DATA-EFFICIENT CONTRASTIVE LEARNING BY DIFFERENTIABLE HARD SAMPLE AND HARD POSITIVE PAIR GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Contrastive learning (CL), a self-supervised learning approach, can effectively learn visual representations from unlabeled data. However, CL requires learning on vast quantities of diverse data to achieve good performance, without which the performance of CL will greatly degrade. To tackle this problem, we propose a framework with two approaches to improve the data efficiency of CL training by generating beneficial samples. The first approach generates hard samples for the main model. In the training process, hard samples are dynamically customized to the training state of the main model, rather than fixed throughout the whole training process. With the progressively growing knowledge of the main model, the generated samples also become harder to constantly encourage the main model to learn better representations. Besides, a pair of data generators are proposed to generate similar but distinct samples as positive pairs. The hardness of positive pair is progressively increased by decreasing the similarity between a positive pair. In this way, the main model learns to cluster hard positives by pulling the representations of similar yet distinct samples together, by which the representations of similar samples are well-clustered and better representations can be learned. Comprehensive experiments show superior accuracy of the proposed approaches over the state-of-the-art on multiple datasets. For example, about 5% and 6% improvements are achieved on CIFAR-10/100 and ImageNet-100/10 with limited training data, respectively. Besides, about 2$\times$ data efficiency is achieved when reaching a similar accuracy as other methods.

1 INTRODUCTION

Contrastive learning (CL), a highly effective self-supervised learning approach (Chen et al., 2020; He et al., 2020; Kalantidis et al., 2020), has recently shown great promise to learn visual representations from unlabeled data. CL performs a proxy task of instance discrimination to learn data representations without requiring labels, leading to well-clustered and transferable representations for downstream tasks. In the proxy task, the representations of two transformations of one image (a positive pair) are pulled close to each other and pushed away from the representations of other samples (negatives), by which high-quality representations are learned (Kalantidis et al., 2020).

Most state-of-the-art CL methods focus on designing variants of the proxy task (Caron et al., 2020; Zbontar et al., 2021; Chen & He, 2020; Grill et al., 2020), while the underlying data availability for training CL has received less emphasis. CL heavily relies on vast quantities of diverse and high-quality training examples. CL models are usually trained on large-scale datasets with millions of samples (He et al., 2020; Chen et al., 2020; Caron et al., 2020). For example, the ImageNet dataset (Russakovsky et al., 2015) of general natural images widely used in CL has more than a million images with various object categories. Collecting such large-scale datasets require months or even years of considerable human efforts (Zhao et al., 2020). Besides, for many application domains, collecting a sufficiently large-scale dataset could be prohibitive and even infeasible due to imaging cost, privacy, and copyright constraints. Examples of such domains include medical images, images from scientific experiments, or images from rare natural phenomenons (Chen et al., 2021).

Training CL with scarce samples leads to drastic degradation in the quality of learned representations because of insufficient contrast between diverse samples. More specifically, CL is trained by performing a proxy task to discriminate two transformations of one image from other images (Caron et al., 2020). Without diverse samples to discriminate from, the proxy task becomes relatively easy and the model can achieve high accuracy on this simple task without the need to learn high-quality discriminative representations (Kalantidis et al., 2020). The poor quality of learned representations will further degrade the performance of downstream tasks. For example, when training CL on the CIFAR-10 dataset without using labels and then fine-tuning the trained model on 1% labeled data (500 samples), the classification accuracy reaches 84.99%. However, training CL with only 10% of the CIFAR-10 dataset without labels, and then also fine-tuning with 500 labeled samples, the accuracy dramatically decreases to 68.77%, which is substantially lower than training CL on the full dataset. Therefore, it is crucial to achieving data-efficient CL training in the absence of large datasets to make CL applicable to more scenarios.

Towards this goal, we propose a framework to generate effective data for CL training based on small datasets. The framework consists of two approaches. The first approach generates hard samples for the main model. The generated samples are dynamically adapted to the training state of the main model, rather than fixed throughout the whole training process. With the progressively growing knowledge of the main model, the generated samples also become harder to encourage the main model to learn better representations. The generated samples adversarially explore the weakness of the main model, which forces it to learn discriminative features from these hard samples and improves the generalization of learned features.

The second approach generates two similar but distinct images as hard positive pairs. Existing CL frameworks form a positive pair by applying two data transformations (e.g. cropping and color distortions) to one image to generate two transformed images. While the two transformed images look different, they still share the same identity since they originate from one image. Only clustering these positive pairs will limit the quality of learned representation since other similar objects are not considered in the clustering process. We form positive pairs by generating two images of distinct but similar objects without using labels, which is achieved by using a generator and its slowly evolving version. In this way, the main model has to learn to cluster hard positives when minimizing the contrastive loss. By pulling the representations of similar but distinct (hard) objects together, better clustering of the representation space can be learned (Khosla et al., 2020). With better representations, the performance of downstream tasks will also be improved.

In summary, the main contributions of the paper include:

- **Data-efficient contrastive learning framework.** We propose a framework with two approaches for data-efficient contrastive learning by jointly optimizing contrastive learning and hard samples generation. The first approach generates hard samples and the second approach generates hard positive pairs without using labels.
- **Dynamic hard samples generation.** We propose an approach to generate hard samples by dynamically tracking the training state of the main model. In the training process, hard samples are customized on-the-fly to the progressive knowledge of the main model, which are fed into the main model to constantly encourage the main model to learn better representations.
- **Hard positive pair generation without using labels.** We propose an approach to further generate hard positive pairs without leveraging labels. The generator and its slowly evolving version generate a pair of similar but distinctive objects as positive pairs. The hardness of positive pair is further increased by decreasing the similarity between a positive pair. In this way, similar objects are well-clustered and better representations can be learned.

