Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

HHS Public Access

Author manuscript

*Adv Neural Inf Process Syst*. Author manuscript; available in PMC 2022 May 10.

Published in final edited form as:

*Adv Neural Inf Process Syst*. 2021 December ; 34: 4974–4986.

# Can contrastive learning avoid shortcut solutions?

**Joshua Robinson**[†], **Li Sun**[*], **Ke Yu**[*], **Kayhan Batmanghelich**[*], **Stefanie Jegelka**[†], **Suvrit Sra**[†]

[†]Massachusetts Institute of Technology

[*]University of Pittsburgh

## 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. The code is available at: https://github.com/joshr17/IFM.

## 1 Introduction

Representations trained with contrastive learning are adept at solving various vision tasks including classification, object detection, instance segmentation, and more [5, 15, 44]. 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, 43] and failure outside the training distribution [2, 37]. Simplicity bias has various potential sources [11] including training methods [8, 29, 41] 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?*

Correspondence to Joshua Robinson (joshrob@mit.edu).

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

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.

**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, 44, 51]. Contrastive learning methods, such as SimCLR and MoCo [5, 15, 45], have been especially successful. Beyond contrastive methods, other Siamese approaches that avoid representation collapse without explicitly use of negatives have also been proposed [6, 13, 51].

Pretext task design has been at the core of progress in self-supervised learning. Previously popular tasks include image colorization [54] and inpainting [35], and theoretical work shows pre-trained encoders can provably generalize if a pretext task necessitates the learning of features that solve downstream tasks [27, 39]. In contrastive learning, augmentation strategies are a key design component [5, 48, 50], as are negative mining techniques [9, 15, 25, 40]. 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 [55], 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.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

## 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 lead to poor 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 $\mathcal{Z}^1, \ldots, \mathcal{Z}^n$ 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 $[n] = \{1, \ldots, n\}$. 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 $\lambda$. 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 input data $x$ to the surface of the unit sphere $\mathbb{S}^{d-1} = \left\{u \in \mathbb{R}^d : \|u\|_2 = 1\right\}$ 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 $\nu$ 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 $z^S, \tilde{z}^S \in \mathcal{Z}^S$, we have $\mu(\cdot \mid z^S) = \mu(\cdot \mid \tilde{z}^S)$.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

2. *f distinguishes S* if for any pair of distinct $z^S, \bar{z}^S \in \mathcal{Z}^S$, measures $\mu(\cdot \mid z^S), \mu(\cdot \mid \bar{z}^S)$ 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. We consider training an encoder $f: \mathcal{X} \to \mathbb{S}^{d-1}$ to optimize the InfoNCE loss [33, 14],

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

where $\tau$ 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"), and that there are no collisions: $a(x) \neq a'(x')$ for all $a$, $a'$, and all $x \neq x'$. 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 [40, 49, 56], we analyze the loss as the number of negatives goes to infinity,

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

We subtract $\log m$ to ensure the limit is finite, and use $x^-$ to denote a random sample with the same distribution as $x_i^-$. Prop. 1 (proved in App. A) shows that, assuming the marginals $p_j$ are uniform, the InfoNCE loss is optimized both by encoders that suppress feature $j$, and by encoders that distinguish $j$.

**Proposition 1.**—*Suppose that $p_j$ is uniform on $\mathcal{Z}^j = \mathbb{S}^{d-1}$ for all $j \in [n]$. Then for any feature $j \in [n]$ there exists an encoder $f_{supp}$ that suppresses feature $j$ and encoder $f_{disc}$ that discriminates $j$ but both attain* $\min_{f:\ measurable} \mathcal{L}(f)$.

The condition that $p_j$ is uniformly distributed on $\mathcal{Z}^j = \mathbb{S}^{d-1}$ is similar to conditions used in previous work [56]. 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 can occur,

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Prop. 1 also suggests another counter-intuitive possibility: *lower InfoNCE loss may actually lead to worse performance on some tasks*.

To empirically study whether this possibility manifests 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 possible 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 [5]. 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.2 for full 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.

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 [49] 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 instance discrimination

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 $\tau$ in the InfoNCE loss (Eqn. 1). Smaller $\tau$ places higher importance on positive an negative pairs with high similarity [47].
2. Hard negative sampling method of Robinson et al. [40], which uses importance sampling to sample harder negatives. The method introduces a hardness concentration parameter $\beta$, with larger $\beta$ corresponding to harder negatives (see [40] for full details).

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Results reported in Fig. 3 (also Fig. 13 in App. C.2) show that varying instance discrimination difficulty—i.e., varying temperature $\tau$ or hardness concentration $\beta$—enables trade-offs between which features are represented. On Trifeature, easier instance discrimination (large $\tau$, small $\beta$) 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 $\tau$, small $\beta$). We hypothesize 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).—***Suppose that $p_j$ is uniform on $\mathcal{X}^j = \mathbb{S}^{d-1}$ for all $j \in [n]$. Further, for $S \subseteq [n]$ suppose that $x$, $x^+$, $\{x_i^-\}_i$ are conditioned on the event that they have the same features $S$. Then 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

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

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 $x$, $x^+$, $\{x_i^-\}_{i=1}^m$ 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,

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

**Definition 2 (Implicit feature modification).**

Given budget $\varepsilon \in \mathbb{R}_+^m$, and encoder $f : \mathcal{X} \to \mathbb{S}^d$, an adversary removes features from $f$ that discriminates batch $x$, $x^+$, $\{x_i^-\}_{i=1}^m$ by maximizing the point-wise InfoNCE loss,

$$\ell_\varepsilon\left(v, v^+, \{v_i^-\}_{i=1}^m\right) = \max_{\delta^+ \in \mathcal{B}_\varepsilon +, \left\{\delta_i^- \in \mathcal{B}_{\varepsilon_i}\right\}_{i=1}^m} \ell\left(v, v^+ + \delta^+, \{v_i^- + \delta_i^-\}_{i=1}^m\right).$$

Here $\mathcal{B}_\varepsilon$ denotes the $\ell_2$-ball of radius $\varepsilon$. 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. In order to learn new features using the perturbed loss while still learning potentially complementary information using the original InfoNCE objective, we propose optimizing the the multi-task objective $\min_f \{\mathcal{L}(f) + \alpha \mathcal{L}_\varepsilon(f)\}/2$ where $\mathcal{L}_\varepsilon = \mathbb{E}\,\ell_\varepsilon$ is the adversarial perturbed loss, and $\mathcal{L}$ the standard InfoNCE loss. For simplicity, all experiments set the balancing parameter $\alpha = 1$ unless explicitly noted, and all take $\varepsilon^+$, $\varepsilon_i^-$ to be equal, and denote this single value by $\varepsilon$. Crucially, $\ell_\varepsilon$ can be computed analytically, allowing efficient implementation.

**Lemma 1.**

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

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

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

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

This expression 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 $\{v_i^-\}_{i=1}^m$ and $v^+$, we can analytically compute optimal perturbations in $\mathscr{B}_\varepsilon$. 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 $\mathrm{sim}(v, v_i^- + \varepsilon_i v) \to 1$ as $\varepsilon_i \to \infty$ (resp. $\mathrm{sim}(v, v^+ - \varepsilon^+ v) \to -1$ 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,

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

In other words, IFM amounts to simply perturbing the logits – reduce the positive logit by $\varepsilon^+/\tau$ and increase negative logits by $\varepsilon_i/\tau$. From this we see that $\ell_\varepsilon$ is automatically symmetrized in the positive samples: perturbing $v$ instead of $v^+$ results in the exact same objective. Eqn. 2 shows that IFM re-weights each negative sample by a factor $e^{\varepsilon_i/\tau}$ and positive samples 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 $\mathscr{M} = \{x_i\}_i$. A newly synthesized sample $s$ can be approximately visualized by retrieving the 1-nearest neighbour using cosine similarity $\arg\min_{x \in \mathscr{M}} \mathrm{sim}(s, f(x))$ 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 $\varepsilon$ 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.

**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

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

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 $\ell_\infty$-attack on input space (see [24] for protocol details), whereas IFM improves standard accuracy (full details and discussion in Appdx. C.3). Synthesizing harder negatives in latent space using Mixup [53] 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 ~ $1M$ more parameters and ~ 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 enhance the representation of all features – i.e. reduce reliance on shortcut solutions – is an important contrast compared to tuning temperature $\tau$ 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

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

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.4 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 $\mathscr{L}_\varepsilon$ (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 $\text{loss}(\mathscr{L} + \mathscr{L}_\varepsilon)/2$. This suggests that $\mathscr{L}$ and $\mathscr{L}_\varepsilon$ learn complementary features. Tab. 1 benchmarks IFM on ImageNet100 [44] 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 $lr_{\text{neg}}$ and negatives temperature $\tau_{\text{neg}}$ – 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 $\varepsilon$: 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.4.1 for details on our hyperparameter settings.

**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 [42] which is a variant of MoCo-v2 that incorporates the anatomical context in the medical images. We evaluate our method on the COPDGene dataset [38], which is a multi-center observational study focused on the genetic epidemiology of Chronic obstructive pulmonary disease (COPD). See Appdx. C.5 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.

Tab. 2 reports results. For fair comparison we use same experimental configuration for the baseline approach [42] 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 $\varepsilon$ 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.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

### 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 $\mathscr{D}_{\text{train}} \in \{$ 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] $x_k \leftarrow x_{k-1} - \epsilon \text{sign}(\nabla_x \ell(h \circ f(x_{k-1}), t))$. 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 $\epsilon = 0.01$. We form a dataset of "robust" features by adding $(x_{\text{adv}}, y)$ to $\mathscr{D}_R$, and a dataset of "non-robust" features by adding $(x_{\text{adv}}, t)$ to $\mathscr{D}_{NR}$. To a human the pair $(x_{\text{adv}}, t)$ will look mislabeled, but for the encoder $x_{\text{adv}}$ contains features predictive of $t$. Finally, we re-finetune (i.e. re-train) linear classifier $g$ using $\mathscr{D}_R$ (resp. $\mathscr{D}_{NR}$) as training data.

Fig. 8 compares accuracy of the re-finetuned models on a test set of *standard* $\mathscr{D}_{\text{test}}$ examples (no perturbations are applied to the test set). Note that $\mathscr{D}_R$, $\mathscr{D}_{NR}$ depend on the original encoder $f$. When re-finetuning $f$ we always use datasets $\mathscr{D}_R$, $\mathscr{D}_{NR}$ formed via FGSM attacks on $f$ itself. So there is one set $\mathscr{D}_R$, $\mathscr{D}_{NR}$ for SimCLR, and another set for IFM. Fig. 8 shows that IFM achieves superior generalization ($\mathscr{D}$) compared to SimCLR *by better representing robust features* ($\mathscr{D}_R$). Representation of non-robust features ($\mathscr{D}_{NR}$) 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, 46].

## 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, 51]. Understanding differences in feature learning biases between different methods may inform which methods are best suited for a given task, as well as point the way to further improved self-supervised techniques.

