \( z = f(a(x)) \) that is a feature representation of \( x \) after applying data augmentation \( a : \mathcal{X} \rightarrow \mathcal{X} \). Data augmentation is a pre-defined stochastic function such as a composition of the horizontal flipping and cropping. Note that the output of \( f \) is normalized by its L2 norm. Let \( z^+ = f(a^+(x)) \) that is a positive feature representation created from \( x \) with different data augmentation \( a^+(\cdot) \).

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

\[
\ell_{\text{Info}}(z, Z) := -\ln \frac{\exp (z \cdot z^+/t)}{\sum_{z_k \in Z} \exp (z \cdot z_k/t)},
\]

(1)

where \( Z = \{z^+, z_1^-, \ldots, z_K^-\} \) that is a set of positive representation \( z^+ \) and \( K \) negative representations \( \{z_1^-, \ldots, z_K^-\} \) created from other samples, \((\cdot, \cdot)\) is an inner product of two representations, and \( t \in \mathbb{R}_+ \) is a temperature parameter.\(^2\) 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 \( f \) learns an invariant encoder with respect to data augmentation \( a \). After minimizing Eq. (1), \( f \) 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

\(^1\)In fact, our theoretical analysis is valid with asymmetric feature extractors such as MoCo [He et al., 2020], where the positive and negative features do not come from feature extractor \( f \).

\(^2\)We perform the analysis with \( t = 1 \) for simplicity, but our analysis holds with any temperature.
in Figure 1 from the existing analysis. Table 2 in Appendix A summarizes notations used in this paper for convenience.

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 $f$ 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.\footnote{Technically, we do not need $c$ to draw $x$ 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. In the main analysis, we assume that the supervised classes in the downstream task are subset of $\mathcal{C}$; however, different relationship between supervised and latent classes is discussed in Appendix B.}

Let $\rho$ be a probability distribution over $\mathcal{C}$, and let $x$ be an input sample drawn from a data distribution $D_c$ conditioned on a latent class $c \in \mathcal{C}$. We draw two data augmentations $(a, a^+)$ from the distribution of data augmentation $\mathcal{A}$ and apply them independently to the sample. As a result, we have two augmented samples $(a(x), a^+(x))$ and call $a^+(x)$ as a positive sample of $a(x)$. Similarly, we draw $K$ negative samples from $D^{-}_{c_k}$ for each $c_k \in \{c_1, \ldots, c_{\rho^K}\}$ and $K$ data augmentations from $\mathcal{A}$, and then we apply them to negative samples. Definition 1 summarizes the data generation process. Note that we suppose data augmentation $a \sim \mathcal{A}$ does not change the latent class of $x \in \mathcal{X}$.

**Definition 1** (Data Generation Process in Self-supervised Representation Learning Step).

1. Draw latent classes: $c \sim \rho^K$.
2. Draw input sample: $x \sim D_c$.
3. Draw data augmentations: $(a, a^+) \sim \mathcal{A}^2$.
4. Apply data augmentations: $a(x), a^+(x)$.
5. Draw negative samples: $\{x^{-}_{k}\}_{k=1}^{K} \sim D^{-}_{c_k}$.
6. Draw data augmentations: $\{a^{-}_{k}\}_{k=1}^{K} \sim \mathcal{A}^K$.
7. Apply data augmentations: $\{a^{-}_{k} (x^{-}_{k})\}_{k=1}^{K}$.

Note that the original data generation process of CURL samples a positive sample of $x$ from $D_c$ 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).

$$L_{\text{Info}}(f) := \mathbb{E}_{c,\{c^{-}_{k}\}_{k=1}^{K} \sim \rho^K} \mathbb{E}_{x \sim D_c} \mathbb{E}_{(a, a^+) \sim \mathcal{A}^2} \mathbb{E}_{\{x^{-}_{k} \sim D^{-}_{c_k}\}_{k=1}^{K}} \mathbb{E}_{\{a^{-}_{k}\}_{k=1}^{K}} \ell_{\text{Info}}(z, Z),$$

where recall that $z = f(a(x))$ and $Z = \{f(a^+(x)), f(a^{-}_{1} (x^{-}_{1})), \ldots, f(a^{-}_{K} (x^{-}_{K}))\}$.