2 Background and Related Work

Revisiting Contrastive Learning. Contrastive learning is a self-supervised approach to learn an encoder for extracting visual representations from the unlabeled images by performing a proxy task of instance discrimination (Chen et al., 2020; He et al., 2020; Wu et al., 2018).

Our work in this paper is built upon SimCLR (Chen et al., 2020), which is a simple yet powerful contrastive learning approach for unsupervised representation learning. For an input image x, its representation vector v is obtained by $v = f(x, \theta),\ z \in \mathbb{R}^d$, where $f(\cdot, \theta)$ is the encoder with parameters θ. In the training process of CL, a raw batch of N samples $\{x_k\}_{k=1\ldots N}$ are first randomly sampled from the dataset. Then for each raw sample x_k, two transformations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) sampled from a family of augmentations $\mathcal{T}$ (e.g. cropping, color distortion, etc.) are applied to x_k to generate two transformed samples (a.k.a. two views). $\tilde{x}_{2k-1} = t(x_k)$ and $\tilde{x}_{2k} = t'(x_k)$ are a positive pair, and all of the generated pairs form the batch for training $\{\tilde{x}_l\}_{l=1...2N}$, consisting of $2N$ samples (Khosla et al., 2020). In the remainder of this paper, we will refer to the set of N samples as a *raw batch* and the set of $2N$ transformed samples as a *multiviewed batch*.

In the multiviewed batch, let $i \in I = \{1...2N\}$ be the index of a transformed sample, and let $j(i)$ be the index of the sample originating from the same raw sample as i. The contrastive loss is as follows.

$$\mathcal{L}_{CL} = \sum_{i \in I} \mathcal{L}_i = -\sum_{i \in I} \log \frac{\exp\left(v_i \cdot v_{j(i)}/\tau\right)}{\sum_{a \in A(i)} \exp(v_i \cdot v_a/\tau)} \tag{1}$$

where $v_i = f(\tilde{x}_i, \theta)$, the operator $\cdot$ is the inner product to compute the cosine similarity of two vectors, τ is the temperature. The index i is the anchor. $A(i) = I \backslash \{i\}$ is the set of indices excluding i. For each anchor i, there is one positive and $2N-2$ negatives. The index $j(i)$ is the positive to i (i.e. a positive pair $(i, j(i))$), while other $2N-2$ indices $\{k \in A(i) \backslash \{j(i)\}\}$ are the negatives.

Existing works focus on developing contrastive learning methods while assuming vast quantities of training data is available, without which the performance will drastically degrade. Different from these works, we aim to achieve data-efficient CL on scare samples by dynamically customizing hard samples based on the training state of the main model in the whole learning cycle.

Adversarial Samples for Improving Accuracy and Robustness of CL. To improve the quality of learned representations, adversarial attacks can be used to create additional training samples by adding pixel-level perturbations to clean samples (Ho & Vasconcelos, 2020). While adversarial samples are more challenging and can generate a higher loss than the original samples, the perturbed samples still have the same identities as the original ones, which provide limited additional information for learning. Besides, training with adversarial samples is originally designed for the robustness of models against attacks, instead of improving model performance on clean samples (Kurakin et al., 2016; Madry et al., 2017; Goodfellow et al., 2014). As a result, only marginal improvement (Ho & Vasconcelos, 2020) or even degraded performance (Kim et al., 2020; Jiang et al., 2020) of the learned CL model is observed. Different from these works, we generate hard images as a whole, instead of adding pixel-level noises to the existing images, which are more informative for improving the learned representations of CL.

GAN for Data Augmentation. (Zhang et al., 2019; Bowles et al., 2018; Antoniou et al., 2017; Perez & Wang, 2017) employ supervised class-conditional GAN to augment the training data to improve classification performance. However, these works require fully labeled datasets for training GAN. When labels are not available, the quality of images generated by GAN will greatly degrade (Miyato et al., 2018; Zhao et al., 2020) and hence the trained CL model. Besides, either GAN and the target model are isolated and the generated data are not adapted to the training state of the main model (Zhang et al., 2019; Bowles et al., 2018; Antoniou et al., 2017), or both GAN and the classification model aim to minimize the classification loss (Perez & Wang, 2017). Different from these works, our approaches do not rely on labels. Besides, the generator and main model are jointly optimized, in the way that the generator aims to maximize the CL loss while the CL model aims to minimize the CL loss. Also, we generate hard positive pairs for unsupervised representation learning by CL, which is unexplored in these works on conventional supervised learning.

3 Differentiable Hard Sample Generation

Framework overview. We propose a framework *DiffHard* to generate individually hard samples and pairs of hard positives for contrastive learning, such that better representations can be learned with limited data. The framework is shown in Figure 1. As shown in the figure, there are two major components: the hard sample generator (pink) and the main model (blue).