## Acknowledgments

We warmly thank Katherine Hermann and Andrew Lampinen for sharing the Trifeature dataset.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

# Appendix

# Appendix

## A Proofs for Section 2

In this section we give proofs for all the results in Sec. 2, which explores the phenomenon of feature suppression in contrastive learning using the InfoNCE loss. We invite the reader to consult Sec. 2.1 for details on any notation, terminology, or formulation details we use.

Recall, for a measure $\nu$ on a space $\mathcal{U}$ and a measurable map $h: \mathcal{U} \to \mathcal{V}$ let $h\#\nu$ denote the *pushforward* $h\#\nu(V) = \nu(h^{-1}(V))$ of a measure $\nu$ 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. We now recall the definition of feature suppression and distinction.

**Definition 1.**

Consider an encoder $f: \mathcal{X} \to \mathbb{S}^{d-1}$ and features $j \subseteq [n]$. For each $z^j \in \mathcal{Z}^S$, let $\mu(\cdot|z^S) = (f \circ g)\#\lambda(\cdot|z^j)$ 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 $z^S, \tilde{z}^S \in \mathcal{Z}^S$, we have $\mu(\cdot \mid z^S), \mu(\cdot \mid \tilde{z}^S)$.
2. *$f$ distinguishes $S$* if for any pair of distinct $z^S, \tilde{z}^S \in \mathcal{Z}^S$, measures $\mu(\cdot \mid z^S), \mu(\cdot \mid \tilde{z}^S)$ have disjoint support.

Suppression of features $S$ is thereby captured by the characteristic of distributing points in the same way on the sphere independently of what value $z^S$ takes. Feature distinction, meanwhile, is characterized by being able to partition the sphere into different pieces, each corresponding to a different value of $z^S$. Other (perhaps weaker) notions of feature distinction may be useful in other contexts. However here our goal is to establish that it is possible for InfoNCE optimal encoders both to suppress features in the sense of Def. 1, but also to separate concepts out in a desirable manner. For this purpose we found this strong notion of distinguishing to suffice.

Before stating and proving the result, recall the limiting InfoNCE loss that we analyze,

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

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 [49] we denote the first term by $\mathcal{L}_{\text{align}}$ and the second term by the "uniformity loss" $\mathcal{L}_{\text{unif}}$, so $\mathcal{L} = \mathcal{L}_{\text{align}} + \mathcal{L}_{\text{unif}}$.

**Proposition 1.**

Suppose that $p_j$ is uniform on $\mathcal{Z}^j = \mathbb{S}^{d-1}$. For any feature $j \in [n]$ there exists an encoder $f_{\text{supp}}$ that suppresses feature $j$ and encoder $f_{\text{disc}}$ that discriminates $j$ but both attain $\min_{f:\ \text{measurable}} \mathcal{L}(f)$.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

**Proof.—**The existence of the encoders $f_{\text{supp}}$ and $f_{\text{disc}}$ is demonstrated by constructing explicit examples. Before defining $f_{\text{supp}}$ and $f_{\text{disc}}$ themselves, we begin by constructing a family $\{f^k\}_{k\in[n]}$ of optimal encoders.

Since $g$ is injective, we know there exists a left inverse $h:\mathcal{X}\to\mathcal{Z}$ such that $h\circ g(z)=z$ for all $z\in\mathcal{Z}$. For any $k\in[n]$ let $\Pi^k:\mathcal{Z}\to\mathbb{S}^{d-1}$ denote the projection $\Pi^k(z)=z^k$. Since $p_k$ is uniform on the sphere $\mathbb{S}^{d-1}$, we know that $\Pi^k\circ h\circ g(z)=z^j$ is uniformly distributed on $\mathbb{S}^{d-1}$. Next we partition the space $\mathcal{X}$. Since we assume that for all $a\neq a'$ and $z\neq z'$ that $a(z)\neq a'(z')$, the family $\{\mathcal{X}_z\}_{z\in\mathcal{Z}}$ where $\mathcal{X}_z=\{a\circ g(z):z\in\mathcal{Z}\}$ is guaranteed to be a partition (and in particular, disjoint). We may therefore define an encoder $f_k:\mathcal{X}\to\mathbb{S}^{d-1}$ to be equal to $f_k(x)=\Pi^k\circ h\circ g(z)=z^k$ for all $x\in\mathcal{X}_z$.

First we check that this $f_k$ is optimal. Since for any $z$, and any $a\sim\mathcal{A}$, by definition we have $a\circ g(z)\in\mathcal{X}_z$, we have that $f_k(x)=f_k(a(x))$ almost surely, so $\mathcal{L}_{\text{align}}(f_k)=0$ is minimized. To show $f_k$ minimizes $\mathcal{L}_{\text{unif}}$ note that the uniformity loss can be re-written as

$$\begin{aligned}\mathcal{L}_{\text{unif}}(f_k)&=\int_a\int_z\log\int_{a^-}\int_{z^-}e^{f_k\circ a(g(z))^\top f_k\circ a^-(g(z^-))/\tau}\lambda(\mathrm{d}z)\lambda(\mathrm{d}z^-)\mathcal{A}(\mathrm{d}a)\mathcal{A}(\mathrm{d}a^-)\\&=\int_z\log\int_{z^-}e^{f_k\circ g(z)^\top f_k\circ g(z^-)/\tau}\lambda(\mathrm{d}z)\lambda(\mathrm{d}z^-)\\&=\int_{\mathbb{S}^{d-1}}\log\int_{\mathbb{S}^{d-1}}e^{u^\top v/\tau}\mu(\mathrm{d}u)\mu(\mathrm{d}v)\end{aligned}$$

where $\mu=f_k\circ g\#\lambda$ is the pushforward measure on $\mathbb{S}^{d-1}$, and the second equality follows from the fact that $\mathcal{L}_{\text{align}}(f_k)=0$. Theorem 1 of Wang and Isola [49] establishes that the operator,

$$\mu\mapsto\int_{\mathbb{S}^{d-1}}\log\int_{\mathbb{S}^{d-1}}e^{u^\top v/\tau}\mu(\mathrm{d}u)\mu(\mathrm{d}v)$$

is minimized over the space of Borel measures on $\mathbb{S}^{d-1}$ if and only if $\mu=\sigma_d$, the uniform distribution on $\mathbb{S}^{d-1}$, as long as such an $f$ exists. However, since by construction $f_k(x)=\Pi^k\circ h\circ g(z)=z^k$ is uniformly distributed on $\mathbb{S}^{d-1}$, we know that $(f_k\circ g)\#\lambda=\sigma_d$, and hence that $f_k$ minimizes $\mathcal{L}_{\text{align}}$ and $\mathcal{L}_{\text{unif}}$ and hence also the sum $\mathcal{L}=\mathcal{L}_{\text{align}}+\mathcal{L}_{\text{unif}}$.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

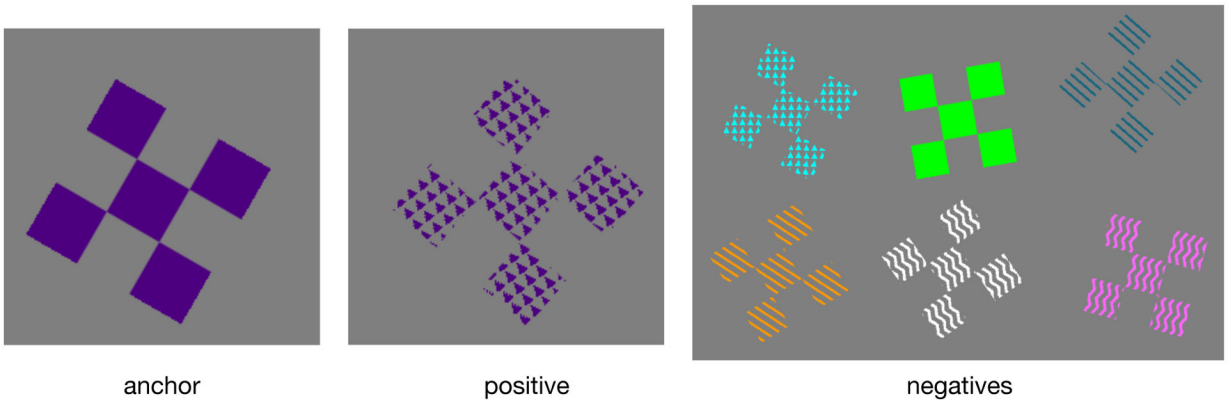



**Figure 9:**
Visual illustration of Prop. 2 using Trifeature samples [16]. In this example the shape feature is kept fixed across all positives and negatives ($S = \{\text{shape}\}$), with color and texture to varying. As a consequence, the positive pair cannot be discriminated from negatives using the shape feature. The encoder must learn color features in order to identify this positive pair. In other words, if a given set of features (e.g. $S = \{\text{shape}\}$) are constant across positive and negative pairs, then instance discrimination task demands the use of features in the compliment (e.g. {color,texture}).

Recall that we seek encoder $f_{\text{supp}}$ that suppress feature $j$, and $f_{\text{disc}}$ that distinguishes feature $j$. We have a family $\{f^k\}_{k\in[n]}$ that are optimal, and select the two encoders we seen from this collection. First, for $f_{\text{supp}}$ define $f_{\text{supp}} = f^k$ for any $k \neq j$. Then by construction $f_{\text{supp}}(x) = z^k$ (where $x \in \mathcal{X}_z$) depends only on $z^k$, which is independent of $z^j$. Due to independence, we therefore know that for any pair $z^j, \bar{z}^j \in \mathcal{Z}^j$, we have $\mu(\cdot \mid z^j) = \mu(\cdot \mid \bar{z}^j)$, i.e., that $f_{\text{supp}}$ is optimal but suppresses feature $j$. Similarly, simply define $f_{\text{disc}} = f^j$. So $f_{\text{disc}}(x) = z^j$ where $x \in \mathcal{X}_z$, and for any $z^j, \bar{z}^j \in \mathcal{Z}^j$ with $z^j \neq \bar{z}^j$ the pushforwards $\mu(\cdot|z^j)$, $\mu(\cdot \mid \bar{z}^j)$ are the Dirac measures $\delta_{z_j}$, $\delta_{\bar{z}_j}$, which are disjoint. □

Next we present a result showing that, under suitable conditions that guarantee that minimizers exists, any $f$ optimizing the InfoNCE loss is guaranteed to suppress features $S$ if all batches $x_1^+$, $x_2^+$, $\{x_i^-\}_{i=1}^N$ are have the same features $S$ (but that the value $z_S$ taken is allowed to vary). This result captures the natural intuition that if a feature cannot be used to discriminate instances, then it will not be learned by the InfoNCE loss. Before reading the proposition, we encourage the reader to see Fig. 9 for an intuitive visual illustration of the idea underlying Prop. 2 using Trifeature samples [16].

