Can contrastive learning avoid shortcut solutions?

Anonymous Author(s)
Affiliation
Address
email

Abstract

The generalization of representations learned via contrastive learning depends crucially on what features of the data are extracted. However, we observe that the contrastive loss does not always sufficiently guide which features are extracted, a behavior that can negatively impact the performance on downstream tasks via "shortcuts", i.e., by inadvertently suppressing important predictive features. We find that feature extraction is influenced by the *difficulty* of the so-called instance discrimination task (i.e., the task of discriminating pairs of similar points from pairs of dissimilar ones). Although harder pairs improve the representation of some features, the improvement comes at the cost of suppressing previously well represented features. In response, we propose *implicit feature modification* (IFM), a method for altering positive and negative samples in order to guide contrastive models towards capturing a wider variety of predictive features. Empirically, we observe that IFM reduces feature suppression, and as a result improves performance on vision and medical imaging tasks.

1 Introduction

Representations trained with contrastive learning are adept at solving various vision tasks including classification, object detection, instance segmentation, and more [5, 15, 42]. In contrastive learning, encoders are trained to discriminate pairs of positive (similar) inputs from a selection of negative (dissimilar) pairs. This task is called *instance discrimination*: It is often framed using the InfoNCE loss [14, 33], whose minimization forces encoders to extract input features that are sufficient to discriminate similar and dissimilar pairs.

However, learning features that are discriminative during training does not guarantee a model will generalize. Many studies find inductive biases in supervised learning toward *simple* "shortcut" features and decision rules [16, 21, 32] which result in unpredictable model behavior under perturbations [22, 41] and failure outside the training distribution [2, 35]. Simplicity bias has various potential sources [11] including training methods [8, 29, 39] and architecture design [10, 17]. Bias towards shortcut decision rules also hampers transferability in contrastive learning [4], where it is in addition influenced by the instance discrimination task. These difficulties lead us to ask: *can the contrastive instance discrimination task itself be modified to avoid learning shortcut solutions?*

We approach this question by studying the relation between contrastive instance discrimination and feature learning. First, we theoretically explain why optimizing the InfoNCE loss alone does not guarantee avoidance of shortcut solutions that *suppress* (i.e., discard) certain input features [4, 11]. Second, despite this negative result, we show that it is still possible to trade off representation of one feature for another using simple methods for adjusting the difficulty of instance discrimination. However, these methods have an important drawback: improved learning of one feature often comes at the cost of harming another. That is, feature suppression is still prevalent. In response, we propose *implicit feature modification*, a technique that encourages encoders to discriminate instances using multiple input features. Our method introduces no computational overhead, reduces feature suppression (without trade-offs), and improves generalization on various downstream tasks.

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

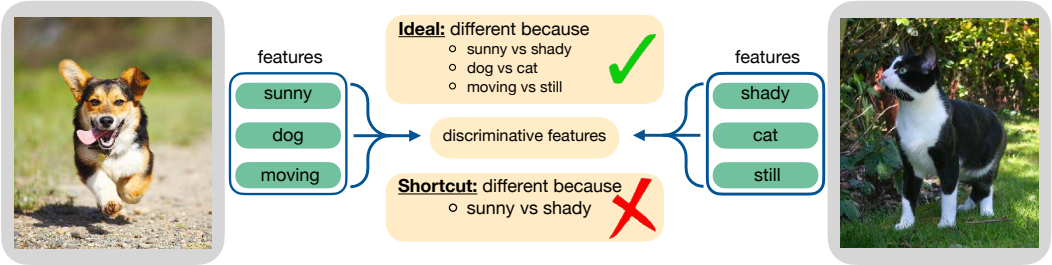

Figure 1: An ideal encoder would be capable of discriminating between instances using multiple distinguishing features instead of finding simple shortcuts that suppress features. We show that InfoNCE-trained encoders can suppress features (Sec. 2.2). However, making instance discrimination harder during training can trade off representation of different features (Sec. 2.3). To avoid the need for trade-offs we propose *implicit feature modification* (Sec. 3), which reduces suppression in general, and improves generalization (Sec. 4).

Contributions. In summary, this paper makes the following main contributions:

1. It analyzes feature suppression in contrastive learning, and explains why feature suppression can occur when optimizing the InfoNCE loss.
2. It studies the relation between instance discrimination tasks and feature learning; concretely, it finds that adjustments to the difficulty of instance discrimination leads to different features being learned.
3. It proposes *implicit feature modification*, a simple and efficient method that reduces the tendency to use feature suppressing shortcut solutions and improves generalization.

1.1 Related work

Unsupervised representation learning is enjoying a renaissance driven by steady advances in effective frameworks [3, 5, 15, 18, 33, 42, 49]. Contrastive learning methods, such as SimCLR and MoCo [5, 15, 43], have been especially successful. Beyond explicitly contrastive approaches, other Siamese methods such as SimSiam and BYOL have also been proposed [6, 13].

Pretext task design has been at the core of progress in self-supervised learning. Previously popular tasks include image colorization [52] and inpainting [34], and theoretical work shows pre-trained encoders can provably generalize if a pretext task necessitates the learning of features that solve downstream tasks [27, 37]. In contrastive learning, augmentation strategies are a key design component [5, 46, 48], as are negative mining techniques [9, 15, 25, 38]. While feature learning in contrastive learning has received less attention, recent work finds that low- and mid-level features are more important for transfer learning [53], and feature suppression can occur [4] just as with supervised learning [10, 16]. Combining contrastive learning with an auto-encoder has also been considered [28], but was found to harm representation of some features in order to avoid suppression of others. Our work is distinguished from prior work through our focus on how the design of the instance discrimination task itself affects which features are learned.

2 Feature suppression in contrastive learning

Feature suppression refers to the phenomenon where, in the presence of multiple predictive input features, a model uses only a subset of them and ignores the others. The selected subset often corresponds to intuitively "simpler" features, e.g., color as opposed to shape. Such features lead to "shortcut" decision rules that might perform well on training data, but can harm generalization and robustness to data shifts. Feature suppression has been identified as a common problem in deep learning [11], and both supervised and contrastive learning suffer from biases induced by the choice of optimizer and architecture. However, contrastive learning bears an additional potential source of bias: *the choice of instance discrimination task*. Which positive and negative pairs are presented critically affects which features are discriminative, and hence which features are learned. In this work we study the relation between feature suppression and instance discrimination.

