Exploring Spurious Learning in Self-Supervised Representations

Anonymous Author(s)

Affiliation

Address

email

Abstract

Recent empirical studies have found inductive biases in supervised learning toward simple features that may be spuriously correlated with the label, resulting in suboptimal performance on minority subgroups. Despite the growing popularity of methods which learn representations from unlabeled data, it is unclear how potential spurious features may be manifested in the learnt representations. In this work, we explore whether recent Self-Supervised Learning (SSL) methods would produce representations which exhibit similar behaviors under spurious correlation. First, we show that classical approaches in combating spurious correlations, such as dataset re-sampling during SSL, do not consistently lead to invariant representation. Second, we find that spurious information is represented disproportionately heavily in the later layers of the encoder. Motivated by these findings, we propose a method to remove spurious information from these representations during pretraining, by pruning or re-initializing later layers of the encoder. We find that our method produces representations which outperform the baseline on three datasets, without the need for group or label information during SSL.

1 Introduction

Many real-world predictive tasks contain spurious correlations – features that are correlated with the label only for certain subsets of the data [1, 2, 3]. For instance, models trained to detect pneumonia [4] or COVID-19 [5], from chest X-rays across hospitals use a spurious underlying feature – information about the source hospital – as a shortcut for predicting the pathology, rather than invariant pulmonary characteristics which are the core or invariant features. In cases where spurious correlations are easier to learn than invariant correlations [6, 7], Empirical Risk Minimization (ERM) models have been shown to make predictions based on spurious correlations. These models have systematically poor performance for minority subgroups where such correlations do not hold [8]. Learning to mitigate spurious shortcuts is well explored in supervised learning with targeted solutions like importance weighting [9], re-sampling [10, 11], or approaches based on group distributionally robust optimization [12].

Spurious learning becomes more complex when such correlations appear in unlabeled data. Self-Supervised Learning (SSL) methods aim to learn representations from unlabeled datasets through solving an auxiliary pretext tasks [13]. For instance, recently proposed SSL algorithms learn representations [14, 15, 16, 17, 18, 19] by discriminating instances within the training set [20]. In this context, important underlying features of the data, e.g., different identifiers of the disease, should be captured in the representations for downstream tasks rather than a simple proxy. For instance, in a setting where we have unlabeled x-rays from multiple hospital sites, we may want to perform SSL initially on this unlabeled data, and use the learnt representations for a downstream task like infection prediction. During SSL the spurious feature (hospital) could be correlated with core feature (infection present in x-ray), and be more easily learnt for pre-text tasks. However, this spurious feature is not invariantly useful for determining infection status – our intended downstream task. In this setting, biases embedded in latent correlations propagate to downstream tasks where the
We use model transformation modules to create new views of training examples in the representation space. The introduced set of transformations removes the features learned in the final few layers, and final representations are invariant to such transformations. 

In this work we address the open question of how spurious correlations are reflected in the representations that SSL models learn. We consider a simple setting where we are interested in capturing one core feature in the representations, which is correlated with a spurious, simpler feature. Inspired by a crucial observations from supervised learning on overparameterized models, we target a method that addresses two potential issues. First, overparameterized models have an inductive bias towards learning the spurious feature and “memorizing” the minority examples, even after re-weighting or re-sampling. Second, memorization predominately occurs in the deeper layers where earlier layers of the network correctly classify the easier examples while the final layers memorize the difficult examples.

We propose model transformations or Late-TVG - a method that induces invariance to spurious feature in the representation space by eliminating the memorization of minority groups. We run experiments in different settings on four datasets, and find that we are able to maintain discriminative ability for downstream predictive tasks, without access to group or label information. Our work makes the following contributions:

- We find that known techniques for avoiding spurious correlations during supervised learning, such as re-weighting or re-sampling of the training set with group information, does not consistently improve representations learnt with SSL.
- We show that minority groups are predicted by features learnt in the final layers of SSL networks, and hypothesize that regularizing the final layers improves learning of the more complex, or core, features.
- We find that Late-TVG effectively improves worst-group performance in downstream tasks in four datasets by enforcing core feature learning.