However, this result also points to a way to manage which features are learned by an encoder, since if $f$ is guaranteed *not* to learn features $S$, then necessarily $f$ must use other features to solve the instance discrimination task. This insight lays the foundation for the implicit feature modification technique, which perturbs the embedding $v = f(x)$ to remove information that $f$ uses to discriminate instances – and then asks for instance discrimination using *both* to original embedding, and the modified one – with the idea that this encourages $f$ to learn new features that it previously suppressed.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

**Proposition 2.**

For a set $S \subseteq [n]$ of features let

$$\mathscr{L}_S(f) = \mathscr{L}_{\text{align}}(f) + \mathbb{E}_X + \left[ -\log \mathbb{E}_{X^-} \left[ e^{f(x^+)^\top f(x^-)} \mid z^S = z^S - \right] \right]$$

denote the (limiting) InfoNCE conditioned on $x^+$, $x^-$ having the same features $S$. Suppose that $p_j$ is uniform on $\mathscr{X}^j = \mathbb{S}^{d-1}$ for all $j \in [n]$. Then the infimum $\inf \mathscr{L}_S$ is attained, and every $f \in \min_{f'} \mathscr{L}_S(f')$ suppresses features $S$ almost surely.

**Proof.—**By Prop 2, we know that for each $z^S$ there is a measurable $f$ such that $\mathscr{L}_{\text{align}}(f) = 0$ and $f$ achieves perfect uniformity $(f \circ g)\#\lambda(\cdot | z^S) = \sigma_d$ conditioned on $z^S$. So consider such an $f$. Since $\mathscr{L}_{\text{align}}(f) = 0$ we may write,

$$\begin{aligned}
\mathscr{L}_S(f) &= \mathbb{E}_X + \left[ -\log \mathbb{E}_{X^-} \left[ e^{f(x^+)^\top f(x^-)} \mid z^S = z^S - \right] \right] \\
&= \mathbb{E}_{z^S} \mathbb{E}_{z^S} - \left[ -\log \mathbb{E}_{z^-} \left[ e^{f \circ g(z)^\top f \circ g(z^-)} \mid z^S = z^S - \right] \right] \\
&= \mathbb{E}_{z^S} \mathscr{L}\left(f; z^S\right).
\end{aligned}$$

Where we have introduced the conditional loss function

$$\mathscr{L}\left(f; z^S\right) = \mathbb{E}_{z^S} - \left[ -\log \mathbb{E}_{z^-} \left[ e^{f \circ g(z)^\top f \circ g(z^-)} \mid z^S = z^S - \right] \right]$$

We shall show that any minimizer $f$ of $\mathscr{L}_S$ is such that $f$ minimizes $\mathscr{L}\left(f; z^S\right)$ for all values of $z^S$. To show this notice that $\min_f \mathscr{L}_S(f) = \min_f \mathbb{E}_{z^S} \mathscr{L}\left(f; z^S\right) \geq \mathbb{E}_{z^S} \min_f \mathscr{L}\left(f; z^S\right)$ and if there is an $f$ such that $f$ minimizes $\mathscr{L}\left(f; z^S\right)$ for each $z^S$ then the inequality is tight. So we make it our goal to show that there is an $f$ such that $f$ minimizes $\mathscr{L}\left(f; z^S\right)$ for each $z^S$.

For fixed $z^S$, by assumption there is an $f_{z^S}$ such that $(f_{z^S} \circ g)\#\lambda\left(\cdot \mid z^S\right) = \sigma_d$. That is, $f_{z^S}$ achieves perfect uniformity given $z^S$. Theorem 1 of Wang and Isola [49] implies that $f_{z^S}$ must minimize $\mathscr{L}\left(f; z^S\right)$. Given $\{f_{z^S}\}_{z^S}$ we construct an $f : \mathscr{X} \to \mathbb{S}_\tau^{d-1}$ that minimizes $\mathscr{L}\left(f; z^S\right)$ *for all* $z^S$. By injectivity of $g$ we may partition $\mathscr{X}$ into pieces $\cup_{z^S \in \mathscr{Z}^S} \mathscr{X}_{z^S}$ where $\mathscr{X}_{z^S} = \left\{ x : x = g\left(\left(z^S, z^{S^c}\right)\right) \text{ for some } z^{S^c} \in \mathscr{Z}^{S^c} \right\}$. So we may simply define $f$ on domain $\mathscr{X}$ as follows: $f(x) = f_{z^S}(x)$ if $x \in \mathscr{X}_{z^S}$.

This construction allows us to conclude that the minimum of $\mathscr{L}_S$ is attained, and any minimizer $f$ of $\mathscr{L}_S$ also minimizes $\mathscr{L}\left(f; z^S\right)$ for each $z^S$. By Theorem 1 of Wang and Isola [49] any such $f$ is such that $(f_{z^S} \circ g)\#\lambda\left(\cdot \mid z^S\right) = \sigma_d$ for all $z^S$, which immediately implies that $f$ suppresses features $S$. □

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

## B Computation of implicit feature modification updates

This section gives detailed derivations of two simple but key facts used in the development of IFM. The first result derives an analytic expression for the gradient of the InfoNCE loss with respect to positive sample in latent space, and the second result computes the gradient with respect to an arbitrary negative sample. The analysis is very simple, only requiring the use of elementary tools from calculus. Despite its simplicity, this result is very important, and forms the core of our approach. It is thanks to the analytic expressions for the gradients of the InfoNCE loss that we are able to implement our adversarial method *without introducing any memory or run-time overheads*. This is a key distinction from previous adversarial methods for contrastive learning, which introduce significant overheads (see Fig. 5).

Recall the statement of the lemma.

**Lemma 2.**

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

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

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

**Proof.**

Both results follow from direct computation. First we compute $\nabla_{v_j^-} \ell \left( v, v^+, \{v_i^-\}_{i=1}^m \right)$. Indeed, for any $j \in \{1, 2, \ldots, m\}$ we have,

$$\begin{aligned} \nabla_{v_j^-} \left\{ -\log \frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \right\} &= \nabla_{v_j^-} \log \left\{ e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau} \right\} \\ &= \frac{\nabla_{v_j^-} \left\{ e^{v^\top v^+} + \sum_{i=1}^m e^{v^\top v_i^-/\tau} \right\}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \\ &= \frac{e^{v^\top v_j^-/\tau} \cdot v/\tau}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} \end{aligned}$$

the quantity $\frac{e^{v^\top v_j^-/\tau}}{e^{v^\top v^+/\tau} + \sum_{i=1}^m e^{v^\top v_i^-/\tau}} > 0$ is a strictly positive scalar, allowing us to conclude the derivative $\nabla_{v_j^-} \ell$ is proportional to $v$. We also compute $\nabla_{v^+} \ell \left( v, v^+, \{v_i^-\}_{i=1}^m \right)$ in a similar fashion,

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

$$\nabla_v+\left\{-\log\frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau}+\sum_{i=1}^m e^{v^\top v_i^-/\tau}}\right\}=\nabla_v+\left\{-\log e^{v^\top v^+/\tau}\right\}+\nabla_v+\log\left\{e^{v^\top v^+/\tau}+\sum_{i=1}^m e^{v^\top v_i^-/\tau}\right\}$$
$$=-\frac{v}{\tau}+\frac{\nabla_v+\left\{e^{v^\top v^+/\tau}+\sum_{i=1}^m e^{v^\top v_i^-/\tau}\right\}}{e^{v^\top v^+/\tau}+\sum_{i=1}^m e^{v^\top v_i^-/\tau}}$$
$$=-\frac{v}{\tau}+\frac{e^{v^\top v^+/\tau}\cdot v/\tau}{e^{v^\top v^+/\tau}+\sum_{i=1}^m e^{v^\top v_i^-/\tau}}$$
$$=\left(\frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau}+\sum_{i=1}^m e^{v^\top v_i^-/\tau}}-1\right)\cdot\frac{v}{\tau}.$$

Since $0<\frac{e^{v^\top v^+/\tau}}{e^{v^\top v^+/\tau}+\sum_{i=1}^m e^{v^\top v_i^-/\tau}}<1$ we conclude in this case that the derivative $\nabla_v+\ell$ points in the direction $-v$. □

### B.1 Alternative formulations of implicit feature modification

This section contemplates two simple modifications to the IFM method with the aim of confirming that these modifications do not yield superior performance to the default proposed method. The two alternate methods focus around the following observation: IFM perturbs embeddings of unit length, and returns a modified version that will no longer be of unit length in general. We consider two alternative variations of IFM that yield normalized embeddings. The first is the most simple solution possible: simply re-normalize perturbed embeddings to have unit length. The second is slightly more involved, and involves instead applying perturbations *before* normalizing the embeddings. Perturbing unnormalized embeddings, then normalizing, guarantees the final embeddings have unit length. The key property we observed in the original formulation was the existence of an analytic, easily computable closed form expressions for the derivatives. This property enables efficient computation of newly synthesized "adversarial" samples in latent space. Here we derive corresponding formulae for the pre-normalization attack.

For clarity, we introduce the slightly modified setting in full detail. We are given positive pair $x$, $x^+$ and a batch of negative samples $\{x_i^-\}_{i=1}^m$ and denote their encodings via $f$ as $v = f(x)$, $v^+ = f(x^+)$, and $v_i^- = f(x_i^-)$ for $i = 1, \ldots m$ where we *do not* assume that $f$ returns normalized vectors. That is, $f$ is allowed to map to anywhere in the ambient latent space $\mathbb{R}^d$. The re-parameterized point-wise contrastive loss for this batch of samples is

$$\ell\left(v,v^+,\{v_i^-\}_{i=1}^m\right)=-\log\frac{e^{\mathrm{sim}(v,v^+)/\tau}}{e^{\mathrm{sim}(v,v^+)/\tau}+\sum_{i=1}^m e^{\mathrm{sim}(v,v_i^-)/\tau}},$$

where $\mathrm{sim}(u, v) = u \cdot v/\|u\|\|v\|$ denotes the cosine similarity measure. As before we wish to perturb $v^+$ and negative encodings $v_j^-$ to increase the loss, thereby making the negatives harder. Specifically we wish to solve

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

$\max_{\delta^+ \in \mathcal{B}_\epsilon +, \left\{\delta_i^- \in \mathcal{B}_{\epsilon_j}\right\}_{i=1}^m} \ell\left(v, v^+ + \delta^+, \{v_i^- + \delta_i^-\}_{i=1}^m\right)$. The following lemma provides the corresponding gradient directions.

**Lemma 3.**—*For any* $v$, $v^+$, $\{v_i^-\}_{i=1}^m \in \mathbb{R}^d$ *we have*

$$\nabla_{v_j^-} \ell \propto \frac{v}{\|v\|} - sim\left(v_j^-, v\right)\frac{v_j^-}{\|v_j^-\|} \quad and \quad \nabla_{v^+} \ell \propto \frac{v}{\|v\|} - sim\left(v^+, v\right)\frac{v^+}{\|v^+\|}.$$

To prove this lemma we rely on the following well-known closed form expression for the derivative of the cosine similarity, whose proof we omit.