First, we explain why optimizing the InfoNCE loss is insufficient in general to avoid feature suppression, and show how it can lead to counter-intuitive generalization (Sec. 2.2). Given this negative result, we then ask if it is at least possible to *control* which features a contrastive encoder learns? We find that this is indeed the case, and that adjustments to the instance discrimination task lead to different features being learned (Sec. 2.3). However, the primary drawback of these adjustments is that improving one feature often comes at the cost of harming representation of another. That is, feature suppression is still prevalent. Addressing this drawback is the focus of Sec. 3.

2.1 Setup and definition of feature suppression

Formally, we assume that the data has underlying feature spaces $\mathcal{Z}^1, \dots, \mathcal{Z}^n$ with a distribution p_j on each $\mathcal{Z}^j$. Each $j \in [n]$, corresponding to a latent space $\mathcal{Z}^j$, models a distinct feature. We write the product as $\mathcal{Z}^S = \prod_{j \in S} \mathcal{Z}^j$, and simply write $\mathcal{Z}$ instead of $\mathcal{Z}^{[n]}$ where $[n] = \{1, \dots, n\}$. A set of features $z = (z^j)_{j \in [n]} \in \mathcal{Z}$ is generated by sampling each coordinate $z^j \in \mathcal{Z}^j$ independently, and we denote the measure on $\mathcal{Z}$ induced by z by λ. Further, let $\lambda(\cdot|z^S)$ denote the conditional measure on $\mathcal{Z}$ for fixed z^S. For $S \subseteq [n]$ we use z^S to denote the projection of z onto $\mathcal{Z}^S$. Finally, an injective map $g : \mathcal{Z} \to \mathcal{X}$ produces observations $x = g(z)$.

Our aim is to train an encoder $f : \mathcal{X} \to \mathbb{S}^{d-1}$ to map data inputs x to the surface of the unit sphere $\mathbb{S}^{d-1} = \{u \in \mathbb{R}^d : \|u\|_2 = 1\}$ in such a way that f extracts useful information. To formally define feature suppression, we need the *pushforward* $h\#\nu(V) = \nu(h^{-1}(V))$ of a measure ν on a space $\mathcal{U}$ for a measurable map $h : \mathcal{U} \to \mathcal{V}$ and measurable $V \subseteq \mathcal{V}$, where $h^{-1}(V)$ denotes the preimage.

Definition 1. Consider an encoder $f : \mathcal{X} \to \mathbb{S}^{d-1}$ and features $S \subseteq [n]$. For each $z^S \in \mathcal{Z}^S$, let $\mu(\cdot|z^S) = (f \circ g)\#\lambda(\cdot|z^S)$ be the pushforward measure on $\mathbb{S}^{d-1}$ by $f \circ g$ of the conditional $\lambda(\cdot|z^S)$.

1. f *suppresses* S if for any pair $z^S, \bar{z}^S \in \mathcal{Z}^S$, we have $\mu(\cdot|z^S) = \mu(\cdot|\bar{z}^S)$.
2. f *distinguishes* S if for any pair of distinct $z^S, \bar{z}^S \in \mathcal{Z}^S$, measures $\mu(\cdot|z^S), \mu(\cdot|\bar{z}^S)$ have disjoint support.

Feature suppression is thus captured in a distributional manner, stating that S is suppressed if the encoder distributes inputs in a way that is invariant to the value z^S. Distinguishing features, meanwhile, asks that the encoder f separates points with different features z^S into disjoint regions.

In contrastive learning, the encoder $f : \mathcal{X} \to \mathbb{S}^{d-1}$ is trained to optimize the InfoNCE loss

$$\mathcal{L}_m(f) = \mathbb{E}_{x, x^+, \{x_i^-\}_{i=1}^m} \left[-\log \frac{e^{f(x)^\top f(x^+)/\tau}}{e^{f(x)^\top f(x^+)/\tau} + \sum_{i=1}^m e^{f(x)^\top f(x_i^-)/\tau}} \right], \tag{1}$$

where τ is known as the *temperature*. Positive pairs x, x^+ are generated by first sampling $z \sim \lambda$, then independently sampling two random augmentations $a, a^+ \sim \mathcal{A}$, $a : \mathcal{X} \to \mathcal{X}$ from a distribution $\mathcal{A}$, and setting $x = a(g(z))$ and $x^+ = a^+(g(z))$. We assume $\mathcal{A}$ samples the identity function $a(x) = x$ with non-zero probability ("x is similar to itself"). Each negative example x_i^- is generated as $x_i^- = a_i(g(z_i))$, by independently sampling features $z_i \sim \lambda$ and an augmentation $a_i \sim \mathcal{A}$.

2.2 Why optimizing the InfoNCE loss can still lead to feature suppression

Do optimal solutions to the InfoNCE loss automatically avoid shortcut solutions? Unfortunately, as we show in this section, this is not the case in general; there exist both optimal solutions of the InfoNCE loss that do and solutions that do not suppress a given feature. Following previous work [38, 47, 54], we analyze the loss as the number of negatives goes to infinity,

$$\mathcal{L} = \lim_{m \to \infty} \left\{ \mathcal{L}_m(f) - \log m - \tfrac{2}{\tau} \right\} = \tfrac{1}{2\tau} \mathbb{E}_{x, x^+} \|f(x) - f(x^+)\|^2 + \mathbb{E}_{x^+} \log \left[\mathbb{E}_{x^-} e^{f(x^+)^\top f(x^-)/\tau} \right].$$

We subtract $\log m$ to ensure the limit is finite, and use x^- to denote a random sample with the same distribution as x_i^-. Following [47] we denote the first term by $\mathcal{L}_{\text{align}}$ and the second term by the "uniformity loss" $\mathcal{L}_{\text{unif}}$, so $\mathcal{L} = \mathcal{L}_{\text{align}} + \mathcal{L}_{\text{unif}}$. Indeed, any f optimizing $\mathcal{L}_{\text{unif}}$ achieves perfect uniformity, i.e., a uniform distribution σ_d of the data representations on the sphere $\mathbb{S}^{d-1}$: $(f \circ g)\#\lambda = \sigma_d$ [47]. Prop. 1 (proved in App. A) shows that it is possible to achieve perfect uniformity both with encoders that suppress features and encoders that do not.