2 Methods

We assume that data is generated from underlying latent feature space $Z = \{ z_{\text{core}}, z_{\text{spur}}, \ldots \}$, where $z_{\text{core}}$ and $z_{\text{spur}}$ are correlated for unlabeled data available for pre-text task, and $z_{\text{core}}$ determines labels $y$ for our downstream task of interest, while $z_{\text{spur}}$ determines the spurious attribute, which is easier to learn, and is not of interest of downstream tasks. Our goal, is to be able to predict $y$ from the learned representations in the downstream task where such correlations do not hold.

Motivated by improved SSL model invariance when trained with augmentations in image space, we propose a model transformation module that specifically targets augmentations that modify the spurious feature in representation space. We propose Late-layer Transformation-based View Generation - Late-TVG, which uses transformations to overcome spurious learning in SSL models and improve core feature representation.

2.1 Late-layer Transformation-based View Generation

Formally, we consider a model transformation module $U$, that transforms any given function $f_{\theta}$ parameterized by $\theta = \{ W_1, \ldots, W_n \}$ to $f_{\theta}'$. At each step, we draw a transformation $\phi_{M, \theta'}$ from $U$ to obtain transformed encoder $f_{\theta}'$ from $f_{\theta}$. Each model transformation can be defined with a mask $M \in \{0, 1\}^{|\theta|} = \{ M^1, \ldots, M^n \}$, where we re-parameterize the unmasked weights $(1 - M) \odot \theta'$. 

Figure 1: We use model transformation modules to create new views of training examples in the representation space. The introduced set of transformations removes the features learned in the final few layers, and final representations are invariant to such transformations.
We investigate the performance of SSL without model transformations (SSL\textsubscript{Base}).

**Transformations:** In our experiments, we consider two types of transformation modules as below:

- **Re-initialization of the final-layers (SSL\textsubscript{Reinit,L}):** Re-initializing the weights in layers deeper than \( L: \mathcal{U}_{\text{Reinit,L}} = \{ \phi_{M_L, \theta_{\text{Base}}} \mid \theta_{\text{Reinit}} \sim D_\theta \} \) where \( M_L \) is masking all weights before layer \( L \) or \( M_L = \{ M^l_L \mid M^l_L = 1 \} \), and \( D_\theta \) is the parameter initialization distribution.

- **Threshold Pruning (SSL\textsubscript{Prune,L,a}):** Magnitude pruning \( a \% \) of the weights in all layers deeper than \( L: \mathcal{U}_{\text{Prune,L,a}} = \{ \phi_{M_\theta, \theta_0} \mid \theta_0 = (0)^{\| \theta \|} \} \) where \( M_\theta, \alpha = \{ M^l_L \mid \text{Top}_\alpha(W_\theta) \mid l \in [n] \} \) and \( \text{Top}_\alpha(W_\theta)_{i,j} = \| (W_\theta(i,j) \text{ in top } a \% \) of \( \theta \) \)

To learn these representations, given two random augmentations \( t, t' \sim \mathcal{T} \) from the augmentation module \( \mathcal{T} \), two views \( x_1 = t(x) \) and \( x_2 = t'(x) \) are generated from an input image \( x \). At each step, given a feature encoder \( f \), and an augmentation module \( \mathcal{U} \), we obtain a transformed model \( \tilde{f} = \phi(f) \) where \( \phi \sim \mathcal{U} \). During training, one example \( x_1 \) and \( x_2 \) are respectively passed through the normal encoder \( v_1 = f(x_1) \) and the transformed encoder \( \tilde{v}_2 = \tilde{f}(x_2) \). Encoded feature \( \tilde{v}_2 \) is now a positive example that should be close to \( v_1 \) in the representation space (see Appendix D).

Similarly to Image Augmentation modules inducing representation space invariance based on visual similarities, View Generation modules encourage the encoder to be invariant to final layer transformations. We hypothesize that this enables learning more complex features in the earlier layers of the encoder.

