A LIMITATIONS

In this work, we validated our method on several benchmark datasets containing spurious correlations from prior work (Sagawa et al., 2020a; Liang & Zou, 2022; Yang et al., 2023). However, we recognize that the scale of these datasets are small, relative to typical SSL training corpora (e.g. ImageNet (Deng et al., 2009)). As these large datasets do not contain annotations of spurious features, we are unable to evaluate our method in these settings. In addition, we primarily focus on SimSiam Chen & He (2020) in our experiments, as it does not rely on large batch sizes and shows improved performance for smaller datasets. Moreover, we expect LATETVG to perform best in cases where Siamese encoders coupled with stop-gradient operation are used when learning the representations.

B LATETVG ALGORITHM

We provide an algorithm representation of our proposed method LATETVG in Section 5.1 as follows.

Algorithm 1 Self-supervised Learning with LATETVG

Inputs: Encoder f parameterized by $\theta = \{W_1, \ldots, W_n\}$, Projection head and predictor g, Augmentation module $\mathcal{T}$, Threshold L, Pruning rate a, Training epochs N.
Initialize $\tilde{f}$ with θ

for all $i = 1 \to N$ **do**
  Stage 1: Self-supervised Training
  for all $i = 1 \to N$ **do**
    Draw two random augmentations $t, t' \sim \mathcal{T}$
    $x_1 = t(x), x_2 = t'(x)$ ▷ Generate views x_1, x_2 from input x using augmentation t
    $v_1 = f(x_1)$ ▷ Obtain encoded features from normal encoder f
    $\tilde{v}_2 = \tilde{f}(x_2)$ ▷ Obtain encoded features from transformed encoder $\tilde{f}$
    $\mathcal{L} = \text{Loss}(v_1, \tilde{v}_2; g)$ ▷ Calculate contrastive loss given views v_1 and $\tilde{v}_2$
    Update f, g to minimize $\mathcal{L}$ ▷ Update the encoder and other SSL parameters

  Stage 2: Model Transformation
    Compute the mask $M_{L,a} = \{M_L^l \odot \text{Top}_a(W_l) \mid l \in [n]\}$
      where $\text{Top}_a(W_l)_{i,j} = \mathbb{I}(|W_{l_{(i,j)}}| \text{ in top } a\% \text{ of } \theta)$
    Update $\tilde{f}$ with parameters $\tilde{\theta} = M_{L,a} \odot \theta$ ▷ Magnitude pruning of weights
Return encoder $\tilde{f}$

C THEORETICAL ANALYSIS OF SPURIOUS CONNECTIVITY

Setup We consider the pre-text task of learning representations from unlabeled population data $\mathcal{X}$ consisting of unknown groups $\mathcal{G}$ which are not equally represented. For a given downstream task with labeled samples, we assume that each $x \in \mathcal{X}$ belongs to one of $c = |\mathcal{Y}|$ classes, and let $y : \mathcal{X} \to [c]$ denote the ground-truth labeling function. Let us define $a : \mathcal{X} \to [m]$ as the deterministic attribute function creating groups (of potential different sizes) as $\mathcal{G} = \mathcal{Y} \times \mathcal{S}$.

Spectral Contrastive Learning In order to investigate why the invariant feature can be suppressed in contrastive learning, we consider the setting from HaoChen et al. (2021) – Spectral Contrastive learning, which achieves similar empirical results to other contrastive learning methods and is easier for theoretical analysis.

Given the set of all natural data or data without any augmentation $\overline{\mathcal{X}}$, we use $\mathcal{A}(\cdot|\bar{x})$ to denote the distribution of augmentations of $\bar{x} \in \overline{\mathcal{X}}$. For instance, when $\bar{x}$ represents an image, $\mathcal{A}(\cdot|\bar{x})$ can be the distribution of common augmentations that includes Gaussian blur, color distortion and random cropping.

Let $\mathcal{P}_{\overline{\mathcal{X}}}$ be the population distribution over $\overline{\mathcal{X}}$ from which we draw training data and test our final performance. For any two augmented data points $x, x' \in \mathcal{X}$, the weight between a pair $w_{xx'}$ is the marginal probability of generating the pair x and x' from a random data point $\bar{x} \sim \mathcal{P}_{\overline{\mathcal{X}}}$:

$$w_{xx'} = \mathbb{E}_{\bar{x} \sim \mathcal{P}_{\overline{\mathcal{X}}}} [\mathcal{A}(x|\bar{x})\mathcal{A}(x'|\bar{x})]$$

Define expansion between two sets similar to HaoChen et al. (2021) as below:

$$\phi(S_1, S_2) = \frac{\sum_{x \in S_1, x' \in S_2} w_{xx'}}{\sum_{x \in S_1} w_x}$$

where $w_x = \sum_{x' \in \mathcal{S}} w_{xx'}$. We note that this is similar to our definition of connectivity, where we have assumed the marginal distribution over x is uniform, or $w_x = \frac{1}{N}$.