Proposition 1. *As before, let $\mu(\cdot|z^S) = (f \circ g)\#\lambda(\cdot|z^S)$ denote the pushforward measure on $\mathbb{S}^{d-1}$. For any set of features $S \subsetneq [n]$ the following statements hold:*

1. *Suppose f is such that $\mathcal{L}_{align}(f) = 0$ and for each fixed $z^S \in \mathcal{Z}^S$, we have $\mu(\cdot|z^S) = \sigma_d$. Then f optimizes $\mathcal{L}$ and suppresses features S.*
2. *Suppose f is such that $\mathcal{L}_{align}(f) = 0$ and there is a partition $\{A_{z^S}\}_{z^S \in \mathcal{Z}^S}$ of $\mathbb{S}^{d-1}$ into equal sized regions (i.e. $A_{z^S}, A_{\bar{z}^S}$ have the same volume for all $z^S, \bar{z}^S$) and for fixed $z^S \in \mathcal{Z}^S$ we have $\mu(\cdot|z^S) = \sigma_d(\cdot|A_{z^S})$. Then f optimizes $\mathcal{L}$ and distinguishes features S.*

Thus, the InfoNCE loss is optimized both by encoders that suppress and by encoders that distinguish features S.

Similarly to the existence of perfectly uniform encoders [47], encoders satisfying the first and second conditions are guaranteed to exist when $\mathcal{Z}^i = \mathbb{S}^{d-1}$, and the marginal distribution of z^i on $\mathcal{Z}^i$ is bounded below by a strictly positive constant (e.g., the uniform distribution). Prop. 1 shows that empirical observations of feature suppression [4] (see also Fig. 3) are not simply due to a failure to sufficiently optimize the loss, but that the possibility of feature suppression is *built into* the loss.

What does Prop. 1 imply for the generalization behavior of encoders? Besides explaining why feature suppression occurs, Prop. 1 suggests another counter-intuitive possibility: *lower InfoNCE loss may actually lead to worse performance on some tasks*.

To empirically study whether this possibility manifests itself in practice, we use two datasets with known semantic features: (1) In the Trifeature data, [16] each image is 128×128 and has three features: color, shape, and texture, each taking 10 values. See Fig. 10, App. C for sample images. (2) In the STL-digits data, samples combine MNIST digits and STL10 objects by placing copies of a randomly selected MNIST digit on top of an STL10 image. See Fig. 11 App. C for sample images.

We train encoders with ResNet-18 backbone using SimCLR. To study correlations between the loss value and error on downstream tasks, we train 33 encoders on Trifeature and 7 encoders on STL-digits with different hyperparameter settings (see App. C.3 for details on training and hyperparameters). For Trifeature, we compute the Pearson correlation between InfoNCE loss and linear readout error when predicting {color, shape, texture}. Likewise, for STL-digits we compute correlations between the InfoNCE loss and MNIST and STL10 prediction error.

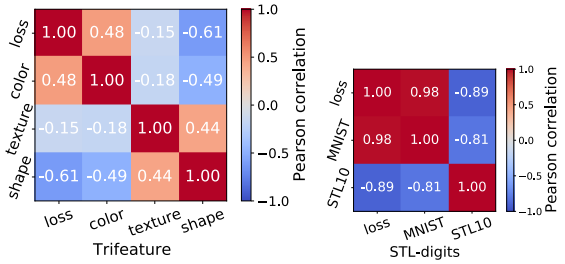

Figure 2: Linear readout error on different downstream tasks can be negatively correlated. Further, lower InfoNCE loss does not always yield not lower error: error rates on texture, shape and STL10 prediction are *negatively correlated* with InfoNCE loss.

Fig. 2 shows that performance on different downstream tasks is not always positively correlated. For Trifeature, color error is negatively correlated with shape and texture, while for STL-digits there is a strong negative correlation between MNIST digit error and STL10 error. Importantly, lower InfoNCE loss is correlated with lower prediction error for color and MNIST-digit, but with *larger* error for shape, texture and STL10. Hence, lower InfoNCE loss can improve representation of some features (color, MNIST digit), but may actually *hurt* others. This conflict is likely due to the simpler color and MNIST digit features being used as shortcuts. Our observation is an important addition to the statement of Wang and Isola [47] that lower InfoNCE loss improves generalization: the situation is more subtle – whether lower InfoNCE helps generalization on a task depends on the use of shortcuts.

2.3 Controlling feature learning via the difficulty of the instance discrimination task

The previous section showed that the InfoNCE objective has solutions that suppress features. Next, we ask what factors determine which features are suppressed? Is there a way to target *specific* features and ensure they are encoded? One idea is to use *harder* positive and negative examples. Hard examples are precisely those that are not easily distinguishable using the currently extracted features. So, a focus on hard examples may change the scope of the captured features. To test this hypothesis, we consider two methods for adjusting the difficulty of positive and negative samples:

1. Temperature τ in the InfoNCE loss (Eqn. 1). Smaller τ places higher importance on positive an negative pairs with high similarity [45].
2. Hard negative sampling method of Robinson et al. [38], which uses importance sampling to sample harder negatives. The method introduces a hardness concentration parameter β, with larger β corresponding to harder negatives (see [38] for full details).

Results reported in Fig. 3 (also Fig. 13 in App. C.3) show that varying instance discrimination difficulty—i.e., varying temperature τ or hardness concentration β—enables trade-offs between which features are represented. On Trifeature, easier instance discrimination (large τ, small β) yields good performance on 'color'—an "easy" feature for which a randomly initialized encoder already has high linear readout accuracy—while generalization on the harder texture and shape features is poor. The situation *reverses* for harder instance discrimination (large τ, small β). We hypothesize

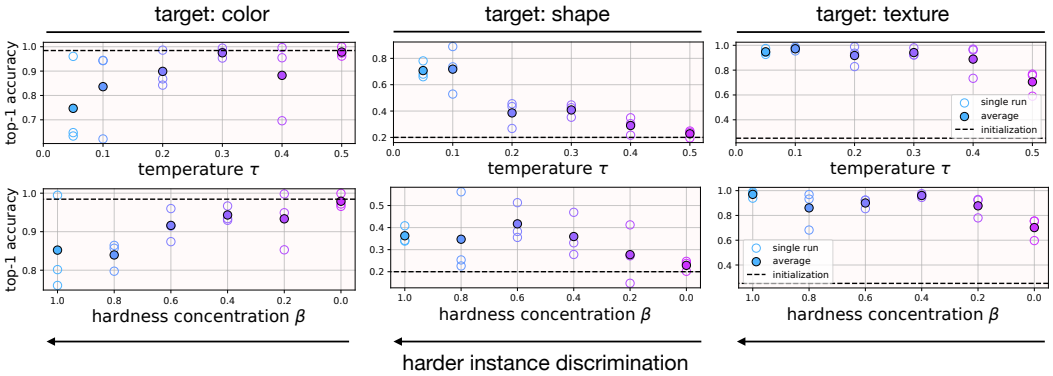

Figure 3: Trifeature dataset [16]. The *difficulty* of instance discrimination affects which features are learned (Sec. 2.3). When instance discrimination is easy (big τ, small β), encoders represent color well and other features badly. When instance discrimination is hard (small τ, big β), encoders represent more challenging shape and texture features well, at the expense of color.

that the use of "easy" features with easy instance discrimination is analogous to simplicity biases in supervised deep networks [17, 21]. As with supervised learning [10, 17], we observe a bias for texture over shape in convolutional networks, with texture prediction always outperforming shape.