### 3 Experiments

#### 3.1 Investigating the Extent of Spurious Learning in SSL

We design three experiments to establish the extent of spurious learning in SSL, how easily it can be removed by simple solutions, and the impact of using Late-TVG.

**Baseline Evaluation of Spurious Learning in SSL:** We first empirically evaluate learning of the core and spurious features on self-supervised representations. Furthermore, we compare features of a supervised model trained by ERM, to self-supervised representations.

**Spurious Feature Removal Effectiveness using Group Information:** In the next step, we examine whether classical approaches for combating spurious correlations, such as re-sampling training examples [10], are effective in removing spurious information during SSL. Assuming that group information is available, we train SimSiam on datasets re-sampled using the following strategies: (i) downsampling examples in majority groups to have the same number of examples in all groups, (ii) upsampling minority examples to have the same number of examples in all groups.

**Investigating Spurious Signals in Layer-Wise Feature Representations:** We also design experiments to verify that spurious features are easier to extract than core features in SSL, and are sufficient for the instance discrimination task. To do so, we evaluate the mutual information between feature representations across the layers of the trained encoder with (1) the labels \( I(Z; Y) \), and (2) the spurious attribute \( I(Z; G) \).

#### 3.1.1 Experimental Setup

We investigate the performance of Late-TVG on four commonly used datasets containing spurious correlations – CMNIST [25], MetaShift [26], Spurious CIFAR-10 [6], and Waterbirds [27] (See Appendix E for dataset descriptions). We train SimSiam [17] models with ResNet-18 backbones on these datasets which contain spurious correlations. Then, we evaluate the learned representations using a balanced dataset where the correlation does not hold. To create the downstream training dataset, we subsample majority groups [7, 10], to avoid the statistical and geometrical skews [6] when of the linear classifier on representations. In each case we report the average and worst-group accuracy of downstream logistic regression (LR) and k-nearest neighbors (KNN) classifiers. For completeness, we also report the mutual information between the representations and the label \( I(Z; Y) \) and group variables \( I(Z; G) \), as well as the alignment loss [28]. We compare our method against standard SSL without model transformations (SSL\textsubscript{Base}).

3
4 Results

SSL Suffers From Spurious Correlations. From Table 2 we find that across all datasets, SSL models exhibit gaps between worst-group and average accuracy when predicting the core feature, indicating that even when spurious correlation does not hold for downstream tasks, the learnt features are more predictive of the spurious feature in comparison to the core one. This is in contrast with supervised learning [29], where such models contain enough core information to perform well on all subgroups, and only needing a re-training of the final layer on a balanced validation set. [30].

Re-sampling During SSL is Not Useful. From Table 2 we observe that re-sampling during self-supervised training does not improve downstream worst-group accuracy. Given that the downstream linear model is trained on a down-sampled dataset where such correlations do not exist, this means that re-sampling during self-supervised training does not necessarily improve linear separability of representations with respect to the core feature, even in balanced datasets.

Layer-Wise Feature Representations. In Figure 2, we see that spurious features are disproportionately represented in later layers of the network, while invariant features are represented throughout the network. This confirms our hypothesis that later layers contain more spurious information during SSL, and motivates our proposed method.