**Lemma 4.**—$\nabla_v sim(v,u) = \frac{u}{\|v\|\|u\|} - sim(v,u)\frac{v}{\|v\|^2}$.

**Proof of Lemma 3.:** We compute,

$$\begin{aligned}\nabla_{v_j^-} \ell &= \nabla_{v_j^-} \log\left(e^{\mathrm{sim}\left(v, v^+\right)} + \sum_{i=1}^m e^{\mathrm{sim}\left(v, v_i^-\right)}\right) \\ &= \frac{e^{\mathrm{sim}\left(v, v_j^-\right)}}{e^{\mathrm{sim}\left(v, v^+\right)} + \sum_{i=1}^m e^{\mathrm{sim}\left(v, v_i^-\right)}} \cdot \nabla_{v_j^-} \mathrm{sim}\left(v, v_j^-\right)\end{aligned}$$

Using the formula for the derivative of the cosine similarity, we arrive at a closed form formula,

$$\begin{aligned}\nabla_{v_j^-} \ell &= \frac{e^{\mathrm{sim}\left(v, v_j^-\right)}}{e^{\mathrm{sim}\left(v, v^+\right)} + \sum_{i=1}^m e^{\mathrm{sim}\left(v, v_i^-\right)}} \cdot \left(\frac{v}{\|v_j^-\|\|v\|} - \mathrm{sim}\left(v_j^-, v\right)\frac{v_j^-}{\|v_j^-\|^2}\right). \\ &\propto \frac{v}{\|v\|} - \mathrm{sim}\left(v_j^-, v\right)\frac{v_j^-}{\|v_j^-\|}\end{aligned}$$

Similar computations yield

$$\begin{aligned}\nabla_{v^+} \ell &= -\nabla_{v^+} \log \frac{e^{\mathrm{sim}\left(v, v^+\right)}}{e^{\mathrm{sim}\left(v, v^+\right)} + \sum_{i=1}^m e^{\mathrm{sim}\left(v, v_j^-\right)}} \\ &= \nabla_{v^+}\left(-\mathrm{sim}\left(v, v^+\right) + \log\left(e^{\mathrm{sim}\left(v, v^+\right)} + \sum_{i=1}^m e^{\mathrm{sim}\left(v, v_i^-\right)}\right)\right) \\ &= \left(\frac{e^{\mathrm{sim}\left(v, v^+\right)}}{e^{\mathrm{sim}\left(v, v^+\right)} + \sum_{i=1}^m e^{\mathrm{sim}\left(v, v_i^-\right)}} - 1\right) \cdot \nabla_{v^+} \mathrm{sim}\left(v, v^+\right) \\ &= \left(\frac{e^{\mathrm{sim}\left(v, v^+\right)}}{e^{\mathrm{sim}\left(v, v^+\right)} + \sum_{i=1}^m e^{\mathrm{sim}\left(v, v_i^-\right)}} - 1\right) \cdot \left(\frac{v}{\|v^+\|\|v\|} - \mathrm{sim}\left(v^+, v\right)\frac{v^+}{\|v^+\|^2}\right) \\ &\propto \frac{v}{\|v\|} - \mathrm{sim}\left(v^+, v\right)\frac{v^+}{\|v^+\|}\end{aligned}$$

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Lemma 3 provides precisely the efficiently computable formulae for the derivatives we seek. One important difference between this pre-normalization case and the original setting is that the direction vector depends on $v_j^-$ and $v^+$ respectively. In the original (unnormalized) setting the derivatives depend only on $v$, which allowed the immediate and exact discovery of the worst case perturbations in an $\varepsilon$-ball. Due to these additional dependencies in the pre-normalized case the optimization is more complex, and must be approximated iteratively. Although only approximate, it is still computationally cheap since we have simple analytic expressions for gradients.

It is possible give an interpretation to the pre-normalization derivatives $\nabla_{v_j^-} \ell$ by considering the $\ell_2$ norm,

$$\begin{aligned}\|\nabla_{v_j^-}\|_2 &= \sqrt{\left(\frac{v}{\|v\|} - \frac{v^\top v_j^-}{\|v\|\|v_j^-\|}\frac{v_j^-}{\|v_j^-\|}\right)\cdot\left(\frac{v}{\|v\|} - \frac{v^\top v_j^-}{\|v\|\|v_j^-\|}\frac{v_j^-}{\|v_j^-\|}\right)} \\ &= \sqrt{1+\mathrm{sim}(v,v_i^-)^2 - 2\mathrm{sim}(v,v_i^-)^2} \\ &= \sqrt{1-\mathrm{sim}(v,v_i^-)^2}\end{aligned}$$

So, samples $v_i^-$ with higher cosine similarity with anchor $v$ receive smaller updates. Similar calculations for $v^+$ show that higher cosine similarity with anchor $v$ leads to larger updates. In other words, the pre-normalization version of the method automatically adopts an adaptive step size based on sample importance.

**B.1.1 Experimental results using alternative formulations—**In this section we test the two alternative implementations to confirm that these simple alternatives do not obtain superior performance to IFM. We consider only object-based images, so it remains possible that other modalities may benefit from alternate formulations. First note that $f$ encodes all points to the boundary of the same hypersphere, while perturbing $v_i^- \leftarrow v_i^- + \varepsilon_i v$ and $v^+ \leftarrow v^+ - \varepsilon_+ v$ moves adversarial samples off this hypersphere. We therefore consider normalizing all points again *after* perturbing (+ *norm*). The second method considers applying attacks *before* normalization (+ *pre-norm*), whose gradients were computed in the Lem. 3. It is still possible to compute analytic gradient expressions in this setting; we refer the reader to Appendix B.1 for full details and derivations. Results reported in Tab. 3, suggest that all versions improve over MoCov2, and both alternatives perform comparably to the default implementation based on Eqn. 2.

**Table 3:**

Linear readout performance of alternative latent space adversarial methods. We report the best performance over runs for $\varepsilon \in \{0.05, 0.1, 0.2, 0.5\}$. We find that the two modifications to IFM we considered do not improve performance compared to the default version of IFM.

| Dataset | MoCo-v2 | IFM-MoCo-v2 | | |
|---|---|---|---|---|
| – | – | default | + norm | + pre-norm |
| STL10 | 92.4% | 92.9% | 92.9% | **93.0%** |

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

| Dataset | MoCo-v2 | IFM-MoCo-v2 | | |
|---|---|---|---|---|
| – | – | default | + norm | + pre-norm |
| CIFAR10 | 91.8% | **92.4%** | 92.2% | 92.0% |
| CIFAR100 | 69.0% | **70.3%** | 70.1% | 70.2% |

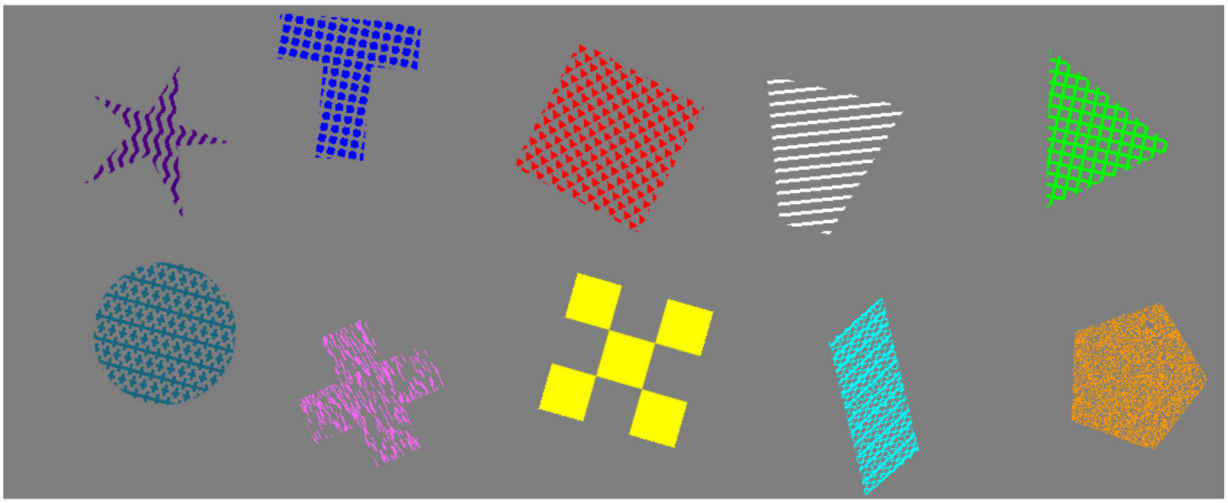


**Figure 10:**
Sample images from the Trifeature dataset [16]. There are three features: shape, color, and texture. Each feature has 10 different possible values. We show exactly one example of each feature.

# C Supplementary experimental results and details

## C.1 Hardware and setup

Experiments were run on two internal servers. The first consists of 8 NVIDIA GeForce RTX 2080 Ti GPUs (11GB). The second consists of 8 NVIDIA Tesla V100 GPUs (32GB). All experiments use the PyTorch deep learning framework [34]. Specific references to pre-existing code bases used are given in the relevant sections below.

## C.2 Feature suppression experiments

This section gives experimental details for all experiments in Sec. 2 in the main manuscript, the section studying the relation between feature suppression and instance discrimination.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

**Figure 11:**
Sample images from the STL-digits dataset. There are two features: object class, and MNIST digit. Both features have 10 different possible values.

### C.2.1 Datasets

**Trifeature [16]:** Introduced by Hermann and Lampinen, each image is 128 × 128 and has three features: color, shape, and texture each taking 10 values. For each (color, shape, texture) triplet (1000 in total) Trifeature contains 100 examples, forming a dataset of 100K examples in total. Train/val sets are obtained by a random 90/10 split. See Fig. 10, Appdx. C for sample images.

**STL10-digits dataset:** We artificially combine MNIST digits and STL10 object to produce data with two controllable semantic features. We split the STL10 image into a 3 × 3 grid, placing a copy of the MNIST digit in the center of each sector. This is done by masking all MNIST pixels with intensity lower than 100, and updating non-masked pixels in the STL10 image with the corresponding MNIST pixel value.

### C.2.2 Experimental protocols

**Training:** We train ResNet-18 encoders using SimCLR with batch size 512. We use standard data SimCLR augmentations [5], but remove grayscaling and color jittering when training on Trifeature in order to avoid corrupting color features. We use Adam optimizer, learning rate $1 \times 10^{-3}$ and weight decay $1 \times 10^{-6}$. Unless stated otherwise, the temperature $\tau$ is set to 0.5.

**Linear evaluation:** For fast linear evaluation we first extract features from the trained encoder (applying the same augmentations to inputs as used during pre-training) then use the LogisticRegression function in scikit-learn [36] to train a linear classifier. We use the Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm with a maximum iteration of 500 for training.

### C.2.3 Details on results

**Correlations Fig. 2:** For the Trifeature heatmap 33 encoders are used to compute correlations. The encoders are precisely encoders used to plot Fig. 3. Similarly, the 7 encoders used to generate the STL-digits heatmap are precisely the encoders whose training is shown in Fig. 13. When computing the InfoNCE loss for Fig. 2, for fair comparison all losses are computed using temperature normalization value $\tau = 0.5$. This is independent of

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

training, and is necessary only in evaluation to ensure loss values are comparable across different temperatures.

Fig. 13 displays results for varying instance discrimination difficult on the STL-digits dataset. These results are complementing the Trifeature results in Fig. 3 in Sec. 2 in the main manuscript. For STL-digits we report only a single training run per hyperparameter setting since performance is much more stable on STL-digits compared to Trifeature (see Fig. 12). See Sec. 2 for discussion of STL-digits results, which are qualitatively the same as on Trifeature. Finally, Fig. 14 shows the effect of IFM on encoders trained on STL-digits. As with Trifeature, we find that IFM improves the performance on suppressed features (STL10), but only slightly. Unlike hard instance discrimination methods, IFM does not harm MNIST performance in the process.