That there are simple levers for controlling which features are learned already distinguishes contrastive learning from supervised learning, where attaining such control is less easy (though efforts have been made [23]). However, these results show that representation of one feature must be sacrificed in exchange for learning another one better. To understand how to develop methods for improving feature representation without suppressing others, the next result (proof in App. A) examines more closely *why* there is a relationship between (hard) instance discrimination tasks and feature learning.

Proposition 2 (Informal)**.** *For $S \subseteq [n]$ suppose that $x, x^+, \{x_i^-\}_i$ are conditioned on the event that they have the same features S. Then, under suitable conditions, any f that minimizes the (limiting) InfoNCE loss suppresses features S.*

The positive and negative instances in Prop. 2 must be distinguished with features in S^c. Relating this point to the above observations, assume that an encoder exclusively uses features S. Any positives and negatives that do not (much) differ in features S are difficult for the encoder. By Prop. 2, focusing the training on these difficult examples pushes the encoder to instead use features in S^c, i.e., to learn *new* features. But at the same time, the proposition also says that a strong focus on such hard negative pairs leads to suppressing the originally used features S, explaining the results in Fig. 3. While the two techniques for adjusting instance difficulty we studied were unable to avoid feature suppression, this insight forms the motivation for *implicit feature modification*, which we introduce next.

3 Implicit feature modification for reducing feature suppression

The previous section found that simple adjustments to instance discrimination *difficulty* could significantly alter which features a model learns. Prop. 2 suggests that this ability to modify which features are learned stems from holding features constant across positive and negative samples. However, these methods were unable to avoid trade-offs in feature representation (Fig. 3) since features that are held constant are themselves suppressed (Prop. 2).

To avoid this effect, we develop a technique that *adaptively* modifies samples to remove whichever features are used to discriminate a particular positive pair from negatives, then trains an encoder to discriminate instances using *both* the original features, and the features left over after modification. While a natural method for modifying features is to directly transform raw input data, it is very challenging to modify the semantics of an input in this way. So instead we propose modifying features by applying transformations to encoded samples $v = f(x)$. Since we modify the encoded samples, instead of raw inputs x, we describe our method as *implicit*.

We set up our notation. Given batch $x, x^+, \{x_i^-\}_{i=1}^N$ we write $v = f(x)$, $v^+ = f(x^+)$, and $v_i^- = f(x_i^-)$ to denote the corresponding embeddings. As in Eqn. 1, the point-wise InfoNCE loss is,

$$\ell(v, v^+, \{v_i^-\}_{i=1}^N) = -\log \frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^N e^{v^\top v_i^-/\tau}}.$$

Definition 2 (Implicit feature modification)**.** Given budget $\boldsymbol{\varepsilon} \in \mathbb{R}_+^N$, and encoder $f : \mathcal{X} \to \mathbb{S}^d$, an adversary removes features from f that discriminates batch $x, x^+, \{x_i^-\}_{i=1}^N$ by maximizing the point-wise InfoNCE loss, $\ell_{\boldsymbol{\varepsilon}}(v, v^+, \{v_i^-\}_{i=1}^N) = \max_{\delta^+ \in \mathcal{B}_{\varepsilon^+}, \{\delta_i^- \in \mathcal{B}_{\varepsilon_i}\}_{i=1}^N} \ell(v, v^+ + \delta^+, \{v_i^- + \delta_i^-\}_{i=1}^N)$.

Here $\mathcal{B}_\varepsilon$ denotes the ℓ_2-ball of radius ε. Implicit feature modification (IFM) removes components of the current representations that are used to discriminate positive and negative pairs. In other words, the embeddings of positive and negative samples are modified to remove well represented features. So, if the encoder is currently using a simple shortcut solution, IFM removes the features used, thereby encouraging the encoder to also discriminate instances using other features. By applying perturbations in the embedding space IFM can modify high level semantic features (see Fig. 4), which is extremely challenging when applying perturbations in input space. Since shortcut solutions still offer useful complementary information, we propose optimizing the the multi-task objective $\min_f \{\mathcal{L}(f) + \alpha \mathcal{L}_{\boldsymbol{\varepsilon}}(f)\}/2$ where $\mathcal{L}_{\boldsymbol{\varepsilon}} = \mathbb{E}\ell_{\boldsymbol{\varepsilon}}$ is the adversarial perturbed loss, and $\mathcal{L}$ the standard InfoNCE loss. For simplicity, all experiments set the balancing parameter $\alpha = 1$, and take all $\varepsilon^+, \varepsilon_i^-$ to be equal, and denote this single value by ε. Crucially, $\ell_{\boldsymbol{\varepsilon}}$ can be computed analytically, allowing efficient implementation.

Lemma 1. *For any $v, v^+, \{v_i^-\}_{i=1}^N \in \mathbb{R}^d$ we have,*

$$\nabla_{v_j^-} \ell = \frac{e^{v^\top v_j^-/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^N e^{v^\top v_i^-/\tau}} \cdot \frac{v}{\tau} \quad \textit{and} \quad \nabla_{v^+} \ell = \left(\frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^N e^{v^\top v_i^-/\tau}} - 1 \right) \cdot \frac{v}{\tau}.$$

In particular, $\nabla_{v_j^-} \ell \propto v$ and $\nabla_{v^+} \ell \propto -v$.