Algorithm 1 shows the process of *DiffHard*. We formulate *DiffHard* as a Min-Max game, where the generator maximizes the contrastive loss by generating hard samples, while the main model minimizes the contrastive loss by learning representations from both the generated and real samples. The sample generator consists of two components, the generator G and its slowly-evolving version G_{ema} (by exponential moving average). G and G_{ema} are first pre-trained on the available unlabeled dataset to generate data following the real data distribution. Meanwhile, G is jointly pre-trained with a discriminator D based on the GAN objective (Brock et al., 2018). Then the contrastive learning of the main model starts, which consists of 3 steps. First, G generates individually hard samples, and G and

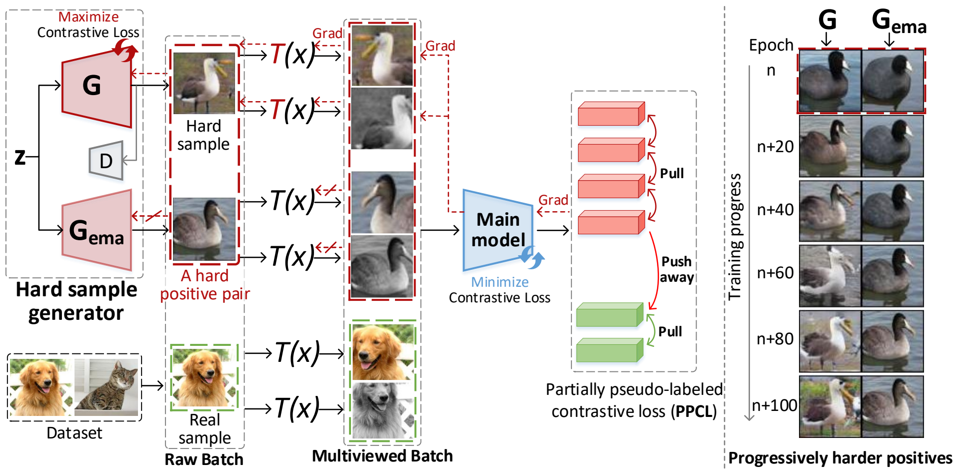


Figure 1: (Left) Generation of hard samples and pairs of hard positives, and the training of the main model and generator. We generate two similar but distinct raw samples, and use two views of each raw sample (four views in total) as positives. No labels are used in the generation of these two similar raw samples. (Right) The generated positive pair becomes progressively harder by tracking the training state of the main model.

Algorithm 1 Differential Hard Sample Generation (*DiffHard*)

```
Input: Main model $f(\cdot,\theta)$, generators $\{G(\cdot,w), G_{\text{ema}}(\cdot,w_{\text{ema}})\}$, discriminator $D(\cdot,\phi)$; dataset $\mathcal{S}$
Output: Trained main model $f(\cdot,\theta)$
Pre-train $G$ and $D$ by Eq.(5) and $G_{\text{ema}}$ by Eq.(7) for $m$ iterations with limited training data $\mathcal{S}$
for iteration $t = 1$ to T do
    # Step 1: Forming batch
    Sample generated batch $B_{\text{gen}}$ with $N$ hard samples of $\frac{N}{2}$ positive pairs following Eq.(6)
    Sample real batch $B_{\text{real}}$ with $N$ samples from $\mathcal{S}$
    Form multiviewed batch $B_{\text{mv}}$ from $\{B_{\text{gen}} \cup B_{\text{real}}\}$ by applying transformations
    # Step 2: Training main model
    Compute loss $\mathcal{L}_{\text{PPCL}}$ (Eq.(8)) with $B_{\text{mv}}$ and optimize $f$ by Eq.(4) to minimize $\mathcal{L}_{\text{PPCL}}$
    # Step 3: Updating generator
    In every $n$ iterations, update $G$ by Eq.(4) to maximize $\mathcal{L}_{\text{PPCL}}$, update $G_{\text{ema}}$ by Eq.(7); optimize
    both $G$ and $D$ by Eq.(5)
end for
```

G_{ema} collaboratively generate pairs of hard positives. The hard samples, hard positives, together with real samples from the dataset form a batch and are fed into the main model to compute the contrastive loss. After that, main model f is updated to minimize the contrastive loss. Finally, G is updated to maximize the contrastive loss to generate harder samples for the main model based on the current training state of the main model. Momentum update is applied to G_{ema} to follow G. Meanwhile, we are using D to force G to generate meaningful data following real data distributions. In this way, the generator and the main model are jointly optimized, such that we can generate progressively harder samples and adapt to the training progress of the main model as shown in Figure 1. The details of each step will be discussed in the following subsections.

3.1 Differentiable Hard Sample Generation

In this subsection, we first introduce the details of the hard sample generator. G generates synthetic data to augment the training data for a higher data diversity and the objective is:

$$\mathcal{L}_{\text{gen+real}} = \sum_{i \in \{I_{\text{gen}} \cup I_{\text{real}}\}} \mathcal{L}_i \tag{2}$$

where $\mathcal{L}_i$ is the contrastive loss of multiviewed sample i defined in Eq.(1). I_{gen} is the set of indices of generated and then transformed (multiviewed) samples $\{\tilde{x}^i_{\text{gen}}\}_{i\in I_{\text{gen}}}$, and I_{real} is the set of indices of multiviewed real samples. The generated raw samples $\{x^k_{\text{gen}}\} = \{G(z_k)\}$ are from the generator G by taking a set of vectors $\{z_k\} \sim p(z)$, drawn from a Gaussian distribution $p(z) = \mathcal{N}(0, I)$, as input. Two transformations are then applied to each x^k_{gen} to get two views $\tilde{x}^{2k-1}_{\text{gen}}$ and $\tilde{x}^{2k}_{\text{gen}}$ to form mutltviewed samples $\{\tilde{x}^i_{\text{gen}}\}_{i\in I_{\text{gen}}}$.