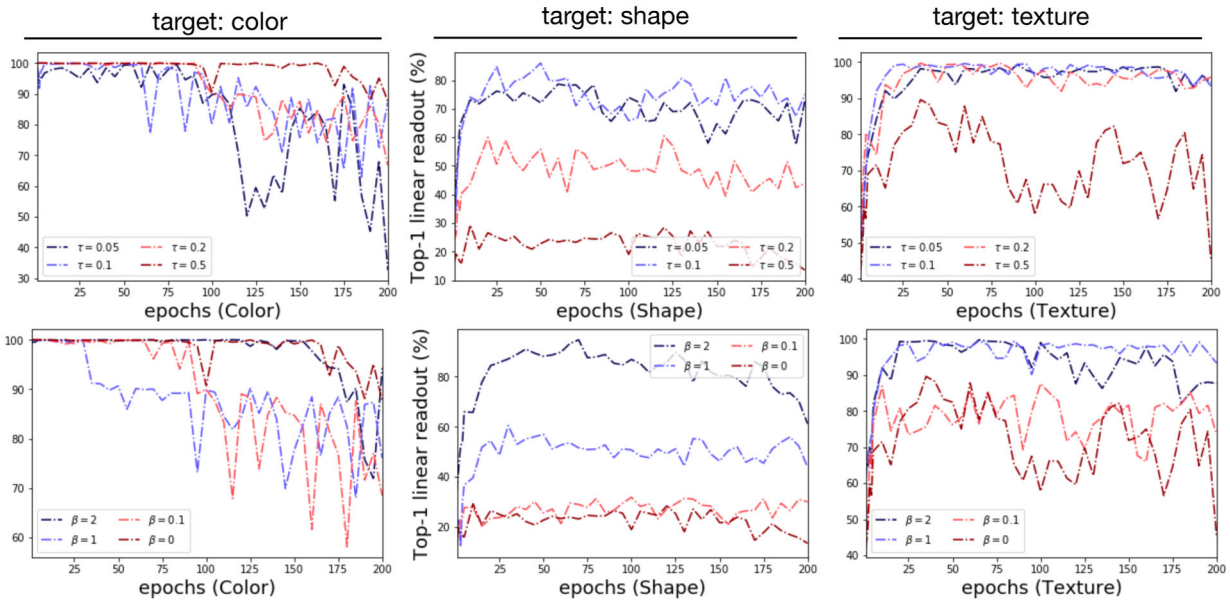
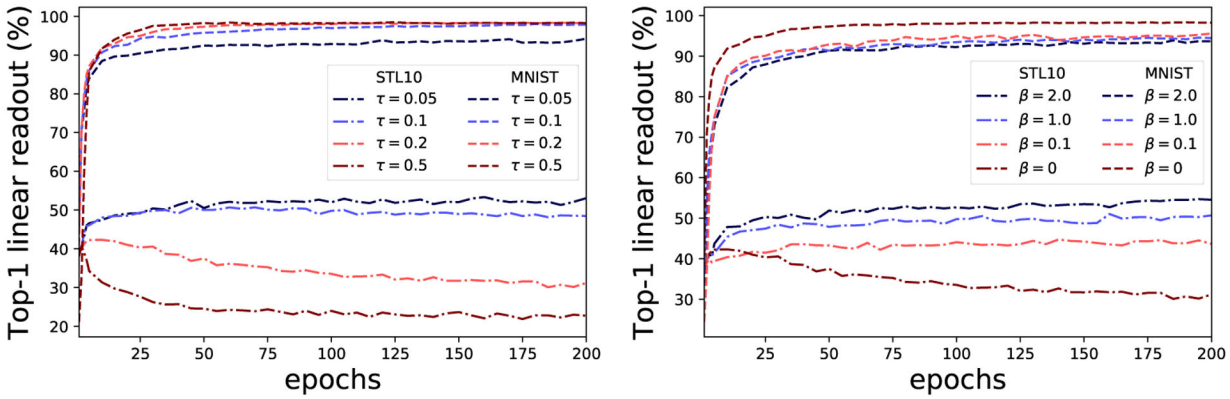



**Figure 12:**
Single run experiments showing training dynamics of Trifeature contrastive training. Linear readout performance on color prediction is particularly noisy.



Author Manuscript Author Manuscript Author Manuscript Author Manuscript

**Figure 13:**

STL-digits dataset. **Left:** performance on STL10 and MNIST linear readout for different temperature $\tau$ values. **Right:** performance on STL10 and MNIST linear readout for different hardness concentration $\beta$ values [40]. In both cases harder instance discrimination (smaller $\tau$, bigger $\beta$) improves STL10 performance at the expense of MNIST. When instance discrimination is too easy (big $\tau$, small $\beta$) STL10 features are *suppressed*: achieving worse linear readout after training than at initialization.

## C.3 Comparing IFM and ACL(DS)

**H4—**We give details for Fig 5. Similarly to concurrent work [19, 26], ACL [24] directly performs PGD attacks in input space. We compare to the top performing version ACL(DS) – which uses a duel stream structure and combines standard and adversarial loss terms. We use the official ACL implementation[1] and for fair comparison run IFM by changing only the loss function. All hyperparameters are kept the same for both runs, and follow the ACL recommendations.

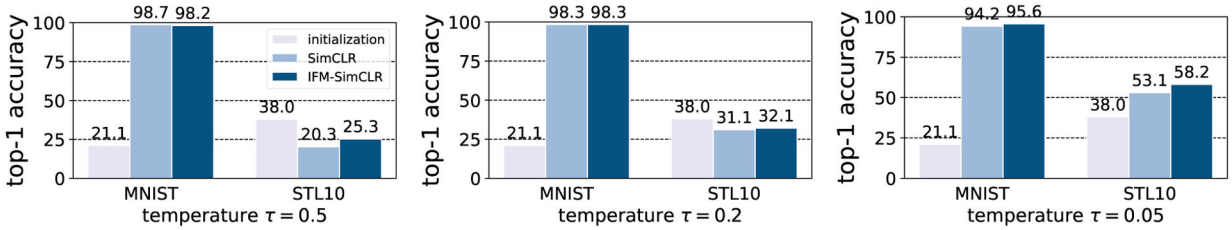



**Figure 14:**

STL-digits dataset. Implicit feature modification reduces feature suppression, enhancing the representation of both MNIST and STL10 features simultaneously. All IFM runs use a fixed value $\varepsilon = 0.1$, and loss $\mathscr{L} + 0.5 \cdot \mathscr{L}_{\varepsilon}$ (i.e. weighting parameter $\alpha = 0.5$) to illustrate robustness to the choice of parameters.

**Training:** We use the SimCLR framework with a ResNet-18 backbone and train for 1000 epochs. We use a base learning rate of 5 with cosine annealing scheduling and batch size 512. LARS optimizer is used. For ACL(DS), we run the PGD for 5 steps in the pre-training stage following the practice of [24].

**Linear evaluation:** We use two schemes to evaluate the quality of learnt representation: standard accuracy and robust accuracy. Robust accuracy reports the accuracy in the setting where an adversary is allowed to apply an $\ell_\infty$ attack to each input. For standard accuracy, we only finetune the last layer and test on clean images following the practice of MoCo-v2 [7]. The initial learning rate is set as 0.1 and we tune for 100 epochs for CIFAR10, 25 epochs for CIFAR100 respectively. An SGD optimizer is used to finetune the model. We use a step scheduler that decreases the learning rate by a factor of 10 after epochs: 40, 60 for CIFAR10; 15, 20 for CIFAR100 respectively. For robust accuracy, we finetune the model using the loss in TRADE [52], and evaluate classification accuracy on adversarially

[1] https://github.com/VITA-Group/Adversarial-Contrastive-Learning

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

perturbed testing images. We use the same hyperparameters as ACL [24] for adversarial finetuning. We perform experiments on CIFAR10 and CIFAR100 and the results are shown in Fig. 5.

**Results:** See Fig. 5 in the main manuscript for the results. There are significant qualitative differences between the behaviour of IFM and ACL(DS). IFM improves (standard) linear readout accuracy with zero memory or compute time cost increase, whereas ACL(DS) has improved adversarial linear readout performance, but at the cost of worse standard linear readout and 2× memory and 6× time per epoch. This shows that these two method are addressing two distinct problems. ACL(DS) is suitable for improving the adversarial robustness of a model, whereas IFM improves the generalization of a representation.

### C.4 Object classification experiments

We first describe the protocol used for evaluating IFM on the following datasets: CIFAR10, CIFAR100, STL10, tinyImageNet. For simplicity, the objective weighting parameter is fixed at $\alpha = 1$. Official train/val data splits are used for all datasets.

**Training—**All encoders have ResNet-50 backbones and are trained for 400 epochs with temperature $\tau = 0.5$ for SimCLR and $\tau = 0.1$ for MoCo-v2. Encoded features have dimension 2048 and are followed by a two layer MLP projection head with output dimension 128. Batch size is taken to be 256, yielding negative batches of size $m = 510$ for SimCLR. For MoCo-v2, we use a queue size of $k = 4096$ (except for STL10 dataset we use $k = 8192$), and we use batch size of 256 for CIFAR10, CIFAR100 and tinyImageNet, 128 for STL10. For both SimCLR and MoCo-v2 we use the Adam optimizer.

SimCLR uses initial learning rate $1 \times 10^{-3}$ and weight decay $1 \times 10^{-6}$ for CIFAR10, CIFAR100 and tinyImageNet, while STL10 uses $1 \times 10^{-1}$ learning rate, and weight decay $5 \times 10^{-4}$ (since we found these settings boosted performance by around 5% in absolute terms). MoCo-v2 training uses weight decay $5 \times 10^{-4}$, and an initial learning rate $3 \times 10^{-2}$ for CIFAR10 and CIFAR100; and learning rate $1 \times 10^{-1}$ for STL10 and tinyImageNet. Cosine learning rate schedule is used for MoCo-v2.

**Linear evaluation—**Evaluation uses test performance of a linear classifier trained ontop of the learned embedding (with embedding model parameters kept fixed) trained for 100 epochs.

For SimCLR, the batch size is set as 512, and the linear classifier is trained using the Adam optimizer with learning rate $1 \times 10^{-3}$ and weight decay $1 \times 10^{-6}$, and default PyTorch settings for other hyperparameters. For CIFAR10 and CIFAR100 the same augmentations as SimCLR are used for linear classifier training, while for STL10 and tinyImageNet no augmentations were used (since we found this can help improve performance).

For MoCo-v2, the batch size is set as 256. Training uses SGD with initial learning rate set to 30, momentum is set as 0.9 and a scheduler that reduces the learning rate by a factor of 10% at epoch 30 and 60. The weight decay is 0. For CIFAR10 and CIFAR100, we normalize images with mean of [0.4914, 0.4822, 0.4465] and standard deviation of [0.2023,

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

0.1994, 0.2010]. For STL10 and tinyImageNet, we normalize images with mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225]. The same augmentations as the official MoCO-v2 implementation are used for linear classifier training.