Proof. See Appendix B. □

The result shows that the adversary perturbs v_j^- (resp. v^+) in the direction of the anchor v (resp $-v$). Since the derivative directions are *independent* of $\{v_i^-\}_{i=1}^N$ and v^+, we can analytically compute optimal perturbations in $\mathcal{B}_\varepsilon$. Indeed, following the constant ascent direction shows the optimal updates are simply $v_i^- \leftarrow v_i^- + \varepsilon_i v$ and $v^+ \leftarrow v^+ - \varepsilon^+ v$. The positive (resp. negative) perturbations increase (resp. decrease) cosine similarity to the anchor $\text{sim}(v, v_i^- + \varepsilon_i v) \to 1$ as $\varepsilon_i \to \infty$ (resp. $\text{sim}(v, v^+ - \varepsilon^+ v) \to -1$ as $\varepsilon^+ \to \infty$). In Fig. 4 we visualize the newly synthesized v_i^-, v^+ and find meaningful interpolation of semantics. Plugging the update rules for v^+ and v_i^- into the point-wise InfoNCE loss yields,

$$\ell_{\boldsymbol{\varepsilon}}(v, v^+, \{v_i^-\}_{i=1}^N) = -\log \frac{e^{(v^\top v^+ - \varepsilon^+)/\tau}}{e^{(v^\top v^+ - \varepsilon^+)/\tau} + \sum_{i=1}^N e^{(v^\top v_i^- + \varepsilon_i)/\tau}}. \tag{2}$$

In other words, IFM amounts to simply perturbing the logits – reduce the positive logit by ε^+/τ and increase negative logits by ε_i/τ. From this we see that $\ell_{\boldsymbol{\varepsilon}}$ is automatically symmetrized in the positive samples: perturbing v instead of v^+ results in the exact same objective. Eqn. 2 suggests an importance-weighting interpretation for IFM: each negative sample is re-weighted by a factor $e^{\varepsilon_i/\tau}$ and positives by $e^{-\varepsilon^+/\tau}$.

3.1 Visualizing implicit feature modification

With implicit feature modification, newly synthesized data points do not directly correspond to any "true" input data point. However it is still possible to visualize the effects of implicit feature modification. To do this, assume access to a memory bank of input data $\mathcal{M} = \{x_i\}_i$. A newly synthesized sample s can be approximately visualized by retrieving the 1-nearest neighbour using cosine similarity $\arg\min_{x \in \mathcal{M}} \text{sim}(s, f(x))$ and viewing the image x as an approximation to s.

Fig. 4 shows results using a ResNet-50 encoder trained using MoCo-v2 on ImageNet1K using the training set as the memory bank. For positive pair v, v^+ increasing ε causes the semantics of v and v^+ to diverge. For $\varepsilon = 0.1$ a different car with similar pose and color is generated, for $\varepsilon = 0.2$ the pose and color then changes, and finally for $\varepsilon = 1$ the pose, color and type of vehicle changes. For negative pair v, v^- the reverse occurs. For $\varepsilon = 0.1$, v^- is a vehicle with similar characteristics (number of windows, color etc.), and with $\varepsilon = 0.2$, the pose of the vehicle v^+ aligns with v. Finally for $\varepsilon = 1$ the pose and color of the perturbed negative sample become aligned to the anchor v. In summary, implicit feature modification successfully *modifies the feature content in positive and negative samples*, thereby altering which features can be used to discriminate instances.

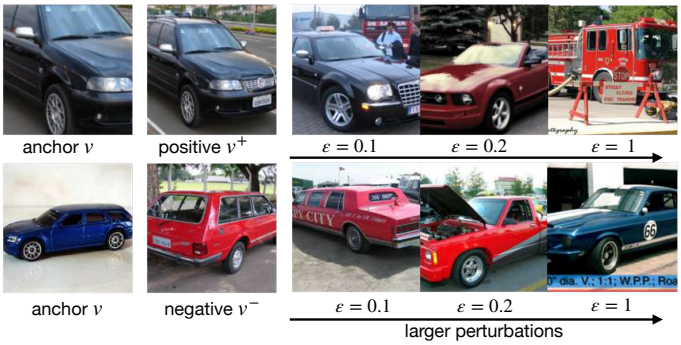

Figure 4: Visualizing implicit feature modification. **Top row:** progressively moving positive sample away from anchor. **Bottom row:** progressively moving negative sample away from anchor. In both cases, semantics such as color, orientation, and vehicle type are modified, showing the suitability of implicit feature modification for altering instance discrimination tasks.

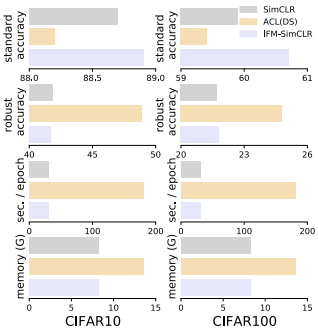

Figure 5: Comparison between IFM and ACL(DS). Under standard linear evaluation IFM performs best. ACL is suited to adversarial evaluation.

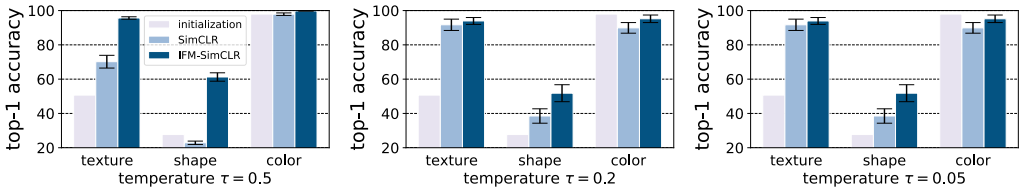

Figure 6: Trifeature dataset. Implicit feature modification reduces feature suppression, enhancing the representation of texture, shape and color features simultaneously. All results are average linear readout accuracy over three independent runs. All IFM runs use a fixed value $\varepsilon = 0.1$ to illustrate robustness to the choice of ε.