Challenge: Low quality of generated samples. Simply pre-training G and D on the dataset and providing additional synthetic data to the main model cannot effectively improve the learned representation since the synthetic data has intrinsically lower quality than the real data (Brock et al., 2018; Heusel et al., 2017; Salimans et al., 2016) even if labels were available to train a class-conditional generator. When the dataset is unlabeled and the generator is trained in a non class-conditional way, the quality of synthetic data becomes worse (Zhao et al., 2020; Miyato et al., 2018), which degrades the performance of the main model or only provides marginal benefits.

To solve this problem, instead of using the isolated generator and main model, we jointly optimize them by formulating a Min-Max game such that they compete with each other. The generator dynamically adapts to the training progress of the main model and generates hard samples for the main model (i.e. high-quality samples from the perspective of training the main model). The dynamically customized hard samples in each training state of the main model will explore the main model's weakness and encourage it to learn better representations to compete with the generator. Formally, the game is defined as follows.

$$\min_{\theta} \max_{w} \mathcal{L}_{\text{gen+real}} \tag{3}$$

where θ and w are the parameters of the main model and the generator, respectively. To solve the Min-Max game, a pair of gradient descent and ascent are applied to the main model and the generator to update their parameters, respectively. The details of the update are shown as follows.

$$\theta \leftarrow \theta - \eta_{\theta} \frac{\partial \mathcal{L}_{\text{gen+real}}}{\partial \theta}, \quad w \leftarrow w + \eta_{w} \frac{\partial \mathcal{L}_{\text{gen+real}}}{\partial w}, \tag{4}$$

where η_θ and η_w are learning rates for the main model and generator, respectively.

Updating w requires gradients to be propagated from the main model through the data transformations of generated samples to G. Therefore, the transformations T must be differentiable as depicted in Figure 1, which is unexplored in existing CL frameworks (Chen et al., 2020; He et al., 2020) since they only need gradient propagation within the main model to update it. Different from this, we adopt differentiable T (E. Riba & Bradski, 2020) to propagate the gradients through the input of the main model to the generator for harder samples.

In this way, G can be trained to generate hard samples for the main model.

Avoiding trivial solutions. Optimizing Eq.(3) is much easier for G than for the main model, which can result in trivial solutions of G, and thus degrade the performance of the learned main model. To avoid trivial solutions of G, we will update discriminator D to differentiate between the generated data distribution by G and real data distribution from the dataset. In this way, trivial solutions (e.g. random noises) can be detected by D, which forces G to generate meaningful and hard samples to maximize Eq.(3) while following real data distributions. To learn D and G, we employ the GAN objectives (Brock et al., 2018), which are shown as follows.

$$\mathcal{L}_D = \mathop{\mathbb{E}}_{x_{\text{real}} \sim p_{\text{data}}(x)} [f_D(-D(x))] + \mathop{\mathbb{E}}_{z \sim p(z)} [f_D(D(G(z)))], \quad \mathcal{L}_G = \mathop{\mathbb{E}}_{z \sim p(z)} [f_G(D(G(z)))] \tag{5}$$

where z is the latent vector drawn from a Gaussian distribution, and p_{data} is the real data distribution. $f_D(x) = \max(0, 1+x)$ and $f_G(x)$ are the hinge losses for D and G, respectively (Miyato et al., 2018). $\mathcal{L}_D$ and $\mathcal{L}_G$ are alternatively minimized by updating D and G, respectively.

3.2 Hard Positive Generation without Using Labels

In addition to generating hard samples, we also propose a new method to generate hard positive pairs. The main idea is that we can use two similar yet different generators G and G_{ema} to generate two similar but distinct samples as a positive pair, when taking the same latent vector as input.

Positive pair generation. To generate each pair of positives, we use a generator G and its slowly-evolving version G_{ema}, which are very similar but different. By feeding a latent vector z_i sampled from a Gaussian distribution to both G and G_{ema}, a pair of raw samples (x_{2i-1}, x_{2i}), which are similar but distinctive, is generated as a positive pair. The process is described as follows:

$$x_{2i-1} = G(z_i), \quad x_{2i} = G_{\text{ema}}(z_i), \quad z_i \sim p(z) \tag{6}$$

To make the positives harder by increasing their difference, G is updated by gradients from the main model by Eq.(4) while G_{ema} is not. On the other hand, to keep the similarity of generated positive pairs, we update G_{ema} by momentum update following G. Denoting the parameters of G as w and the counterpart of G_{ema} as w_{ema}, w_{ema} is updated by:

$$w_{\text{ema}} \leftarrow m w_{\text{ema}} + (1-m) w \tag{7}$$

where $m \in (0, 1)$ is a momentum parameter.

To generate N samples with $\frac{N}{2}$ positive pairs, we sample $\frac{N}{2}$ latent vectors $\{z_i\}_{i=1...\frac{N}{2}}$ to generate a batch of N raw samples $B_{\text{gen}} = \{x_k\}_{k=1...N}$ following Eq.(6). To leverage the diversity and hardness of generated samples, and the high quality of real samples, we further sample N real samples $B_{\text{real}} = \{x_k\}_{k=N+1...2N}$ from the dataset. Then a raw batch is formed as $B = B_{\text{gen}} \cup B_{\text{real}} = \{x_k\}_{k=1...2N}$, with N generated samples and N real samples. After that, two augmentations are applied to each x_k to form a multiviewed batch $B_{\text{mv}} = \{\tilde{x}_l\}_{l=1...4N}$ for training as shown in Step 1 of Algorithm 1.