**C.4.1 ImageNet100—**We adopt the official MoCo-v2 code[2] (CC-BY-NC 4.0 license), modifying only the loss function. For comparison with AdCo method, we adopt the official code[3] (MIT license) and use the exact same hyperparmeters as for MoCo-v2. For the AdCo specific parameters we perform a simple grid search for the following two hyperparameters: negatives learning rate $lr_{\text{neg}}$ and negatives temperature $\tau_{\text{neg}}$. 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]. The result reported for AdCo in Tab. 1 is the best performance over all 8 runs.

**Training:** We use ResNet-50 backbones, and train for 200 epochs. We use a base learning rate of 0.8 with cosine annealing scheduling and batch size 512. The MoCo momentum is set to 0.99, and temperature to $\tau = 0.2$. All other hyperparameters are kept the same as the official defaults.

**Linear evaluation:** We train for 60 epochs with batch size 128. We use initial learning rate of 30.0 and a step scheduler that decreases the learning rate by a factor of 10 after epochs: 30, 40, 50. All other hyperparameters are kept the same as the official MoCo-v2 defaults.

As noted in the manuscript, our combination of training and linear evaluation parameters leads to 80.5% top-1 linear readout for standard MoCo-v2, and 81.4% with IFM-MoCo-v2. The standard MoCo-v2 performance of 80.5% is, to the best of our knowledge, state-of-the-art performance on ImageNet100 using 200 epoch training with MoCo-v2. For comparison, we found that using the default recommended MoCo-v2 ImageNet1k parameters (both training and linear evaluation) achieves ImageNet100 performance of 71.8%. This choice of parameters maybe useful for other researchers using MoCo-v2 as a baseline on ImageNet100.

### C.5 COPDGene dataset

The dataset [38] in our experiments includes 9,180 subjects. Each subject has a high-resolution inspiratory CT scan and five COPD related outcomes, including two continuous spirometry measures: (1) FEV1pp: the forced expiratory volume in one second, (2) $FEV_1$/FVC: the FEV1pp and forced vital capacity (FVC) ratio, and three ordinal variables: (1) six-grade centrilobular emphysema (CLE) visual score, (2) three-grade paraseptal emphysema (Para-septal) visual score, (3) five-grade dyspnea symptom (mMRC) scale. The dataset is publicly available.

For fair comparison, we use the same encoder and data augmentation described in the baseline approach [42]. We set the representation dimension to 128 in all experiments. For simplicity, instead of using a GNN, we use average pooling to aggregate patch

[2] https://github.com/facebookresearch/moco
[3] https://github.com/maple-research-lab/AdCo

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

representations into image representation. The learning rate is set as 0.01. We use Adam optimizer and set momentum as 0.9 and weight decay as $1 \times 10^{-4}$. The batch size is set as 128, and the model is trained for 10 epochs.

### C.6 Further discussion of feature robustness experiments (Sec. 4.3)

Ilyas et al. [22] showed that deep networks richly represent so-called "non-robust" features, but that adversarial training can be used to avoid extracting non-robust features at a modest cost to downstream performance. Although in-distribution performance is harmed, Ilyas et al. argue that the reduction in use of non-robust features – which are highly likely to be statistical coincidences due to the high dimensionality of input data in computer vision – may be desirable from the point of view of trustworthiness of a model under input distribution shifts. In this section we consider similar questions on the effect implicit feature modification on learning of robust vs. non-robust features during self-supervised pre-training.

Compared to supervised adversarial training [22, 30] our approach has the key conceptual difference of being applied in feature space. As well as improved computation efficiency (no PGD attacks required) Fig. 8 shows that this difference translates into different behavior when using implicit feature modification. Instead of suppressing non-robust features as Ilyas et al. observe for supervised representations, IFM *enhances the representation of robust features*. This suggests that the improved generalization of encoders trained with IFM can be attributed to improved extraction of features aligned with human semantics (robust features). However, we also note that IFM has no significant effect on learning of non-robust features. In Appdx. D we discuss the idea of combining IFM with adversarial training methods to get the best of both worlds.

## D Discussion of limitations and possible extensions

While our work makes progress towards understanding, and controlling, feature learning in contrastive self-supervised learning, there still remain many open problems and questions. First, since our proposed implicit feature modification method acts on embedded points instead of raw input data it is not well suited to improving $\ell_p$ robustness, and similarly is not suited to removing pixel-level shortcut solutions. Instead our method focuses on high-level semantic features. It would be valuable to study the properties of our high-level method used in conjunction with existing pixel-level methods.

Second, while we show that our proposed implicit feature modification method is successful in improving the representation of multiple features simultaneously, our method does not admit an immediate method for determining *which* features are removed during our modification step (i.e. which features are currently being used to solve the instance discrimination task). One option is to manually study examples using the visualization technique we propose in Sec. 3.1.

## References

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

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

2. Beery Sara, Van Horn Grant, and Perona Pietro. Recognition in terra incognita. In Proceedings of the European Conference on Computer Vision (ECCV), pages 456–473, 2018.
3. Caron Mathilde, Misra Ishan, Mairal Julien, Goyal Priya, Bojanowski Piotr, and Joulin Armand. Unsupervised learning of visual features by contrasting cluster assignments. In Advances in Neural Information Processing Systems (NeurIPS), pages 9912–9924, 2020.
4. Chen Ting and Li Lala. Intriguing properties of contrastive losses. preprint arXiv:2011.02803, 2020.
5. Chen Ting, Kornblith Simon, Norouzi Mohammad, and Hinton Geoffrey. A simple framework for contrastive learning of visual representations. In Int. Conference on Machine Learning (ICML), pages 10709–10719, 2020.
6. Chen Xinlei and He Kaiming. Exploring simple siamese representation learning. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
7. Chen Xinlei, Fan Haoqi, Girshick Ross, and He Kaiming. Improved baselines with momentum contrastive learning. preprint arXiv:2003.04297, 2020.
8. Chizat Lenaic and Bach Francis. 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. Chuang Ching-Yao, Robinson Joshua, Yen-Chen Lin, Torralba Antonio, and Jegelka Stefanie. Debiased contrastive learning. In Advances in Neural Information Processing Systems (NeurIPS), pages 8765–8775, 2020.
10. Geirhos Robert, Rubisch Patricia, Michaelis Claudio, Bethge Matthias, Wichmann Felix A, and Brendel Wieland. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In Int. Conf. on Learning Representations (ICLR), 2019.
11. Geirhos Robert, Jacobsen Jörn-Henrik, Michaelis Claudio, Zemel Richard, Brendel Wieland, Bethge Matthias, and Wichmann Felix A. Shortcut learning in deep neural networks. Nature Machine Intelligence, 2(11):665–673, 2020.
12. Goodfellow Ian J, Shlens Jonathon, and Szegedy Christian. Explaining and harnessing adversarial examples. In Int. Conf. on Learning Representations (ICLR), 2015.
13. Grill Jean-Bastien, Strub Florian, Altché Florent Tallec Corentin, Tallec Corentin, Richemond Pierre H, Buchatskaya Elena, Doersch Carl, Pires Bernardo Avila, Guo Zhaohan Daniel 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. Gutmann Michael and Hyvärinen Aapo. 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. He Kaiming, Fan Haoqi, Wu Yuxin, Xie Saining, and Girshick Ross. Momentum contrast for unsupervised visual representation learning. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 9729–9738, 2020.
16. Hermann Katherine L and Lampinen Andrew K. What shapes feature representations? Exploring datasets, architectures, and training. In Advances in Neural Information Processing Systems (NeurIPS), pages 9995–10006, 2020.
17. Hermann Katherine L, Chen Ting, and Kornblith Simon. The origins and prevalence of texture bias in convolutional neural networks. In Advances in Neural Information Processing Systems (NeurIPS), pages 19000–19015, 2019.
18. Hjelm R Devon, Fedorov Alex, Lavoie-Marchildon Samuel, Grewal Karan, Bachman Phil, Trischler Adam, and Bengio Yoshua. Learning deep representations by mutual information estimation and maximization. In Int. Conf. on Learning Representations (ICLR), 2019.
19. Ho Chih-Hui and Nvasconcelos Nuno. Contrastive learning with adversarial examples. In Advances in Neural Information Processing Systems (NeurIPS), pages 17081–17093, 2020.
20. Hu Qianjiang, Wang Xiao, Hu Wei, and Qi Guo-Jun. 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. Huh Minyoung, Mobahi Hossein, Zhang Richard, Cheung Brian, Agrawal Pulkit, and Isola Phillip. The low-rank simplicity bias in deep networks. preprint arXiv:2103.10427, 2021.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

22. Ilyas Andrew, Santurkar Shibani, Tsipras Dimitris, Engstrom Logan, Tran Brandon, and Madry Aleksander. Adversarial examples are not bugs, they are features. In Advances in Neural Information Processing Systems (NeurIPS), pages 125–136, 2019.
23. Jacobsen Jörn-Henrik, Behrmann Jens, Zemel Richard, and Bethge Matthias. Excessive invariance causes adversarial vulnerability. In Int. Conf. on Learning Representations (ICLR), 2018.
24. Jiang Ziyu, Chen Tianlong, Chen Ting, and Wang Zhangyang. Robust pre-training by adversarial contrastive learning. In Advances in Neural Information Processing Systems (NeurIPS), pages 16199–16210, 2020.
25. Kalantidis Yannis, Sariyildiz Mert Bulent, Pion Noe, Weinzaepfel Philippe, and Larlus Diane. Hard negative mixing for contrastive learning. In Advances in Neural Information Processing Systems (NeurIPS), pages 21798–21809, 2020.
26. Kim Minseon, Tack Jihoon, and Hwang Sung Ju. Adversarial self-supervised contrastive learning. In Advances in Neural Information Processing Systems (NeurIPS), 2020.
27. Jason D Lee Qi Lei, Saunshi Nikunj, and Zhuo Jiacheng. Predicting what you already know helps: Provable self-supervised learning. preprint arXiv:2008.01064, 2020.
28. Li Tianhong, Fan Lijie, Yuan Yuan, He Hao, Tian Yonglong, and Katabi Dina. Information-preserving contrastive learning for self-supervised representations. preprint arXiv:2012.09962, 2020.
29. Lyu Kaifeng and Li Jian. Gradient descent maximizes the margin of homogeneous neural networks. In Int. Conf. on Learning Representations (ICLR), 2020.
30. Madry Aleksander, Makelov Aleksandar, Schmidt Ludwig, Tsipras Dimitris, and Vladu Adrian. Towards deep learning models resistant to adversarial attacks. In Int. Conf. on Learning Representations (ICLR), 2018.
31. Minderer Matthias, Bachem Olivier, Houlsby Neil, and Tschannen Michael. Automatic shortcut removal for self-supervised representation learning. In International Conference on Machine Learning, pages 6927–6937, 2020.
32. Nguyen Thao, Raghu Maithra, and Kornblith Simon. 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. van den Oord Aaron, Li Yazhe, and Vinyals Oriol. Representation learning with contrastive predictive coding. preprint arXiv:1807.03748, 2018.
34. Paszke Adam, Gross Sam, Massa Francisco, Lerer Adam, Bradbury James, Chanan Gregory, Killeen Trevor, Lin Zeming, Gimelshein Natalia, Antiga Luca, et al. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems (NeurIPS), 2019.
35. Pathak Deepak, Krahenbuhl Philipp, Donahue Jeff, Darrell Trevor, and Efros Alexei A. Context encoders: Feature learning by inpainting. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2536–2544, 2016.
36. Pedregosa Fabian, Varoquaux Gaël, Gramfort Alexandre, Michel Vincent, Thirion Bertrand, Grisel Olivier, Blondel Mathieu, Prettenhofer Peter, Weiss Ron, Dubourg Vincent, et al. Scikit-learn: Machine learning in python. Journal of machine learning research, pages 2825–2830, 2011.
37. Recht Benjamin, Roelofs Rebecca, Schmidt Ludwig, and Shankar Vaishaal. Do ImageNet classifiers generalize to ImageNet? In Int. Conference on Machine Learning (ICML), pages 5389–5400, 2019.
38. Regan Elizabeth A, Hokanson John E, Murphy James R, Make Barry, Lynch David A, Beaty Terri H, Curran-Everett Douglas, Silverman Edwin K, and Crapo James D. Genetic epidemiology of COPD (COPDGene) study design. COPD: Journal of Chronic Obstructive Pulmonary Disease, 7(1):32–43, 2011.
39. Robinson Joshua, Jegelka Stefanie, and Sra Suvrit. Strength from weakness: Fast learning using weak supervision. In Int. Conference on Machine Learning (ICML), pages 8127–8136, 2020.
40. Robinson Joshua, Chuang Ching-Yao, Sra Suvrit, and Jegelka Stefanie. Contrastive learning with hard negative samples. In Int. Conf. on Learning Representations (ICLR), 2021.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