Related Work. Several works consider adversarial contrastive learning [19, 24, 26] using PGD (e.g. FGSM) attacks to alter samples in input space. Unlike our approach, PGD-based attacks require costly inner-loop optimization. Other work takes an adversarial viewpoint in input space for other self-supervised tasks e.g., rotations and jigsaws but uses an image-to-image network to simulate FGSM/PGD attacks [31], introducing comparable computation overheads. They note that low-level (i.e., pixel-level) shortcuts can be avoided using their method. However, all of these works differ from ours by applying attacks in input space, thereby focusing on lower-level features, whereas our latent space attacks modify high-level features. Fig. 5 compares IFM to this family of input-space adversarial methods by comparing to a top performing method ACL(DS) [24]. We find that ACL improves robust accuracy under ℓ_∞-attack on input space (see [24] for protocol details), whereas IFM improves standard accuracy (full details and discussion in Appdx. C.4). Synthesizing harder negatives in latent space using Mixup [51] has also been considered [25] but does not take an adversarial perspective. Other work, AdCo [20], also takes an adversarial viewpoint in latent space (see Tab. 1 for comparison to IFM). There are several differences to our approach. AdCo perturbs all negatives using the same weighted combination of all the queries over the current minibatch, whereas IFM perturbations are query specific. In other words, IFM makes instance discrimination harder point-wise, whereas AdCo perturbation makes the InfoNCE loss larger *on average* (see Fig. 4 for visualizations of instance dependent perturbation using IFM). AdCo also treats the negatives as learnable parameters, introducing $\sim 1M$ more parameters and $\sim 7\%$ computational overhead, while IFM has no computational overhead and is implemented with only two lines of code (see Tab. 1 for empirical comparison). Finally, no previous work makes the connection between suppression of semantic features and adversarial methods in contrastive learning (see Fig. 6).

4 Experimental results

Implicit feature modification (IFM) can be used with any InfoNCE-based contrastive framework, and we write IFM-SimCLR, IFM-MoCo-v2 etc. to denote IFM applied within a specific framework. Code for IFM will be released publicly, and is also available in the supplementary material.

4.1 Does implicit feature modification help avoid feature suppression?

We study the effect IFM has on feature suppression by training ResNet-18 encoders for 200 epochs with $\tau \in \{0.05, 0.2, 0.5\}$ on the Trifeature dataset [16]. All results are averaged over three independent runs, with IFM always using $\varepsilon = 0.1$ for simplicity. Fig. 6 shows that IFM improves the linear readout accuracy across *all* three features for all temperature settings. The capability of IFM to

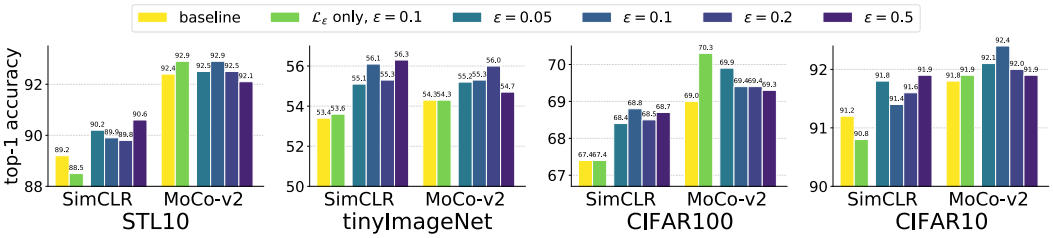

Figure 7: IFM improves linear readout performance on all datasets for all $\varepsilon \in \{0.05, 0.1, 0.2\}$ compared to SimCLR and MoCo-v2 baselines. Protocol uses 400 epochs of training with ResNet-50 backbone.

enhance the representation of all features – i.e. reduce reliance on shortcut solutions – is an important qualitative difference compared to tuning temperature τ or using hard negatives, which Fig. 3 shows can only *trade-off* which features are learned.

4.2 Performance on downstream tasks

Sec. 3.1 and Sec. 4.1 demonstrate that implicit feature modification is adept at altering high-level features of an input, and combats feature suppression. This section shows that these desirable traits translate into improved performance on object classification and medical imaging tasks.

Experimental setup for classification tasks. Having observed the positive effect IFM has on feature suppression, we next test if this feeds through to improved performance on real tasks of interest. We benchmark using both SimCLR and MoCo-v2 [5, 7] with standard data augmentation [5]. All encoders have ResNet-50 backbones and are trained for 400 epochs (with the exception of on ImageNet100, which is trained for 200 epochs). All encoders are evaluated using the test accuracy of a linear classifier trained on the full training dataset (see Appdx. C.2 for full setup details).

Classification tasks. Results given in Fig. 7 and Tab. 1 find that every value of $0 < \varepsilon \leq 0.2$ *improves performance across all datasets using both MoCo-v2 and SimCLR frameworks*. We find that optimizing $\mathcal{L}_\varepsilon$ (76.0% average score across all eight runs in Fig. 7) performs similarly to the standard contrastive loss (75.9% average score), and does worse than the IFM loss $(\mathcal{L} + \mathcal{L}_\varepsilon)/2$. This suggests that $\mathcal{L}$ and $\mathcal{L}_\varepsilon$ learn complementary features. Tab. 1 benchmarks IFM on ImageNet100 [42] using MoCo-v2, observing improvements of 0.9%. We also compare results on ImageNet100 to AdCo [20], another adversarial method for contrastive learning. We adopt the official code and use the exact same training and finetuning hyperparameters as for MoCo-v2 and IFM. For the AdCo-specific hyperparamters – negatives learning rate lr_{neg} and negatives temperature τ_{neg} – we use a grid search and report the best result. We search over all combinations $lr_{\text{neg}} \in \{1, 2, 3, 4\}$ and $\tau_{\text{neg}} \in \{0.02, 0.1\}$, which includes the AdCo default ImageNet1K recommendations $lr_{\text{neg}} = 3$ and $\tau_{\text{neg}} = 0.02$ [20]. That the resulting AdCo performance of 78.9% is slightly below MoCo-v2 serves to highlight the sensitivity of AdCo to its method-specific hyperparameters, and the challenge of tuning them. In contrast, IFM is robust to the choice of ε: all values $\varepsilon \in \{0.05, 0.1, 0.2\}$ were found to boost performance across all datasets and all frameworks. We emphasize that the MoCo-v2 baseline performance of 80.5% on ImageNet100 is very strong. In previouos work the highest ImageNet100 linear readout using 200 epochs of MoCo-v2 training we are aware of reported is 78% [25]. We refer the reader to Appdx. C.5 for details on our hyperparameter settings.

–	MoCo-v2	AdCo [20]	IFM-MoCo-v2		
ε	N/A	N/A	0.05	0.1	0.2
Top-1	80.5%	78.9%	81.1%	**81.4%**	80.9%
Top-5	95.6%	94.5%	**95.8%**	95.5%	95.5%

Table 1: Linear readout on ImageNet100. IFM improves over MoCo-v2 for all settings of ε.