Since we cannot directly minimize Eq. (2), we minimize the empirical counterpart, $\hat{L}_{\text{Info}}(f)$, by sampling from the training dataset. After minimizing $\hat{L}_{\text{Info}}(f)$, learned $\hat{f}$ 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 $y \in \mathcal{Y} = \{1, \ldots, Y\}$ for each $x$ that is used in the self-supervised learning step. Let $\mathcal{S}$ be a supervised data distribution over $\mathcal{X} \times \mathcal{Y}$, and $g \circ \hat{f} : \mathcal{X} \rightarrow \mathbb{R}^Y$ be a classifier, where $g : \mathbb{R}^h \rightarrow \mathbb{R}^Y$ and $\hat{f}$ is the frozen feature extractor. Given the
supervised data distribution and classifier, our goal is to minimize the following supervised loss:
\[
L_{sup}(g \circ \hat{f}) := \mathbb{E}_{x, y \sim D} - \ln \frac{\exp \left( g_y(f(a(x))) \right)}{\sum_{j \in Y} \exp \left( g_j(f(a(x))) \right)}.
\] (3)

By following the CURL analysis, we introduce a mean classifier as a simple instance of \( g \) because its loss is an upper bound of Eq. (3).\(^4\) The mean classifier is the linear classifier whose weight of label \( y \) is computed by averaging representations: \( \mu_y = \mathbb{E}_{x \sim D_y}\mathbb{E}_{a \sim A} f(a(x)), \) where \( D_y \) 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),
\[
L^\mu_{sup}(\hat{f}) := \mathbb{E}_{x, y \sim S} - \ln \frac{\exp \left( \hat{f}(a(x)) \cdot \mu_y \right)}{\sum_{j \in Y} \exp \left( \hat{f}(a(x)) \cdot \mu_j \right)}.
\] (4)

We also introduce a sub-class loss function.\(^5\) Let \( Y_{\text{sub}} \) be a subset of \( Y \) and \( S_{\text{sub}} \) be a data distribution over \( X \times Y_{\text{sub}} \), then we define the sub-class losses of classifier \( g \) and mean classifier:

**Definition 4** (Supervised Sub-class Losses of Classifier \( g \) and Mean Classifier with \( \hat{f} \)),
\[
L_{sub}(g \circ \hat{f}, Y_{\text{sub}}) := \mathbb{E}_{x, y \sim S_{\text{sub}}} - \ln \frac{\exp \left( g_y(f(a(x))) \right)}{\sum_{j \in Y_{\text{sub}}} \exp \left( g_j(f(a(x))) \right)},
\] (5)
\[
L^\mu_{sub}(\hat{f}, Y_{\text{sub}}) := \mathbb{E}_{x, y \sim S_{\text{sub}}} - \ln \frac{\exp \left( \hat{f}(a(x)) \cdot \mu_y \right)}{\sum_{j \in Y_{\text{sub}}} \exp \left( \hat{f}(a(x)) \cdot \mu_j \right)}.
\] (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 \( \mu(x) = \mathbb{E}_{a \sim A} f(a(x)) \) and derive a lower bound of unsupervised loss \( L_{\text{Info}}(f) \).

\[
L_{\text{Info}}(f) \geq \sum_{c, (e_k)} E_{k=1}^{K+1} \mathbb{E}_{x \sim D_c} \mathbb{E}_{a \sim A} \left[ \mathbb{E}_{e_k \sim D_e} \ell_{\text{Info}} \left( f(a(x)), \{ \mu(x), \mu(x^-), \ldots, \mu(x^-) \} \right) \right]
\]
\[
\geq \sum_{c, (e_k)} E_{k=1}^{K+1} \mathbb{E}_{x \sim D_c} \mathbb{E}_{a \sim A} \left[ \ell_{\text{Info}} \left( f(a(x)), \{ \mu(x), \mu(x), \ldots, \mu(x^-) \} \right) \right] + d(f),
\] (7)

where \( d(f) = \sum_{L \in c} E_{e \sim D_e} \mathbb{E}_{x \sim D_c} \left[ \mathbb{E}_{a \sim A} \left[ f(a(x)) \right] \mathbb{E}_{x \sim D_e} \left[ f(a(x)^+) \right] - \mathbb{E}_{a \sim A} \left[ f(a(x)^+) \right] \right] \).

\(^4\) Concrete inequality is found in Appendix C.