41. Soudry Daniel, Hoffer Elad, Nacson Mor Shpigel, Gunasekar Suriya, and Srebro Nathan. The implicit bias of gradient descent on separable data. The Journal of Machine Learning Research, 19(1):2822–2878, 2018.
42. Sun Li, Yu Ke, and Batmanghelich Kayhan. Context matters: Graph-based self-supervised representation learning for medical images. In Association for the Advancement of Artificial Intelligence (AAAI), 2021.
43. Szegedy Christian, Zaremba Wojciech, Sutskever Ilya, Bruna Joan, Erhan Dumitru, Goodfellow Ian, and Fergus Rob. Intriguing properties of neural networks. In Int. Conf. on Learning Representations (ICLR), 2014.
44. Tian Yonglong, Krishnan Dilip, and Isola Phillip. Contrastive multiview coding. In Europ. Conference on Computer Vision (ECCV), 2020.
45. Tian Yonglong, Sun Chen, Poole Ben, Krishnan Dilip, Schmid Cordelia, and Isola Phillip. What makes for good views for contrastive learning. In Advances in Neural Information Processing Systems (NeurIPS), pages 6827–6839, 2020.
46. Tsipras Dimitris, Santurkar Shibani, Engstrom Logan, Turner Alexander, and Madry Aleksander. Robustness may be at odds with accuracy. In Int. Conf. on Learning Representations (ICLR), 2018.
47. Wang Feng and Liu Huaping. Understanding the behaviour of contrastive loss. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
48. Wang Feng, Liu Huaping, Guo Di, and Sun Fuchun. Unsupervised representation learning by invariance propagation. In Advances in Neural Information Processing Systems (NeurIPS), pages 3510–3520, 2020.
49. Wang Tongzhou and Isola Phillip. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In Int. Conference on Machine Learning (ICML), pages 9574–9584, 2020.
50. Wang Xiao and Qi Guo-Jun. Contrastive learning with stronger augmentations. preprint arXiv:2104.07713, 2021.
51. Zbontar Jure, Jing Li, Misra Ishan, LeCun Yann, and Deny Stéphane. Barlow twins: Self-supervised learning via redundancy reduction. In Int. Conference on Machine Learning (ICML), 2021.
52. Zhang Hongyang, Yu Yaodong, Jiao Jiantao, Xing Eric, Ghaoui Laurent El, and Jordan Michael. Theoretically principled trade-off between robustness and accuracy. In Int. Conference on Machine Learning (ICML), pages 7472–7482, 2019.
53. Zhang Hongyi, Cisse Moustapha, Dauphin Yann N, and Lopez-Paz David. mixup: Beyond empirical risk minimization. In Int. Conf. on Learning Representations (ICLR), 2018.
54. Zhang Richard, Isola Phillip, and Efros Alexei A. Colorful image colorization. In Europ. Conference on Computer Vision (ECCV), pages 649–666, 2016.
55. Zhao Nanxuan, Wu Zhirong, Lau Rynson WH, and Lin Stephen. What makes instance discrimination good for transfer learning? In Int. Conf. on Learning Representations (ICLR), 2021.
56. Zimmermann Roland S, Sharma Yash, Schneider Steffen, Bethge Matthias, and Brendel Wieland. Contrastive learning inverts the data generating process. In Int. Conference on Machine Learning (ICML), 2021.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

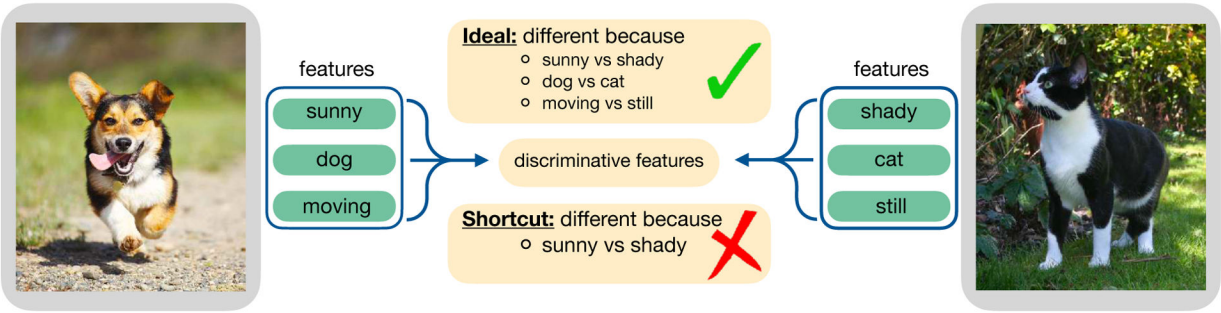



**Figure 1:**

An ideal encoder would discriminate 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).

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

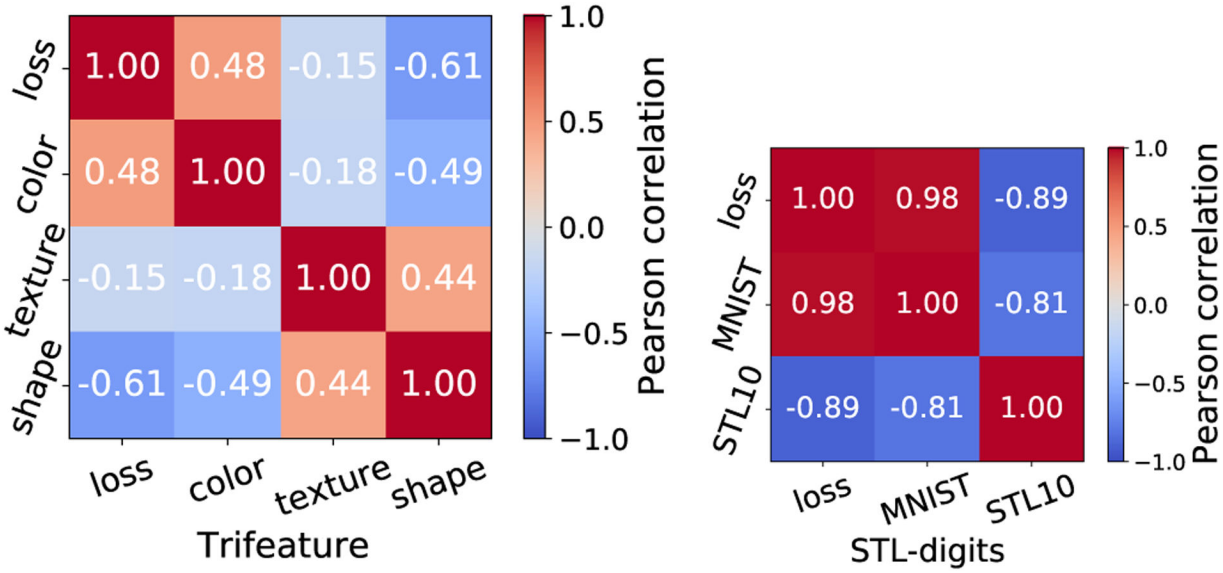



**Figure 2:**

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.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