Medical images. To evaluate our method on a modality differing significantly from object-based images we consider the task of learning representations of medical images. We benchmark using the approach proposed by [40] which is a variant of MoCo-v2 that incorporates the anatomical context in the medical images. We evaluate our method on the COPDGene dataset [36], which is a multi-center observational study focused on the genetic epidemiology of Chronic obstructive pulmonary disease (COPD). See Appdx. C.6 for full background details on the COPDGene dataset, the five COPD related outcomes we use for evaluation, and our implementation. We perform regression analysis for continuous outcomes in terms of coefficient of determination (R-square), and logistic regression to predict ordinal outcomes and report the classification accuracy and the *1-off* accuracy, i.e., the probability of the predicted category is within one class of true value.

Method	logFEV1pp	logFEV$_1$FVC	CLE	CLE *1-off*	Para-septal	Para-septal *1-off*	mMRC	mMRC *1-off*
Loss	R-Square		Accuracy (%)					
$\mathcal{L}$ (baseline)	$0.566_{\pm.005}$	$0.661_{\pm.005}$	$49.6_{\pm0.4}$	$81.8_{\pm0.5}$	$55.7_{\pm0.3}$	$84.4_{\pm0.2}$	$50.4_{\pm0.5}$	$72.5_{\pm0.3}$
$\mathcal{L}_\varepsilon, \varepsilon = 0.1$	$0.591_{\pm.008}$	$0.681_{\pm.008}$	$49.4_{\pm0.4}$	$81.9_{\pm0.3}$	$55.6_{\pm0.3}$	$85.1_{\pm0.2}$	$50.3_{\pm0.8}$	$72.7_{\pm0.4}$
IFM, $\varepsilon = 0.1$	$\mathbf{0.615_{\pm.005}}$	$\mathbf{0.691_{\pm.006}}$	$48.2_{\pm0.8}$	$80.6_{\pm0.4}$	$55.3_{\pm0.4}$	$84.7_{\pm0.3}$	$50.4_{\pm0.5}$	$72.8_{\pm0.2}$
IFM, $\varepsilon = 0.2$	$0.595_{\pm.006}$	$0.683_{\pm.006}$	$48.5_{\pm0.6}$	$80.5_{\pm0.6}$	$55.3_{\pm0.3}$	$85.1_{\pm0.1}$	$49.8_{\pm0.8}$	$72.0_{\pm0.3}$
IFM, $\varepsilon = 0.5$	$0.607_{\pm.006}$	$0.683_{\pm.005}$	$49.6_{\pm0.4}$	$82.0_{\pm0.3}$	$54.9_{\pm0.2}$	$84.7_{\pm0.2}$	$\mathbf{50.6_{\pm0.4}}$	$\mathbf{73.1_{\pm0.2}}$
IFM, $\varepsilon = 1.0$	$0.583_{\pm.005}$	$0.675_{\pm.006}$	$\mathbf{50.0_{\pm0.5}}$	$\mathbf{82.9_{\pm0.4}}$	$\mathbf{56.3_{\pm0.6}}$	$\mathbf{85.7_{\pm0.2}}$	$50.3_{\pm0.6}$	$71.9_{\pm0.3}$

Table 2: Linear readout performance on COPDGene dataset. The values are the average of 5-fold cross validation with standard deviations. IFM yields improvements on all phenotype predictions.

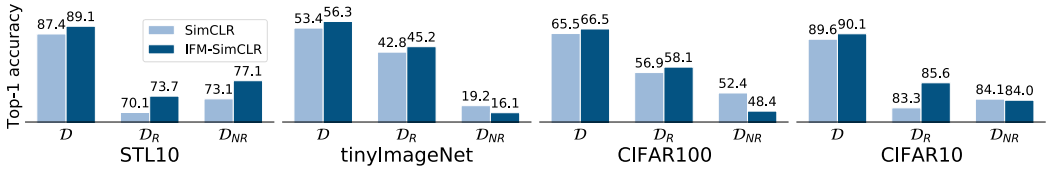

Figure 8: Label $\{\mathcal{D}, \mathcal{D}_{\text{R}}, \mathcal{D}_{\text{NR}}\}$ indicates which dataset was used to train the linear readout function. Improved performance of IFM on standard data $\mathcal{D}$ can be attributed to improved representation of *robust* features $\mathcal{D}_{\text{R}}$. See Sec. 4.3 for construction of robust ($\mathcal{D}_{\text{R}}$) and non-robust ($\mathcal{D}_{\text{NR}}$) feature datasets.

Tab. 2 reports results. For fair comparison we use same experimental configuration for the baseline approach [40] and our method. We find that IFM yields improvements on all outcome predictions. The gain is largest on spirometry outcome prediction, particularly logFEV1pp with improvement of 8.7% with $\varepsilon = 0.1$. Although other settings of ε achieve optimal performance on certain downstream tasks, we note that at least one of $\varepsilon = 0.5$ or 1.0 improves performance on all tasks.

4.3 Further study on the impact of IFM on feature learning

This section further studies the effect implicit feature modification has on *what type* of features are extracted. Specifically, we consider the impact on learning of robust (higher-level) vs. non-robust features (pixel-level features). Our methodology, which is similar to that of Ilyas et al. [22] for deep supervised learning, involves carefully perturbing inputs to obtain non-robust features.

Constructing non-robust features. Given encoder f we finetune a linear probe (classifier) h on-top of f using training data (to avoid smoothing effects we do not use data augmentation). Once h is trained, we consider each labeled example (x, y) from training data $\mathcal{D}_{\text{train}} \in \{\text{tinyImageNet, STL10, CIFAR10, CIFAR100}\}$. A hallucinated target label t is sampled uniformly at random, and we perturb $x = x_0$ until $h \circ f$ predicts t using repeated FGSM attacks [12] $x_k \leftarrow x_{k-1} - \varepsilon \text{sign}(\nabla_x \ell(h \circ f(x_{k-1}), t))$. At each step we check if $\arg\max_i h \circ f(x_k)_i = t$ (we use the maximum of logits for inference) and stop iterating and set $x_{\text{adv}} = x_k$ for the first k for which the prediction is t. This usually takes no more than a few FGSM steps with $\varepsilon = 0.01$. We form a dataset of "robust" features by adding (x_{adv}, y) to $\mathcal{D}_R$, and a dataset of "non-robust" features by adding (x_{adv}, t) to $\mathcal{D}_{NR}$. To a human the pair (x_{adv}, t) will look mislabeled, but for the encoder x_{adv} contains features predictive of t. Finally, we re-finetune (i.e. re-train) linear classifier g using $\mathcal{D}_R$ (resp. $\mathcal{D}_{NR}$) as training data.