Transformation-based Disentanglement in Final Layers Improves Worst-group Performance. As shown in Table 1, Late-TVG improves the worst-group accuracy of the linear classifier is improved by up to more than 10% on spurcifar10. However, it does not seem to improve the worst-group accuracy in metashift. We suspect that this is due to the spurious feature (outdoor vs. indoor) being more difficult to infer than the invariant feature, which violates the assumptions of our method.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>kNN (Target)</th>
<th>LR (Target)</th>
<th>LR (Group)</th>
<th>Other Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Worst Group</td>
<td>Average</td>
<td>Worst Group</td>
<td>Average</td>
</tr>
<tr>
<td>cmnist</td>
<td>SSL_base</td>
<td>41.91%</td>
<td>86.72%</td>
<td>37.61%</td>
<td>83.67%</td>
</tr>
<tr>
<td></td>
<td>SSL_Reinit, 15</td>
<td>33.90%</td>
<td>86.19%</td>
<td>42.37%</td>
<td>78.89%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.7, 19</td>
<td>37.04%</td>
<td>88.49%</td>
<td>50.00%</td>
<td>81.62%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.9, 18</td>
<td>32.20%</td>
<td>89.04%</td>
<td>44.07%</td>
<td>84.32%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.99, 19</td>
<td>35.19%</td>
<td>88.31%</td>
<td>44.07%</td>
<td>79.64%</td>
</tr>
<tr>
<td>metashift</td>
<td>SSL_base</td>
<td>28.82%</td>
<td>54.17%</td>
<td>27.68%</td>
<td>66.57%</td>
</tr>
<tr>
<td></td>
<td>SSL_Reinit, 18</td>
<td>21.38%</td>
<td>61.23%</td>
<td>18.64%</td>
<td>62.95%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.7, 17</td>
<td>27.59%</td>
<td>60.09%</td>
<td>18.64%</td>
<td>63.09%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.8, 17</td>
<td>24.83%</td>
<td>58.66%</td>
<td>20.34%</td>
<td>64.95%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.9, 18</td>
<td>25.52%</td>
<td>59.23%</td>
<td>18.64%</td>
<td>66.66%</td>
</tr>
<tr>
<td>spurcifar10</td>
<td>SSL_base</td>
<td>22.12%</td>
<td>57.05%</td>
<td>36.24%</td>
<td>60.36%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.3, 18</td>
<td>31.25%</td>
<td>61.60%</td>
<td>45.83%</td>
<td>66.86%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.5, 18</td>
<td>30.77%</td>
<td>58.44%</td>
<td>43.75%</td>
<td>66.23%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.7, 19</td>
<td>35.42%</td>
<td>58.06%</td>
<td>41.67%</td>
<td>65.80%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.9, 17</td>
<td>25.00%</td>
<td>59.23%</td>
<td>39.58%</td>
<td>58.59%</td>
</tr>
<tr>
<td>waterbirds</td>
<td>SSL_base</td>
<td>49.72%</td>
<td>53.40%</td>
<td>48.04%</td>
<td>56.18%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.3, 17</td>
<td>48.29%</td>
<td>50.59%</td>
<td>52.34%</td>
<td>55.59%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.5, 17</td>
<td>43.77%</td>
<td>54.99%</td>
<td>51.56%</td>
<td>62.10%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.7, 18</td>
<td>35.83%</td>
<td>60.23%</td>
<td>50.78%</td>
<td>60.10%</td>
</tr>
<tr>
<td></td>
<td>SSL_prune, 0.9, 17</td>
<td>47.54%</td>
<td>50.88%</td>
<td>52.18%</td>
<td>57.82%</td>
</tr>
</tbody>
</table>

Table 1: Top model transformations for each dataset; The learned representations in each case, we freeze the representations to evaluate the representations with: (i) average and worst-group of a 5-NN classifier (ii) Average and worst-group of a linear classifier (iii) Average accuracy of a linear classifier trained to infer the spurious feature (iv) Mutual information between representations and classes (v) Mutual information between representations and spurious feature (vi) Alignment loss which is indicator of how close examples within the same class are in the representation space. In many cases, the worst-group accuracy of linear classifier is improved; in other cases kNN accuracy has improvements;

5 Conclusion

We proposed a method - Late-TVG - that improves the worst-group downstream performance of SSL models on spuriously correlated data without having access to group or label information, and empirically validated its performance. Future work include benchmarking other SSL methods such as SimCLR [14], and adapting the method to other modalities such as natural language [11], and considering multiple core features in the underlying feature space.
References


Appendix

A Summary of Related Work

Spurious Correlation

Spurious correlations arise in supervised learning models in a variety of domains, from medical imaging [4, 5] to natural language processing [11, 31]. A variety of approaches have been proposed to learn classifiers which do not make use of spurious information. Methods like GroupDRO [12] and DFR [32] require group information during training, while methods like JTT [33], LfF [34], CVaR DRO [35], CnC [28] do not. However, all methods require group information for model selection.