\(^5\) Arora et al. [2019] refer to this loss function as averaged supervised loss.
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 D.1. It is worth noting that positive and negative features can be extracted from another feature encoder or memory back He et al. [2020].

**Remark 5** (Effect of gap term $d(f)$). We confirm that $d(f)$ 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: $\tau_K = \mathbb{P}(\text{Col}(c, \{c_k\}_{k=1}^K) \neq 0)$, where $\text{Col}(c, \{c_k\}_{k=1}^K) = \sum_{k=1}^K \mathbb{I}[c = c_k]$ and $\mathbb{I}[]$ is the indicator function. We omit the arguments of Col for simplicity. Let $\text{Col}_{\text{sub}}(\{c, c_1, \ldots, c_K\})$ be a function to remove duplicated latent classes given latent classes. We omit the arguments of $\text{Col}_{\text{sub}}$ as well. The result is an 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 $f$,

$$L_{\text{Info}}(f) \geq (1 - \tau_K) \mathbb{E}_{c, \{c_k\}_{k=1}^K \sim \rho^{K+1}} \left[ L_{\text{sub}}^\mu(f, \text{Col}_{\text{sub}}) | \text{Col} = 0 \right] + \tau_K \mathbb{E}_{c, \{c_k\}_{k=1}^K \sim \rho^{K+1}} \left[ \ln(\text{Col} + 1) | \text{Col} \neq 0 \right] + d(f).$$

(8)

The proof is found in Appendix D.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 $\tau_K$. 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 $C_{\text{sub}} \supseteq \mathcal{Y}$. 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 $\tau_K$ 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 $\tau$ with a different probability since $\tau$ 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 $\upsilon_K$ be a probability that sampled $K$ latent classes contain all latent classes: $\{c_k\}_{k=1}^K \supseteq C$. This probability appears in the coupon collector’s problem of probability theory (e.g., Durrett [2019, 2020]).

$$L_{\text{Info}}(f) \geq (1 - \upsilon_K) \mathbb{E}_{c, \{c_k\}_{k=1}^K \sim \rho^{K+1}} \left[ L_{\text{sub}}^\mu(f, \text{Col}_{\text{sub}}) | \text{Col} = 0 \right] + \upsilon_K \mathbb{E}_{c, \{c_k\}_{k=1}^K \sim \rho^{K+1}} \left[ \ln(\text{Col} + 1) | \text{Col} \neq 0 \right] + d(f).$$

(9)

Theorem 4.1 (CURL Lower Bound of Self-supervised Loss). For all feature extractor $f$,
Example 2.2.7]). If the latent class probability is uniform: \( \forall c, \rho(c) = 1/|C| \), then we can calculate the probability explicitly as follows.

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

\[
v_K := \sum_{n=1}^{K} \sum_{m=0}^{|C|-1} \left( \frac{|C| - 1}{m} \right) (-1)^m \left( 1 - \frac{m + 1}{|C|} \right)^{n-1}, 
\]

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

By replacing \( \tau \) with \( v \), we obtain our lower bound of InfoNCE loss:

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

\[
L_{\text{Info}}(f) \geq \frac{1}{2} \left\{ v_{K+1} \mathbb{E}_{c,\{c_k^{-}\}} \mathbb{E}_{k=1}^{K+1} \left[ L_{\text{sub}}^{\mu}(f, C) \mid C_{\text{sub}} = C \right] \right. \\
+ (1 - v_{K+1}) \mathbb{E}_{c,\{c_k^{-}\}} \mathbb{E}_{k=1}^{K+1} \left[ L_{\text{sub}}^{\mu}(f, C_{\text{sub}}) \mid C_{\text{sub}} \neq C \right] \\
+ \mathbb{E}_{c,\{c_k^{-}\}} \mathbb{E}_{k=1}^{K+1} \left[ \ln(Col + 1) \right] + d(f). 
\]

The proof is found in Appendix D.3.

Theorem 8 tells us that probability \( v_{K+1} \) 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 \( v \) with the same setting discussed in Section 3.3. When \( v_{K=32} \approx 0.719 \), 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 \( \{c_k^{-}\}_{k=1}^{K} \), and feature extractor \( f \),

\[
\ln \left( \text{Col} \left( c, \{c_k^{-}\}_{k=1}^{K} \right) + 1 \right) \leq \alpha + \beta \mathbb{E}_{x \sim D_c} \mathbb{E}_{x' \sim D'_c} \mathbb{E}_{(a,a') \sim A^2, a' \sim A} \left[ f(a(x)) \cdot \left[ f(a'(x')) - f(a^+(x)) \right] \right], 
\]

where \( \alpha \) and \( \beta \) are non-negative constants depending on the number of duplicated latent classes.