Fig. 8 compares accuracy of the re-finetuned models on a test set of *standard* $\mathcal{D}_{\text{test}}$ examples (no perturbations are applied to the test set). Note that $\mathcal{D}_R, \mathcal{D}_{NR}$ depend on the original encoder f. When re-finetuning f we always use datasets $\mathcal{D}_R, \mathcal{D}_{NR}$ formed via FGSM attacks on f itself. So there is one set $\mathcal{D}_R, \mathcal{D}_{NR}$ for SimCLR, and another set for IFM. Fig. 8 shows that IFM achieves superior generalization ($\mathcal{D}$) compared to SimCLR *by better representing robust features* ($\mathcal{D}_R$). Representation of non-robust features ($\mathcal{D}_{NR}$) is similar for IFM (55.5% average across all datasets) and SimCLR (56.7% average). IFM is juxtaposed to the supervised adversarial training of Madry et al., which *sacrifices* standard supervised performance in exchange for not using non-robust features [30, 44].

5 Discussion

This work studies the relation between contrastive instance discrimination and feature learning. While we focus specifically on contrastive learning, it would be of interest to also study feature learning for other empirically successful self-supervised methods [1, 6, 13, 49]. Understanding differences in feature learning biases between different methods may inform which methods are best suited to a given task, as well as point the way to further improved self-supervised techniques.

References

[1] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. *preprint arXiv:2105.04906*, 2021.

[2] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2018.

[3] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9912–9924, 2020.

[4] Ting Chen and Lala Li. Intriguing properties of contrastive losses. *preprint arXiv:2011.02803*, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Int. Conference on Machine Learning (ICML)*, pages 10709–10719, 2020.

[6] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *preprint arXiv:2003.04297*, 2020.

[8] Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory (COLT)*, pages 1305–1338, 2020.

[9] Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8765–8775, 2020.

[10] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *Int. Conf. on Learning Representations (ICLR)*, 2019.

[11] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Int. Conf. on Learning Representations (ICLR)*, 2015.

[13] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 21271–21284, 2020.

[14] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. Int. Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 297–304, 2010.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738, 2020.

[16] Katherine L Hermann and Andrew K Lampinen. What shapes feature representations? Exploring datasets, architectures, and training. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9995–10006, 2020.

[17] Katherine L Hermann, Ting Chen, and Simon Kornblith. The origins and prevalence of texture bias in convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 19000–19015, 2019.

[18] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *Int. Conf. on Learning Representations (ICLR)*, 2019.

[19] Chih-Hui Ho and Nuno Nvasconcelos. Contrastive learning with adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 17081–17093, 2020.

[20] Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[21] Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *preprint arXiv:2103.10427*, 2021.

[22] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 125–136, 2019.

[23] Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *Int. Conf. on Learning Representations (ICLR)*, 2018.

[24] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 16199–16210, 2020.

[25] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 21798–21809, 2020.

[26] Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[27] Jason D Lee, Qi Lei, Nikunj Saunshi, and Jiacheng Zhuo. Predicting what you already know helps: Provable self-supervised learning. *preprint arXiv:2008.01064*, 2020.

[28] Tianhong Li, Lijie Fan, Yuan Yuan, Hao He, Yonglong Tian, and Dina Katabi. Information-preserving contrastive learning for self-supervised representations. *preprint arXiv:2012.09962*, 2020.

[29] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *Int. Conf. on Learning Representations (ICLR)*, 2020.

[30] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Int. Conf. on Learning Representations (ICLR)*, 2018.

[31] Matthias Minderer, Olivier Bachem, Neil Houlsby, and Michael Tschannen. Automatic shortcut removal for self-supervised representation learning. In *International Conference on Machine Learning*, pages 6927–6937, 2020.

[32] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *Int. Conf. on Learning Representations (ICLR)*, 2021.

[33] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *preprint arXiv:1807.03748*, 2018.

[34] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016.

[35] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *Int. Conference on Machine Learning (ICML)*, pages 5389–5400, 2019.

[36] Elizabeth A Regan, John E Hokanson, James R Murphy, Barry Make, David A Lynch, Terri H Beaty, Douglas Curran-Everett, Edwin K Silverman, and James D Crapo. Genetic epidemiology of COPD (COPDGene) study design. *COPD: Journal of Chronic Obstructive Pulmonary Disease*, 7(1):32–43, 2011.

[37] Joshua Robinson, Stefanie Jegelka, and Suvrit Sra. Strength from weakness: Fast learning using weak supervision. In *Int. Conference on Machine Learning (ICML)*, pages 8127–8136, 2020.

[38] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *Int. Conf. on Learning Representations (ICLR)*, 2021.

[39] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[40] Li Sun, Ke Yu, and Kayhan Batmanghelich. Context matters: Graph-based self-supervised representation learning for medical images. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

[41] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Int. Conf. on Learning Representations (ICLR)*, 2014.

[42] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Europ. Conference on Computer Vision (ECCV)*, 2020.

[43] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6827–6839, 2020.

[44] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *Int. Conf. on Learning Representations (ICLR)*, 2018.

[45] Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[46] Feng Wang, Huaping Liu, Di Guo, and Fuchun Sun. Unsupervised representation learning by invariance propagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3510–3520, 2020.

[47] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Int. Conference on Machine Learning (ICML)*, pages 9574–9584, 2020.

[48] Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *preprint arXiv:2104.07713*, 2021.

[49] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Int. Conference on Machine Learning (ICML)*, 2021.

[50] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *Int. Conference on Machine Learning (ICML)*, pages 7472–7482, 2019.

[51] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Int. Conf. on Learning Representations (ICLR)*, 2018.

[52] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Europ. Conference on Computer Vision (ECCV)*, pages 649–666, 2016.

[53] Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? In *Int. Conf. on Learning Representations (ICLR)*, 2021.

[54] Roland S Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. In *Int. Conference on Machine Learning (ICML)*, 2021.

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] See abstract and section 1.
   (b) Did you describe the limitations of your work? [Yes] See discussion in the Appendix (App. D).
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See discussion in App. D and E.
   (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]
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] Code will be provided in supplementary material. The Trifeature dataset is not our intellectual property so we will not releases it without the owners permission.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (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)? [Yes]
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] We cite the creators for the Trifeature dataset, code for MoCo-v2, ACL(DS), AdCo, etc.
   (b) Did you mention the license of the assets? [Yes] We mention the license in Appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] All datasets are either de-identified (CIFAR/ImageNet etc.) or synthetic.
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]