Toy Setup Let the ground-truth labeling function y and the deterministic attribute function a, determine the subgroup $g = (y(x), a(x))$ of a given sample x. We suppose we have n samples from each subgroup, and that labels and attributes take binary values[1].

Suppose that each edge in the augmentation graph is given by connectivity terms α, β, ρ, γ as below:

$$\begin{aligned}\forall x, x' \in \mathcal{X}: \quad P_+(x, x') = &\mathbb{1}(a(x) = a(x'), y(x) = y(x'))\rho \\ &+ \mathbb{1}(a(x) \neq a(x'), y(x) = y(x'))\alpha \\ &+ \mathbb{1}(a(x) = a(x'), y(x) \neq y(x'))\beta \\ &+ \mathbb{1}(a(x) \neq a(x'), y(x) \neq y(x'))\gamma\end{aligned}$$

We suppose that each edge in the augmentation graph is deterministically equal to one of the connectivity terms, and make the following assumptions:

1. $\alpha > \gamma, \beta > \gamma$ – The probability that augmentation changes the spurious attribute only, or the class only is both greater than the probability that augmentation changes both attribute and class (at the same time).
2. $\rho > \alpha, \rho > \beta$ – The probability that augmentation that keeps both attribute and class is greater than the probability that it changes the spurious attribute only, or the class only is both higher than the probability that augmentation changes both domain and class (at the same time).
3. $\alpha > \beta$ or Assumption 3.2 – The probability that augmentation changes the spurious feature is higher than the probability of it changing the class, as observed in 4.

C.1 Proof of Lemma 3.3

Proof. Let the $A \in \mathbb{R}^{4n \times 4n}$ be the adjacency matrix of the simplified augmentation graph. It is easy to show that A is equivalent to adjacency matrix $\bar{A}$ up to a rotation where:

$$\begin{aligned}\bar{A} = &(\beta - \gamma) \cdot I_2 \otimes \left(\mathbf{1}_2 \mathbf{1}_2^\top\right) \otimes \left(\mathbf{1}_n \mathbf{1}_n^\top\right) \\ &+ (\alpha - \gamma) \cdot \left(\mathbf{1}_2 \mathbf{1}_2^\top\right) \otimes I_2 \otimes \left(\mathbf{1}_n \mathbf{1}_n^\top\right) \\ &+ (\rho - \beta - \alpha + \gamma) \cdot I_4 \otimes \left(\mathbf{1}_n \mathbf{1}_n^\top\right) \\ &+ \gamma \cdot \left(\mathbf{1}_4 \mathbf{1}_4^\top\right) \otimes \left(\mathbf{1}_n \mathbf{1}_n^\top\right)\end{aligned}$$

Where $\mathbf{1}_k$ is used to denote the all-one vector of dimension k and let $\bar{\mathbf{1}}_k$ be the normalized version.

For the case of $n = 1$, it is easy to show that the matrix is reduced to an adjacency matrix of 4 nodes, each in one group, where the first two rows/columns correspond to samples with the same spurious attribute, and odd or even rows correspond to samples that are from the same class, based on the placements of α and β in the matrix.

Let F be an embedding matrix with u_x on the x-th row which corresponds to the embeddings of sample x, and consider the matrix factorization based form of the spectral contrastive loss as below

$$\min_{F \in \mathbb{R}^{N \times k}} \mathcal{L}_{\text{mf}}(F) := \left\| \bar{A} - FF^\top \right\|_F^2$$

It is enough to compute the eigenvectors of $\bar{A}$, to obtain F. It is easy to compute the eigenvectors of $\bar{A}$ similar to Shen et al. (2022). The set of four sets of eigenvectors would be as below:

- For eigenvalue $\lambda_1 = \rho + \beta + \alpha + \gamma$, the eigenvector is $\bar{\mathbf{1}}_2 \otimes \bar{\mathbf{1}}_2 \otimes \bar{\mathbf{1}}_n$.

[1] For an ease of notation and operations

- For eigenvalue $\lambda_2 = \rho + \beta - \alpha - \gamma$ the eigenvectors are $[1 \ -1]^T \otimes \bar{\mathbf{1}}_2 \otimes \bar{\mathbf{1}}_n$.
- For eigenvalue $\lambda_3 = \rho - \beta + \alpha - \gamma$ the eigenvectors are $\bar{\mathbf{1}}_2 \otimes [1 \ -1]^T \otimes \bar{\mathbf{1}}_n$.
- $\lambda_4 = \rho - \beta - \alpha + \gamma$ which is smaller than the first three eigenvalues, given the above assumptions.