Self-supervised Representation Learning

Self-supervised learning methods learn representations from large-scale unlabeled datasets where annotations are scarce. In vision applications, the pretext task is typically to maximize similarity between two augmented views of the same image [36]. This can be done in a contrastive fashion using the InfoNCE loss [37], as in SimCLR [14] and MoCo [38], or without the need for negative samples at all, as in BYOL [39], SwAV [18], SimSiam [40], and Barlow Twins [19].

Learning under Dataset Imbalance and Shortcuts

Self-supervised models have been found to be more robust to dataset imbalance [41, 42, 43, 41]. Prior work addressed shortcut learning in contrastive learning by adversarially modifying encoded features [44]. Other works in addressing group robustness or fairness in SSL, however, require group information or labels [45, 46, 47]. In the supervised setting, how subnetworks of a trained model can affect minority examples [48] or out-of-distribution generalization [49], and forgetting features via final-layer re-initialization [50], have also been studied.

For a more comprehensive summary of the background and related work, see Appendix E.

B Evaluating Spurious Learning in Self-supervised Learning

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SSL Training</th>
<th>kNN (Target)</th>
<th>LR (Target)</th>
<th>LR (Group)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Average</td>
<td>Worst</td>
<td>Average</td>
</tr>
<tr>
<td>waterbirds</td>
<td>Normal</td>
<td>53.4</td>
<td>49.7</td>
<td>56.2</td>
</tr>
<tr>
<td></td>
<td>Downsample</td>
<td>51.7</td>
<td>47.0</td>
<td>55.5</td>
</tr>
<tr>
<td></td>
<td>Upsample</td>
<td>66.3</td>
<td>9.5</td>
<td>62.3</td>
</tr>
<tr>
<td>metashift</td>
<td>Normal</td>
<td>54.2</td>
<td>28.8</td>
<td>66.6</td>
</tr>
<tr>
<td></td>
<td>Downsample</td>
<td>45.1</td>
<td>0.0</td>
<td>59.8</td>
</tr>
<tr>
<td></td>
<td>Upsample</td>
<td>56.5</td>
<td>32.9</td>
<td>65.8</td>
</tr>
<tr>
<td>spurcifar10</td>
<td>Normal</td>
<td>57.1</td>
<td>22.1</td>
<td>60.3</td>
</tr>
<tr>
<td></td>
<td>Downsample</td>
<td>36.0</td>
<td>19.6</td>
<td>48.8</td>
</tr>
<tr>
<td></td>
<td>Upsample</td>
<td>43.8</td>
<td>20.7</td>
<td>53.6</td>
</tr>
<tr>
<td>cmnist</td>
<td>Normal</td>
<td>86.7</td>
<td>42.0</td>
<td>83.7</td>
</tr>
<tr>
<td></td>
<td>Downsample</td>
<td>86.2</td>
<td>23.7</td>
<td>80.0</td>
</tr>
<tr>
<td></td>
<td>Upsample</td>
<td>86.9</td>
<td>39.4</td>
<td>80.0</td>
</tr>
</tbody>
</table>

Table 2: Effect of resampling on worst group accuracy: Accuracy of ResNet-18 encoders trained using SSL on four datasets, evaluated on a balanced test set. We vary the SSL training set used (Normal: original dataset, Downsample: downsampling majority groups, Upsample: upsampling minority groups). kNN and LR represent the k-NN classifier or Logistic regression model, trained for the downstream tasks based on self-supervised representations. Re-sampling does not necessarily improve worst-group accuracy in the downstream task.
Figure 2: Comparison of Mutual Information between features and labels (top) vs. spurious attribute and labels (bottom) across layers of a ResNet-18 model trained with SimSiam on coloured MNIST. We observe that $I(Z; G)$ decreases in the intermediate layers, and grows back in the final layers, indicating that the final representations rely on the spurious feature for the instance discrimination task in SSL.