Partially pseudo-labeled contrastive loss (PPCL). The contrastive loss in Eq.(2) treats each generated sample separately and only uses two views generated by T as positive pairs. It does not leverage the fact that samples generated by G and G_{ema} are actually positive pairs to learn better representations. To better cluster the generated positive pairs, we define a partially pseudo-labeled contrastive loss by using the input latent vectors z_i as pseudo-labels. Each z_i $(i = 1...\frac{N}{2})$ generates two positives (x_{2i-1}, x_{2i}) in the raw batch and four positives in the multiviewed batch ($\tilde{x}_{4i-3}$, $\tilde{x}_{4i-2}$, $\tilde{x}_{4i-1}$, $\tilde{x}_{4i}$), which are assigned a same pseudo-label to cluster their features.

Within the multiviewed batch B_{mv}, let $i \in I_{\text{gen}} = \{1...2N\}$ be the indices of generated samples and $i \in I_{\text{real}} = \{2N + 1...4N\}$ be the indices of real samples. The PPCL loss is defined as follows.

$$\mathcal{L}_{\text{PPCL}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(v_i \cdot v_p / \tau)}{\sum_{a \in A(i)} \exp(v_i \cdot v_a / \tau)}, \; P(i) = \begin{cases} \{p \in A(i), z_p = z_i\}, & \text{if } i \in I_{\text{gen}} \\ \{j(i)\}, & \text{if } i \in I_{\text{real}} \end{cases} \tag{8}$$

where $I = I_{\text{gen}} \cup I_{\text{real}}$. $A(i) = I \backslash \{i\}$ is the set of indices of i's positives and negatives, and $P(i)$ is the set of indices of i's positives in the multiviewed batch. For real samples $i \in I_{\text{real}}$, the positive $P(i) = j(i)$ is the index of the other view of i in the multiviewed batch. For generated samples $i \in I_{\text{gen}}$, the positives $P(i)$ are defined by the pseudo-labels from the input latent vector z_i, which includes the indices of multiviewed samples originating from the same z_i.

3.3 Analysis of the Effectiveness of Hard Samples Generation

By plugging the PPCL loss in Eq.(8) into the Min-Max problem in Eq.(3), the main model will minimize the loss to learn effective representations from hard samples. Meanwhile, the generator will maximize the loss, which achieves three goals simultaneously: generating individually hard samples, generating hard positive pairs, and generating hard negatives while following real data distribution.

For the multiviewed positives ($\tilde{x}_{4i-3}$, $\tilde{x}_{4i-2}$, $\tilde{x}_{4i-1}$, $\tilde{x}_{4i}$) originating from z_i, taking $\tilde{x}_{4i-3}$ as an example, its positives are ($\tilde{x}_{4i-2}$, $\tilde{x}_{4i-1}$, $\tilde{x}_{4i}$) and its PPCL loss can be rewritten as:

$$\mathcal{L}_{4i-3} = -\frac{1}{3} \sum_{p \in \{4i-2, 4i-1, 4i\}} \log \frac{\exp(f(\tilde{x}_{4i-3}) \cdot f(\tilde{x}_p)/\tau)}{Q} \tag{9}$$

$$Q = \sum_{k \in \{\{1...2N\}, k \neq 4i-3\} \cup \{2N+1...4N\}} \exp(f(\tilde{x}_{4i-3}) \cdot f(\tilde{x}_k)/\tau) \tag{10}$$

When maximizing Eq.(9) by optimizing G, the cosine similarity between positive pairs (in the numerator) will be minimized for each of the three positives p, and the cosine similarity between negative pairs (in the denominator Q) will be maximized.

Generating individually hard samples. For the first positive pair $(\tilde{x}_{4i-3}, \tilde{x}_{4i-2})$, both of them originate from raw sample $x_{2i-1} = G(z_i)$. To minimize $P_1 = f(\tilde{x}_{4i-3}) \cdot f(\tilde{x}_{4i-2}) = f(t_{2i-1}(x_{2i-1})) \cdot f(t'_{2i-1}(x_{2i-1}))$, where $t_{2i-1}, t'_{2i-1} \sim \mathcal{T}$ are sampled transformations, $x_{2i-1} = G(z_i)$ will become individually harder such that the main model $f(\cdot)$ cannot perfectly generate similar representations for this positive pair originating from x_{2i-1}.