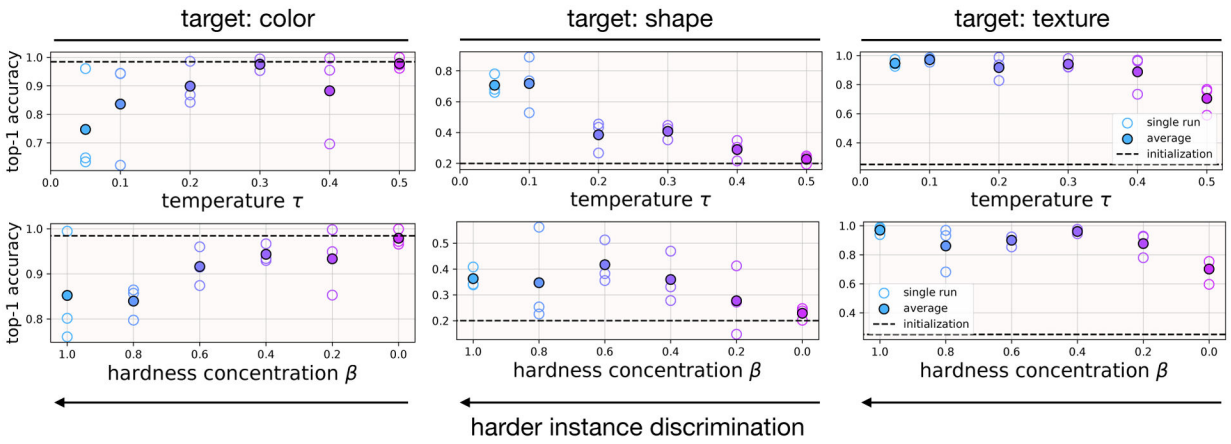



**Figure 3:**

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

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

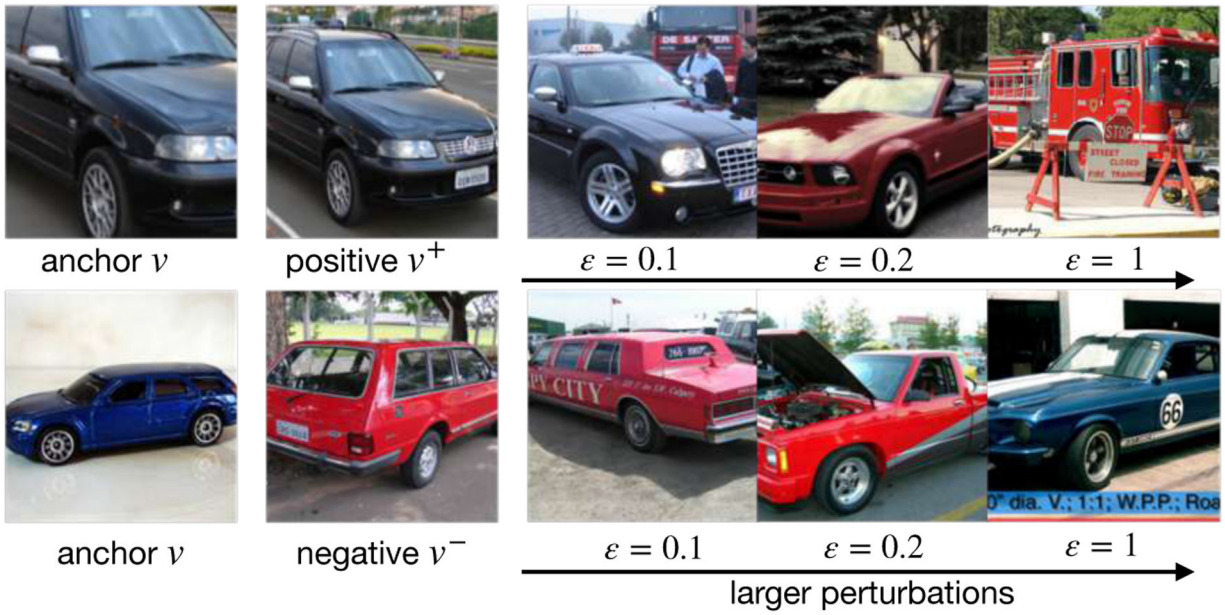



**Figure 4:**

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.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

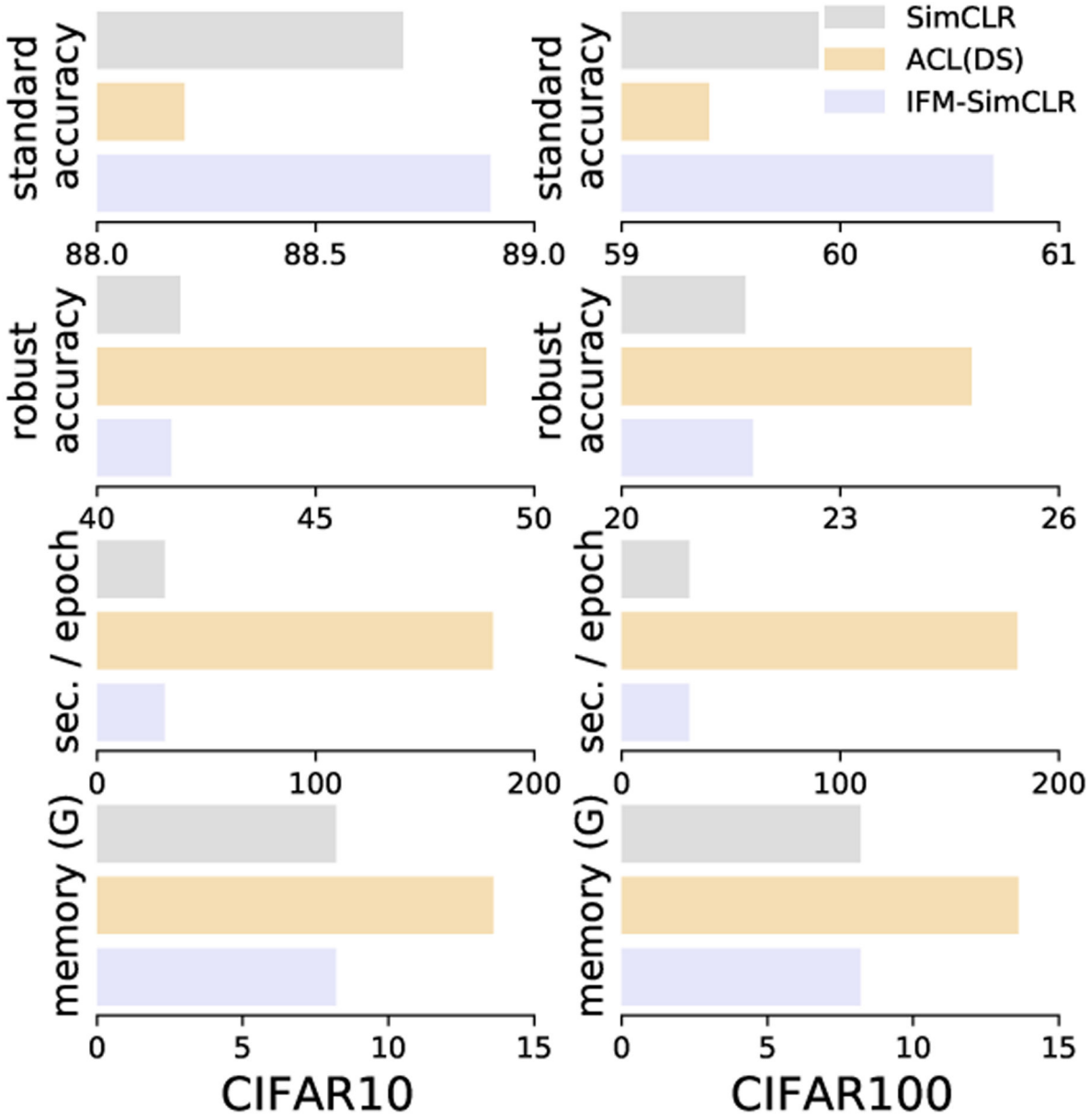



**Figure 5:**
Comparison between IFM and ACL(DS). Under standard linear evaluation IFM performs best. ACL is suited to adversarial evaluation.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

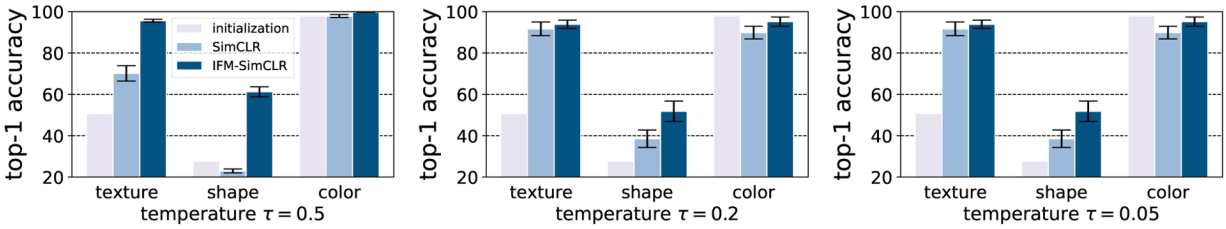



**Figure 6: Trifeature dataset.**

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 and use a fixed value $\varepsilon = 0.1$ to illustrate robustness to $\varepsilon$.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

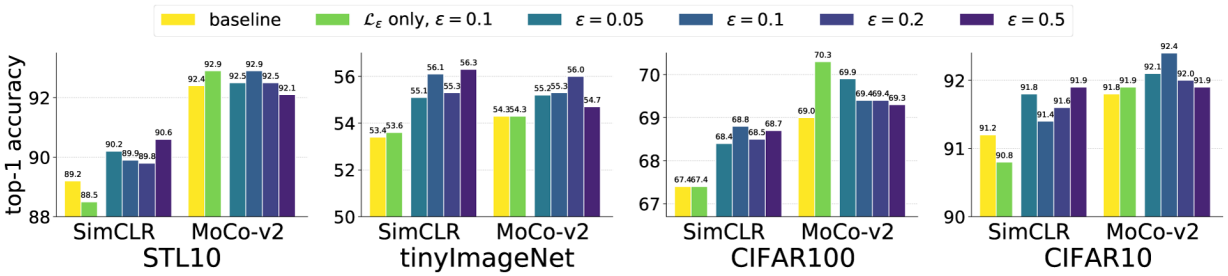



**Figure 7:**

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.

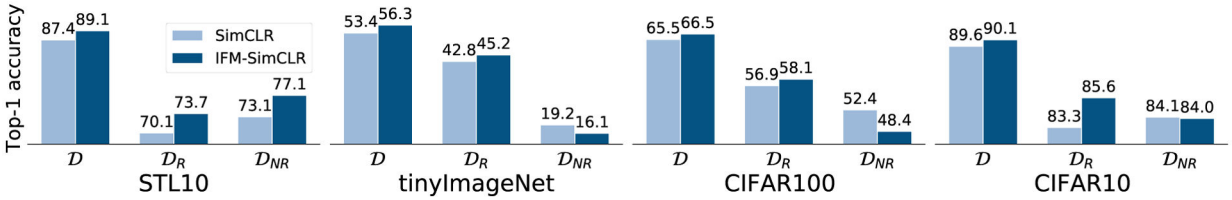



**Figure 8:**

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

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

**Table 1:**

Linear readout on ImageNet100. IFM improves over MoCo-v2 for all settings of $\epsilon$.

| - | MoCo-v2 | AdCo [20] | IFM-MoCo-v2 | | |
|---|---|---|---|---|---|
| $\epsilon$ | 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% |

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

**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.

| **Method** | **logFEV1pp** | **logFEV$_1$FVC** | **CLE** | **CLE *1-off*** | **Para-septal** | **Para-septal *1-off*** | **mMRC** | **mMRC *1-off*** |
|---|---|---|---|---|---|---|---|---|
| **Loss** | **R-Square** | | **Accuracy (%)** | | | | | |
| $\mathscr{L}$ (baseline) | $0.566_{\pm.005}$ | $0.661_{\pm.005}$ | $49.6_{\pm0.4}$ | $81.8_{\pm0.5}$ | $55.7_{\pm0.3}$ | $84.4_{\pm0.2}$ | $50.4_{\pm0.5}$ | $72.5_{\pm0.3}$ |
| $\mathscr{L}_\varepsilon$, $\varepsilon = 0.1$ | $0.591_{\pm.008}$ | $0.681_{\pm.008}$ | $49.4_{\pm0.4}$ | $81.9_{\pm0.3}$ | $55.6_{\pm0.3}$ | $85.1_{\pm0.2}$ | $50.3_{\pm0.8}$ | $72.7_{\pm0.4}$ |
| IFM, $\varepsilon = 0.1$ | **0.615 ±.005** | **0.691 ±.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, $\varepsilon = 0.5$ | $0.607_{\pm.006}$ | $0.683_{\pm.005}$ | $49.6_{\pm0.4}$ | $82.0_{\pm0.3}$ | $54.9_{\pm0.2}$ | $84.7_{\pm0.2}$ | **50.6 ±0.4** | **73.1 ±0.2** |
| IFM, $\varepsilon = 1.0$ | $0.583_{\pm.005}$ | $0.675_{\pm.006}$ | **50.0 ±0.5** | **82.9 ±0.4** | **56.3 ±0.6** | **85.7 ±0.2** | $50.3_{\pm0.6}$ | $71.9_{\pm0.3}$ |