D Experimental Setup

In our experiments, we use SimSiam in which the encoder $f$ aims to maximize the cosine similarity of the two views via predictor network $h$, and a stop-gradient operator as in Figure 1. Given previously defined features $v_1$ and $\hat{v}_2$, the cosine similarity $D(h(v_1), \text{stopgrad}(\hat{v}_2))$ will be maximized at each step.

**Experimental Setup:** We train SimSiam models with proposed transformed modules. For simplicity, we use one module with a fixed pruning percentage and layer threshold throughout training. At each epoch $t$, given a transformation from the fixed module is applied to encoder $f_t$ to generate the transformed model $\tilde{f}_t$. The model transformations are applied to the branch of SimSiam that a gradient-stop operation is applied to later. We use group information in the validation set and measure worst-group accuracy in order to choose layer threshold and pruning percentage hyperparameters (In Appendix ?? we show that even an uncurated set of model transformations enhance worst-group performance), and evaluate them similar to previous experiments.

E Background

E.1 Group Robustness

**Empirical risk minimization (ERM)** minimizes the average training loss across training points. Given a loss function $\ell(x, y; \theta)$, ERM minimizes the following objective:

$$J_{\text{ERM}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i; \theta). \tag{1}$$

**Group distributionally robust optimization (Group DRO)** uses training group information to minimize the worst-group error on the training set, assuming we have access to group annotations on the training data $\{(x_1, y_1, g_1), \ldots, (x_n, y_n, g_n)\}$. Given a loss function $\ell(x, y; \theta)$, the objective can then be written as:

$$J_{\text{groupDRO}}(\theta) = \max_{g \in \mathbb{G}} \frac{1}{n_g} \sum_{i | g_i = g} \ell(x_i, y_i; \theta) \tag{2}$$

where $n_g$ is the number of training points with group $g_i = g$.

**Just Train Twice (JTT)** is a simple two-stage approach that does not require group annotations at training time. First, it trains an identification model $\hat{f_{id}}$ via ERM and then identifies an error set
\[
E = \{(x_i, y_i) \text{ s.t. } \hat{f}_\text{id}(x_i) \neq y_i\} \text{ of training examples that } \hat{f}_\text{id} \text{ misclassifies. Then, it trains a final model } \hat{f}_\text{final} \text{ by upweighting the points in the identified error set.}
\]

\[
J_{\text{sup-ERM}}(\theta, E) = \left( \sum_{(x, y) \in E} \ell(x, y; \theta) + \sum_{(x, y) \notin E} \ell(x, y; \theta) \right),
\]

Correct-n-Contrast (CnC) learns an identification model similar to ITT to identify samples with the same class but dissimilar spurious features, and then trains a model with contrastive learning to learn similar representations for same-class samples. More precisely, it jointly trains the model’s encoder layers \( f_{\text{enc}} \) with a contrastive loss and the full model \( f_\theta \) with a cross-entropy loss with the following objective:

\[
\hat{L}(f_\theta; x, y) = \lambda \hat{L}_{\text{sup}}(f_{\text{enc}} : x, y) + (1 - \lambda) \hat{L}_{\text{cross}}(f_\theta; x, y).
\]

Where \( \hat{L}_{\text{sup}}(f_{\text{enc}} : x, y) \) is the supervised contrastive loss of \( x \) and its positive and negative samples, based on whether the identifier model has made a mistake on samples or no, and \( \hat{L}_{\text{cross}}(f_\theta; x, y) \) is average cross-entropy loss over \( x \), the \( M \) positives, and \( N \) negatives, and \( \lambda \) is a balancing hyperparameter.

E.2 Self-supervised Representation Learning

Self-supervised representation learning methods learn visual representations from large-scale unlabeled images where data annotations are scarce and time-consuming. Contrastive learning is a discriminative approach to learn representations that aims to attract similar or positive samples and push apart different or negative samples, which has become increasingly successful in recent years [14][15][16][18][19]. The standard approach for generating positive pairs without additional annotations is to create multiple views of each data point using random augmentations. The contrastive learning loss or InfoNCE [37] then maximizes a lower bound on the mutual information between the two views.