Generating hard positive pairs. For the second positive pair $(\tilde{x}_{4i-3}, \tilde{x}_{4i-1})$ and the third positive pair $(\tilde{x}_{4i-3}, \tilde{x}_{4i})$, the first multiviewed sample in each pair $\tilde{x}_{4i-3}$ is from raw sample $x_{2i-1} = G(z_i)$ by G, and the second multiviewed sample in each pair $\tilde{x}_{4i-1}$, $\tilde{x}_{4i}$ are from raw sample $x_{2i} = G_{\text{ema}}(z_i)$ by G_{ema}. To minimize $P_2 = f(\tilde{x}_{4i-3}) \cdot f(\tilde{x}_{4i-1}) = f(t_{2i-1}(x_{2i-1})) \cdot f(t_{2i}(x_{2i}))$ and $P_3 = f(\tilde{x}_{4i-3}) \cdot f(\tilde{x}_{4i}) = f(t_{2i-1}(x_{2i-1})) \cdot f(t'_{2i}(x_{2i}))$, both of which take x_{2i-1} and x_{2i} as input, the difference between x_{2i-1} and x_{2i} will be increased. To achieve this, since no gradients are propagated to G_{ema}, it will be fixed when optimizing Eq.(9) and G will be pushed apart from G_{ema}. In this way, distinct samples will be generated as positive pairs by G and G_{ema}. On the other hand, G_{ema} is slowly updated following G by Eq.(7), which encourages them to generate similar samples when taking the same latent vector z_i as input. As a result, by updating G and G_{ema} by Eq.(8) and Eq.(7), distinct yet similar samples can be generated as a positive pair as shown in Figure 1 (Right).

Generating hard negatives while following real distributions. For the negative pairs, when maximizing Eq.(9), Q will be maximized. For the first set $k \in \{\{1...2N\}, k \neq 4i-3\}$, the term $N_k = f(\tilde{x}_{4i-3}) \cdot f(\tilde{x}_k) = f(t_{2i-1}(x_{2i-1})) \cdot f(t(x_{\lceil \frac{k}{2} \rceil})), t \in \{t_{\lceil \frac{k}{2} \rceil}, t'_{\lceil \frac{k}{2} \rceil}\}$ is the cosine similarity of two samples generated by G or G_{ema}. When maximizing N_k by optimizing G, G will generate samples that the main model cannot effectively discriminate in its current training state. As a result, the main model will learn to distinguish these hard negatives.

For the second set $k \in \{2N+1...4N\}$, N_k is the cosine similarity between features of one generated (and transformed) sample $t_{2i-1}(x_{2i-1})$ and one transformed real sample $t(x_{\lceil \frac{k}{2} \rceil}), t \in \{t_{\lceil \frac{k}{2} \rceil}, t'_{\lceil \frac{k}{2} \rceil}\}$. When maximizing N_k, G will generate samples that have larger similarities to real samples. Therefore, G will generate samples following real data distributions, instead of generating meaningless samples.

4 Experimental Results

Datasets and model architecture. We evaluate the proposed approaches on five datasets, including CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), Fashion-MNIST (Xiao et al., 2017), ImageNet-100 and ImageNet-10. ImageNet-100 is a subset of ImageNet (Russakovsky et al., 2015) also used in (Van Gansbeke et al., 2020; Kalantidis et al., 2020; Shen et al., 2020; Tian et al., 2019), and ImageNet-10 is a smaller subset of ImageNet. We use ResNet-18 as the main model and use a 2-layer MLP to project the output to 128-dimensional representation space (Chen et al., 2020; He et al., 2020). We use the generator and discriminator architectures from (Brock et al., 2018). The batch size is 256 and the main model is trained for 300 epochs on CIFAR-10, CIFAR-100 and ImageNet-10, 200 epochs on Fashion-MNIST, and 100 epochs on ImageNet-100 for efficient evaluation. The training details and detailed model architectures can be found in the Appendix.

Metrics. To evaluate the learned representations, we use two metrics *linear evaluation* and *transfer learning* for self-supervised learning (Chen et al., 2020). In linear evaluation, a linear classifier is trained on top of the frozen base encoder, and the test accuracy represents the quality of learned representations. We first perform CL by the proposed approaches without labels to learn representations. Then we *fix* the encoder and train a linear classifier on 100% labeled data on top of the encoder. The classifier is trained for 500 epochs with Adam optimizer and learning rate 3e-4. Transfer learning evaluates the generalization of learned features. The features are first learned on the source dataset, then evaluated on the target task. Following (Caron et al., 2020), we train a linear classifier on top of the frozen encoder on the target task, using the same hyper-parameters as the ones in linear evaluation.

Baselines. We compare the proposed approaches with multiple approaches. *SimCLR* is the original contrastive learning approach by using real data (Chen et al., 2020). *SimCLR-2x* is a variant of *SimCLR* by sampling 2× data for each batch from the dataset, serving as a strong baseline. By comparing our approaches with *SimCLR-2x*, we evaluate if the samples generated by our approaches benefit CL more than using the same amount of additional real data. *CLAE* is the SOTA approach to improve the performance of CL by using pixel-level adversarial perturbations (Ho & Vasconcelos, 2020). *CL-GAN* uses a generator from GAN (Brock et al., 2018) to generate additional training data for *SimCLR* (Chen et al., 2020) without joint learning.

4.1 Linear Evaluation

Table 1: **Linear classification when limited training data are available for CL.** Available data are labeled and used for learning the classifier on the *fixed* encoder and top-1 accuracy is reported.