The proof is found in Appendix D.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 \( f(a(x)) \) and \( f(a^+(x)) \) are computed from the same input sample with different data augmentations, their inner product tends to be 1; thus, \( f(a'(x')) \) is located in the opposite direction of \( f(a(x)) \). 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 \( \alpha \) and \( \beta \) does not increase by increasing \( K \) (see Table 1 and Appendix F for more analysis).

---

*We show expected \( K+1 \) to draw all supervised labels for ImageNet-1K and all used datasets in Appendix E.
4.3 Small $K$ can Give Consistent Loss Function for Supervised Task

As we shown in Theorem 8, $L_{\text{Info}}$ with large $K$ can be viewed as an upper bound of $L_{\sup}^{\mu}$. However, $L_{\text{Info}}$ 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.\footnote{Arora et al. [2019, Table D.1] mentioned that this phenomenon is not covered by the CURL framework.} 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 $L_{\text{sub}}$). Suppose $C \supseteq Y$ and $\rho$ is uniform: $\forall c \in C, \rho(c) = 1/|C|$. Suppose a constant function $q: x \in \mathcal{X} \mapsto [q_1, \ldots, q_{|C|}]^T$. Optimal $q^*$ is a constant function that outputs a vector with the same value if and only if it minimizes $\mathbb{E}_{c,\{(x_k^c)_{k=1}^{K}\} \sim \rho^{K+1}} L_{\text{sub}}(q^*, C_{\text{sub}})$. The optimal $q^*$ is also the minimizer of $L_{\sup}$.

The proof is found in Appendix D.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 $L_{\text{Info}}$, we cannot minimize $L_{\sup}^{\mu}$ 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], SeLa [Asano et al., 2020], SwAV [Caron et al., 2020], and PCL [Li et al., 2021a], use prototype representations instead of $z^+$, $\{z_k^c\}_{k=1}^{K}$ by applying unsupervised clustering on feature representations. This procedure is justified as the approximation of latent class’s mean representation $\mu_c$ 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 F contains NLP experiments and some analyses that have been omitted due to the lack of space. We used datasets with a relatively small number of classes to compare bounds. Our experimental codes are available online.\footnote{https://github.com/nzw0301/Understanding-Negative-Samples. 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.}

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.\footnote{https://github.com/google-research/simclr} 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 [Li et al., 2020] 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][10], 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 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 $z$ 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 $g$, we optimized $g$ 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 $\mu_c$ 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: $(10000/(K+1)) \times (K+1) \times \text{epoch})$ pairs of $(z, Z)$. Note that we used a theoretical value of $\upsilon$ defined by Eq. (9) to avoid dividing by zero if a realized $\upsilon$ value is 0. We also reported the upper bound of collision (11) without constants $\alpha, \beta$ referred to as “Collision Bound” on the training data. See Appendix F for details.

### 5.1 Experimental Results

Table 1 shows 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.[12] We reported mean and linear classifiers’ validation accuracy as “$\mu$ acc” and “Linear acc”, respectively. As a reference, we reported practical linear evaluation’s validation accuracy as “Linear acc w/o”, where we discarded the non-linear projection head from the feature extractor [Chen et al., 2020a]. Since the CURL bound (8) does not contain $L_{sup}^\mu$ explicitly, we subtracted $L_{sup}^\mu$ from $L_{sub}^\mu$ in Eq. (8) and reported $L_{sup}^\mu$ and subtracted $L_{sub}^\mu$ as $L_{sup}^{\mu\mu}$ for the comparison. We confirmed that CURL bounds converged to $L_{collision}$ with relatively small $K$. On the other hand, proposed bound values had still a large proportion of supervised loss $L_{sup}^\mu$ with larger $K$. Figure 1 shows upper bounds of supervised loss $L_{sup}$ and the linear accuracy by rearranging 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.

---

[10]https://github.com/NVIDIA/apex

[11] Precisely, $\mu_c = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{10} \sum_{j=1}^{10} f(a_j(x_i))$.