For instance, SimCLR [14] generates two randomly augmented views of each image \( \tilde{x}_i = t(x), \tilde{x}_j = t'(x), \ t, t' \sim \mathcal{T} \) given a batch of images, and uses all other augmented samples from the batch as negative examples. Then it uses an encoder \( f \) to extract representations from these augmented examples, and a small projection head \( g \) which maps these representations to the contrastive loss space. Given a minibatch of \( N \) samples, the InfoNCE loss is optimized for the sum of all examples in the minibatch.

Some proposed methods discard the need for negative samples in contrastive learning. BYOL [16] uses a siamese architecture with momentum encoders to prevent different representations from collapsing into one vector. SwAV [18] exploits online clustering for each batch to enforce consistency between cluster assignments from different views, and SimSiam [17] uses a simple stop-gradient operation in a siamese architecture to avoid collapsing.

We use SimSiam [17] in particular in our experiments. Similar to SimCLR it creates two randomly augmented views \( x_1 \) and \( x_2 \) from an image \( x \). Then it uses encoder \( f \) consisting of a backbone such as ResNet and a projection MLP head to create representations of the two views. A prediction MLP head \( h \) transforms the output of one view and matches it to the other view. Given two output vectors are \( 1 \triangleq h(f(x_1)) \) and \( 2 \triangleq h(f(x_2)) \); SimSiam minimizes the negative cosine similarity \((1, 2)\):

\[
(p_{1, 2}) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{2}{\|2\|_2},
\]

where \( \|\|_2 \) is \( \ell_2 \)-norm. Then they define a symmetrized loss for each image with a stop-gradient operator to avoid collapse as below:

\[
L = \frac{1}{2}(p_1, \text{stopgrad}(2)) + \frac{1}{2}(p_2, \text{stopgrad}(1)).
\]

Note that we use the word encoder to address the backbone in \( f \), since projection layers are thrown away when evaluating the representations.

E.3 Disentangled Representations

A similar line of research is creating representations where each dimension is independent and corresponds to a particular attribute [51][52], some works study learning such representations in a
supervised manner \cite{53, 54} while unsupervised approaches rely on VAEs \cite{55, 56, 57} and GANs \cite{58, 59}.

The works in fair representation learning usually address removing sensitive attributes from the representations by: obfuscating any information about sensitive attributes in order to approximately satisfy demographic parity \cite{60}, using adversarial methods \cite{61, 62, 63, 64, 65, 66, 67}, or feature disentanglement based using variational approaches. \cite{68, 69, 70, 71, 72, 73, 56, 74, 75}.

Perhaps the closest to our work is \cite{47} where samples are partitioned into two subsets that correspond to an entangled group element followed by minimizing a subset-invariant contrastive loss, where the invariance guarantees to disentangle the group element.

F Datasets

We make use of the following four image datasets:

- **waterbirds** \cite{12}: Background (land, water) is spuriously correlated with bird type (labdbird, waterbird).
- **cmnist** (Colored MNIST): Color of digit on the images spuriously correlated with the binary class based on the number inspired by \cite{25}, with no label slipping.
- **spurcifar10** (Spurious CIFAR10) \cite{6}: Color of lines on the images spuriously correlated with the class.
- **metashift** \cite{26}: Cats vs Dogs task: Background (indoor, outdoor) spuriously correlated with pet type (cat, dog).

G Comparing SSL to CLIP representations

We train linear classifiers with different re-sampled sets of training examples on frozen CLIP \cite{76} representations. These representations have found to be more robust to distribution shifts, and we aim to answer if balanced downstream training set can improve worst-group accuracy. As shown in table 3, even CLIP representations do not help mitigate the geometrical and statistical skews when learning the linear classifier on frozen representations.