Method	CIFAR-10 (20% tr. data)	CIFAR-10 (10% tr. data)	CIFAR-100 (20% tr. data)	CIFAR-100 (10% tr. data)	FMNIST (20% tr. data)	FMNIST (10% tr. data)	ImageNet-100 (20% tr. data)	ImageNet-10 (20% tr. data)
SimCLR	80.19	76.05	44.13	37.46	87.92	87.26	40.39	62.60
SimCLR-2x	81.04	76.06	44.75	37.10	89.60	87.49	42.77	65.00
CLAE	79.84	74.38	45.17	37.28	88.48	87.58	40.14	59.60
CL-GAN	80.81	78.36	46.94	40.03	88.85	87.68	42.79	64.20
Proposed	**85.11**	**80.94**	**50.24**	**43.74**	**91.77**	**89.79**	**46.73**	**68.40**

Linear classification of representations learned with limited training data. We evaluate the proposed approaches by linear evaluation when limited training data is available for contrastive representation learning. Linear classification evaluates the linear separability of learned representations

(Chen et al., 2020), and higher accuracy indicates more discriminative and desirable features. As shown in Table 1, the proposed approaches consistently outperform the SOTA approaches by a large margin. About 5% and 6% improvements are observed on CIFAR-10/100 and ImageNet-100/10, respectively. Notably, the proposed approaches outperform or perform on par with the best-performing baselines trained with 2× data, achieving 2× data efficiency. For example, with 10% training data on CIFAR-10, the best-performing baseline with 20% training data achieves 81.04% accuracy, while the proposed approaches achieve a similar accuracy of 80.94% by using only 10% training data. Besides, the proposed approaches even largely outperform SimCLR-2x, which samples 2× real data in each training batch. This result shows that the customized hard data by the proposed approaches have higher quality in terms of training CL than real data.

Table 2: **Linear classification when the full dataset is available for CL.** 100% labeled data are used for learning the classifier on the *fixed* encoder. And top-1 accuracy is reported.

Method	CIFAR-10	CIFAR-100	FMNIST	ImageNet-100	ImageNet-10
SimCLR (Chen et al., 2020)	90.37	63.93	92.35	56.19	83.20
SimCLR-2x (Chen et al., 2020)	91.37	64.88	92.50	58.73	82.20
CLAE (Ho & Vasconcelos, 2020)	90.13	63.25	92.36	55.20	81.20
CL-GAN (Chen et al., 2020; Brock et al., 2018)	89.90	63.82	92.64	57.58	83.40
Proposed	**92.94**	**67.41**	**93.94**	**60.46**	**87.40**

Linear classification of representations learned with the full dataset. When more training data is available for CL, the proposed approaches still significantly outperform the SOTA approaches. As shown in Table 2, substantial improvements of 2.57%, 3.48%, 1.59%, 4.27%, and 4.00% over the original CL framework SimCLR are observed on five datasets, respectively. This result shows that the proposed methods can effectively leverage given data for CL.

4.2 Transfer Learning

Table 3: **Transfer learning to downstream tasks with limited training data for learning the encoder on the source datasets.** Top-1 accuracy of linear classification on top of a *fixed* encoder is reported.

Source	CIFAR-10 (20% tr. data)	CIFAR-10 (10% tr. data)	CIFAR-100 (20% tr. data)	CIFAR-100 (10% tr. data)	ImageNet-100 (20% tr. data)		ImageNet-10 (20% tr. data)	
Target	CIFAR-100		CIFAR-10		CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
SimCLR	46.88	44.98	74.33	71.79	77.09	52.94	69.86	42.56
SimCLR-2x	48.45	46.74	73.94	72.49	77.49	53.03	71.20	43.82
CLAE	46.93	46.11	73.47	71.48	77.65	52.73	69.98	44.19
CL-GAN	48.54	47.39	75.15	73.43	78.54	54.47	70.44	44.74
Proposed	**52.51**	**50.08**	**77.26**	**75.10**	**81.19**	**58.42**	**73.08**	**46.46**

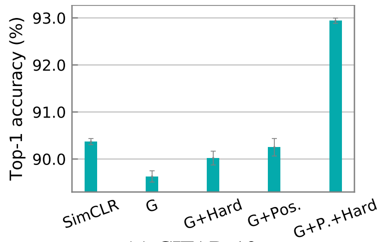
Transfer learning with limited training data on the source dataset. We evaluate the generalization performance of learned representations by transferring to downstream vision tasks. The encoder is trained on the source dataset and transferred to target tasks, and we report the linear classification performance. As shown in Table 3, when limited training data is available for learning the encoder on the source datasets, the proposed approaches consistently outperform the SOTA baselines. Notably, with only 20% training data of the source dataset CIFAR-10 and transferring to CIFAR-100, the proposed approaches outperform the best-performing baseline with 100% training data (Table 4) of CIFAR-10 (52.51% vs. 52.47%), which achieves 5× data efficiency. Besides, with only 10% training data on CIFAR-10, the proposed approaches outperform the best-performing baseline with 20% training data by a large margin (50.08% vs. 48.54%).

Table 4: **Transfer learning with the full dataset for learning the encoder on the source datasets.**

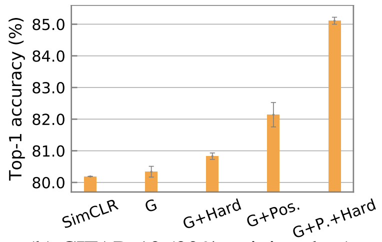
Source	CIFAR-10	CIFAR-100	ImageNet-100		ImageNet-10	
Target	CIFAR-100	CIFAR-10	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
SimCLR	51.63	78.87	83.15	61.63	73.91	44.88
SimCLR-2x	52.22	78.53	83.78	62.92	73.77	46.53
CLAE	52.47	78.10	83.02	61.69	74.07	47.11
CL-GAN	52.16	78.98	83.11	62.03	73.62	44.55
Proposed	**56.38**	**82.24**	**85.75**	**64.94**	**75.38**	**49.03**

Transfer learning with the full dataset. We further evaluate the transfer learning performance when more training data is available for learning the encoder on the source datasets. In Table 4, our approaches consistently outperform the baselines on four source datasets with various target tasks.