[12] Tables 4 and 5 in Appendix F provides the comprehensive results.
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.

<table>
<thead>
<tr>
<th>$K + 1$</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32</td>
<td>128</td>
</tr>
<tr>
<td>$\tau$</td>
<td>0.96</td>
<td>1.00</td>
</tr>
<tr>
<td>$\nu$</td>
<td>0.69</td>
<td>1.00</td>
</tr>
<tr>
<td>$\mu$ acc</td>
<td>72.75</td>
<td>77.22</td>
</tr>
<tr>
<td>Linear acc</td>
<td>77.13</td>
<td>81.33</td>
</tr>
<tr>
<td>Linear acc w/o</td>
<td>82.02</td>
<td>85.43</td>
</tr>
<tr>
<td>$L_{\text{inf}}$</td>
<td>Eq. (2)</td>
<td>2.02</td>
</tr>
<tr>
<td>$d(f)$ Eq. (7)</td>
<td>$-1.16$</td>
<td>$-1.18$</td>
</tr>
<tr>
<td>†Collision bound Eq. (8)</td>
<td>0.23</td>
<td>1.41</td>
</tr>
<tr>
<td>†$L_{\text{inf}}^\mu$</td>
<td>1.32</td>
<td>2.58</td>
</tr>
<tr>
<td>†$L_{\text{inf}}^{\nu_{\text{sub}}}$</td>
<td>0.05</td>
<td>0.00</td>
</tr>
<tr>
<td>$L_{\text{inf}}$ bound Eq. (10)</td>
<td>0.39</td>
<td>1.02</td>
</tr>
<tr>
<td>$L_{\text{inf}}^{\nu_{\text{sup}}}$</td>
<td>0.63</td>
<td>0.91</td>
</tr>
<tr>
<td>$L_{\text{inf}}$</td>
<td>0.26</td>
<td>0.00</td>
</tr>
<tr>
<td>Collision bound Eq. (11)</td>
<td>0.60</td>
<td>0.61</td>
</tr>
</tbody>
</table>

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. Shortly after our publishing a draft of this paper on arXiv, Ash et al. [2021] also
published a paper on the role of negative samples in CURL bound with InfoNCE loss for fully supervised classification using the coupon collector’s problem. Thus we believe that the coupon collector’s problem is a key ingredient to analyze the connection between contrastive learning and supervised classification based on the CURL analysis by Arora et al. [2019]. Their work was developed independently of ours and their analysis is for contrastive unsupervised learning rather than self-supervised learning that is done in this work. The proposed bound by Ash et al. [2021] has also similar issue to Arora et al. [2019] as in our analysis; however, their bound holds with smaller $K$ than $C$. 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. Li et al. [2021b] provided an interpretation of InfoNCE through a lens of kernel method. Mitrovic et al. [2021] provided another theoretical analysis with causality for instance discriminative self-supervised learning. 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., 2021] and language modeling [Saunshi et al., 2021]. The theoretical analysis on self-supervised representation algorithms without negative samples [Tian et al., 2021] cannot be applied to the contrastive learning setting.

6.3 Hard Negative Mining

In metric learning and contrastive learning, hard negative mining, such as Kalantidis et al. [2020], is actively proposed to make training more effective to avoid using inappropriate negative or too easy negative samples. The current work mainly focuses on the quality of negative samples rather than quantity of negative samples. However, removing false-negative samples can reduce the effect of the collision term in our bounds. Thus our analysis might provide a theoretical justification for hard negative mining.

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 real-world benchmark 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.

Acknowledgments

This work is supported (in part) by Next Generation AI Research Center, The University of Tokyo. The experiments were conducted using the SGI Rackable C2112-4GP3/C1102-GP8 (Reedbush-H/L) in the Information Technology Center, The University of Tokyo. We thank Han Bao, Yoshihiro Nagano, and Ikko Yamane for constructive discussion and Yusuke Tsuzuku for supporting our experiments. We also thank Junya Honda and Yivan Zhang for \LaTeX{} support, specifically, for solving our font issue and MathJax, respectively. We appreciate anonymous reviewers of ICML 2021 and NeurIPS 2021 for giving constructive suggestions to improve our manuscript. KN is supported by JSPS KAKENHI Grant Number 18J20470.

References