<table>
<thead>
<tr>
<th>dataset</th>
<th>k-NN Average</th>
<th>k-NN Worst-group</th>
<th>Linear probe Average</th>
<th>Linear probe Worst-group</th>
<th>Spurious Attribute (Linear) Average</th>
<th>Spurious Attribute (Linear) Worst-group</th>
</tr>
</thead>
<tbody>
<tr>
<td>CelebA</td>
<td>83.51%</td>
<td>20.39%</td>
<td>89.67%</td>
<td>16.98%</td>
<td>77.09%</td>
<td>97.04%</td>
</tr>
<tr>
<td>Waterbirds</td>
<td>78.57%</td>
<td>75.56%</td>
<td>88.83%</td>
<td>81.11%</td>
<td>97.04%</td>
<td>97.04%</td>
</tr>
<tr>
<td>Metashift</td>
<td>90.89%</td>
<td>28.33%</td>
<td>91.76%</td>
<td>86.11%</td>
<td>98.57%</td>
<td>90.56%</td>
</tr>
<tr>
<td>Colored MNIST</td>
<td>56.77%</td>
<td>30.53%</td>
<td>61.30%</td>
<td>45.08%</td>
<td>68.06%</td>
<td>45.09%</td>
</tr>
<tr>
<td>Spurious CIFAR10</td>
<td>69.23%</td>
<td>63.81%</td>
<td>79.01%</td>
<td>74.01%</td>
<td>89.59%</td>
<td>86.30%</td>
</tr>
<tr>
<td>Colored MNIST</td>
<td>74.06%</td>
<td>43.93%</td>
<td>75.35%</td>
<td>55.61%</td>
<td>83.26%</td>
<td>68.85%</td>
</tr>
<tr>
<td>Spurious CIFAR10</td>
<td>74.44%</td>
<td>50.85%</td>
<td>76.06%</td>
<td>7.91%</td>
<td>63.19%</td>
<td>22.60%</td>
</tr>
<tr>
<td>Spurious CIFAR10</td>
<td>84.69%</td>
<td>68.97%</td>
<td>80.69%</td>
<td>30.51%</td>
<td>73.68%</td>
<td>22.03%</td>
</tr>
<tr>
<td>Spurious CIFAR10</td>
<td>88.41%</td>
<td>73.79%</td>
<td>82.55%</td>
<td>32.02%</td>
<td>75.54%</td>
<td>35.59%</td>
</tr>
<tr>
<td>Colored MNIST</td>
<td>73.59%</td>
<td>54.65%</td>
<td>72.64%</td>
<td>55.57%</td>
<td>74.14%</td>
<td>70.40%</td>
</tr>
<tr>
<td>Spurious CIFAR10</td>
<td>91.84%</td>
<td>89.39%</td>
<td>89.04%</td>
<td>85.98%</td>
<td>98.08%</td>
<td>96.21%</td>
</tr>
<tr>
<td>Colored MNIST</td>
<td>97.66%</td>
<td>62.88%</td>
<td>93.76%</td>
<td>88.26%</td>
<td>99.75%</td>
<td>97.16%</td>
</tr>
<tr>
<td>Spurious CIFAR10</td>
<td>35.11%</td>
<td>14.38%</td>
<td>55.29%</td>
<td>22.07%</td>
<td>93.52%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Spurious CIFAR10</td>
<td>33.20%</td>
<td>10.42%</td>
<td>52.56%</td>
<td>33.33%</td>
<td>51.29%</td>
<td>30.00%</td>
</tr>
<tr>
<td>Spurious CIFAR10</td>
<td>39.28%</td>
<td>20.83%</td>
<td>58.17%</td>
<td>41.67%</td>
<td>58.14%</td>
<td>35.42%</td>
</tr>
</tbody>
</table>

Table 3: Average and worst-group test accuracy of a logistic regression model trained on CLIP representations of original train, downsampled train, and upsampled train datasets. Balancing the training set used for linear evaluation helps us identify the learned representations by avoiding the statistical and geometrical skews induced by the linear evaluator.