4.3 Ablations



(a) CIFAR-10.



(b) CIFAR-10 (20% training data).

Figure 2: **Effectiveness of hard samples, positive pairs, and hard positive pairs.** Enabling the components in the proposed approaches one by one accumulatively improves the performance. *G* is using a separate generator, *G+Hard* is jointly optimizing G and the main model to generate hard samples, *G+Pos.* is G with positive pair generation, and *G+P.+Hard* is the proposed approach with all the components enabled. Top-1 accuracy of linear classification is reported.

Effectiveness of hard samples, positive pairs, and hard positive pairs. We perform ablation studies to evaluate the effectiveness of hard samples (*G+Hard*), positive pairs (*G+Pos.*), and hard positive pairs (*G+Pos.+Hard*). The results of linear evaluation are shown in Figure 2. On CIFAR-10, using a simple generator degrades the performance of CL compared with the original SimCLR approach due to the low-quality samples from the generator. Using the proposed hard samples and positive pairs recover the accuracy by 0.39% and 0.62%, respectively, while using hard positive pairs outperforms SimCLR by 2.57% (92.94% vs. 90.37%). Similar results are observed on CIFAR-10 with 20% training data, where using a separate generator can slightly improve the accuracy by 0.15%, while enabling all the proposed components significantly improves the accuracy by 4.92%.

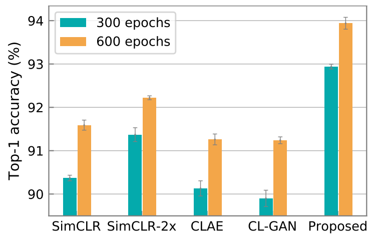


Figure 3: **Linear classification accuracy with more learning epochs on CIFAR-10**. The proposed approaches consistently outperform the baselines with longer training.

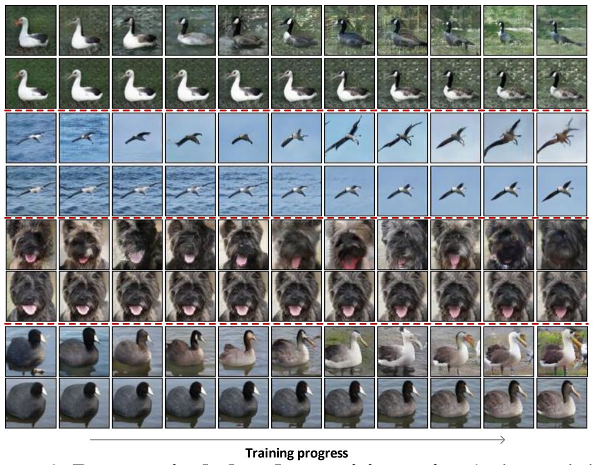


Figure 4: **Progressively harder positive pairs** during training.

Impact of longer training. We evaluate the impact of training epochs. We increase the number of training epochs from the default 300 epochs to 600 epochs on CIFAR-10, and evaluate the learned representations by linear classification. As shown in Figure 3, with longer training, the advantage of the proposed approaches over the baselines remains, and the margin over the best-performing baseline becomes larger. This is because with longer training, more and harder samples are customized for the main model, which improves its performance.

Evolution of positive pairs. The dynamically harder positive pairs in the training progress are shown in Figure 4. Every two adjacent rows show the evolution of positive pairs on ImageNet-10. With growing knowledge of the main model, positive pairs become progressively harder, while being similar objects. Learning from harder positive pairs improves the quality of learned representations.

5 Conclusion

We propose a framework to achieve data-efficient contrastive representation learning. We propose a data generator to track the training state of the main model and customize hard samples on-the-fly in the training process. To further generate hard positive pairs without using labels, we propose a pair of generators to generate similar but distinct samples. Experimental results show superior accuracy of the proposed approaches compared with the state-of-the-art.

References

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Tianlong Chen, Yu Cheng, Zhe Gan, Jingjing Liu, and Zhangyang Wang. Ultra-data-efficient gan training: Drawing a lottery ticket first, then training it toughly. *arXiv preprint arXiv:2103.00397*, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

D. Ponsa E. Rublee E. Riba, D. Mishkin and G. Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020. URL `https://arxiv.org/pdf/1910.02190.pdf`.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

Chih-Hui Ho and Nuno Vasconcelos. Contrastive learning with adversarial examples. *arXiv preprint arXiv:2010.12050*, 2020.

Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. *arXiv preprint arXiv:2010.13337*, 2020.

Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *arXiv preprint arXiv:2010.01028*, 2020.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. *arXiv preprint arXiv:2006.07589*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

Zhiqiang Shen, Zechun Liu, Zhuang Liu, Marios Savvides, and Trevor Darrell. Rethinking image mixture for unsupervised visual representation learning. *arXiv preprint arXiv:2003.05438*, 2020.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Learning to classify images without labels. *arXiv preprint arXiv:2005.12320*, 2020.

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

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

Xiaofeng Zhang, Zhangyang Wang, Dong Liu, and Qing Ling. Dada: Deep adversarial data augmentation for extremely low data regime classification. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2807–2811. IEEE, 2019.

Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *arXiv preprint arXiv:2006.10738*, 2020.