Thus F would be a rank-3 matrix with columns equal to $\sqrt{\lambda_i}$ multiplied by each eigenvector. Given the case of $n = 1$ explained above and by induction, it is easy to show that λ_2 corresponds to the spurious attribute subspace, and λ_3 corresponds to the class. Projecting samples in $\bar{A}$ with representations as rows of F, onto the spurious subspace suggests that the spurious feature takes two values $\{-\sqrt{\lambda_2}, \sqrt{\lambda_2}\}$, and similarly, the invariant feature takes two values $\{-\sqrt{\lambda_3}, \sqrt{\lambda_3}\}$ in the representation space learned by spectral contrastive loss. □

Intuitively, this means that with higher spurious connectivity —or higher weights on edges connecting images that only share the same spurious attribute— spectral clustering will learn representations of the population data based on the spurious feature, rather than the invariant feature.

D Data and Models

D.1 Datasets

We make use of the following four image datasets:

- `celebA` (Liu et al., 2015): Gender (Male, Female) is spuriously correlated with Hair color (blond hair, not blond hair).
- `waterbirds` (Sagawa et al., 2020a): Background (land, water) is spuriously correlated with bird type (landbird, waterbird).
- `cmnist` (Colored MNIST): The color of the digit on the images is spuriously correlated with the binary class based on the number. This is the same setup as Arjovsky et al. (2019), except with no label flipping.
- `spurcifar10` (Spurious CIFAR10) (Nagarajan et al., 2020): The color of lines on the images spuriously correlated with the class.
- `metashift` (Liang & Zou, 2022) We consider the Cats vs Dogs task where Background (indoor, outdoor) is spuriously correlated with pet type (cat, dog).

Note that each data contains both labels (or core attribute) y, and spurious attribute a. We then use the group information $g = (y, a)$ to partition dataset splits into groups.

D.2 Methods and Hyperparameters

We use SimSiam (Chen & He, 2020) with ResNet encoders to train both base models and LateTVG . We select ResNet-18 models as the backbone for all datasets except for `celebA`, which we use ResNet-50 models.

For each dataset, we use the following set of hyperparameters for SimSiam training.

Dataset	Learning Rate	Batch Size	Weight Decay	Number of Epochs
`celebA`	0.01	128	1e-4	400
`cmnist`	1e-3	128	1e-5	1000
`metashift`	0.05	256	0.001	400
`spurcifar10`	0.02	128	5e-4	800
`waterbirds`	0.01	64	1e-3	800

The specific augmentations that we used for learning the representations, are exactly similar to the SimSiam Chen & He (2020) paper but without color jitter.

Note that the model architecture and parameters for SSL-Base and SSL-Late-TVG are exactly the same, but SSL-Late-TVG uses the pruning hyperparameters to prune the encoder during training.

Computational Cost The SSL-LateTVG model updates the same number of parameters as SSL-Base during training, with the forward pass keeping both the original and pruned encoder. The pruning operation is cost O(n) where n is the number of parameters. So any FLOPs used for the extra pruning mechanism will be very small compared to a single forward pass.

D.3 The Role of Downstream Regularization

We investigate the impact of regularization techniques during downstream Linear probing. Interestingly, we find that the presence and type of regularization has a notable effect on the accuracy of the worst-performing group, with improvements of approximately 10% on the `celebA` dataset and 7% on the `metashift` dataset. We hypothesize that the minority samples contribute more to the variance of the linear models, and the additional regularization helps penalize them, leading to a reduction in the variance of the downstream models.

Table 5: Accuracy (%) of SimSiam models pretrained on each dataset with random initialization.

	Average			**Worst-Group**		
	None	L1	L2	None	L1	L2
`celebA`	78.5	81.9	**82.8**	66.1	**77.5**	76.7
`cmnist`	80.5	80.8	**82.7**	76.3	78.9	**81.3**
`metashift`	54.2	**59.8**	56.3	41.8	**48.1**	45.6
`spurcifar10`	69.3	72.5	**73.4**	41.1	45.1	**49.0**
`waterbirds`	**52.0**	51.2	50.5	47.4	**47.5**	47.2

E Measuring Spurious Connectivity in Augmentations

In this section, we present our methodology for measuring spurious connectivity in augmentations. We conduct experiments on four datasets, and our goal is to quantify the extent to which samples within the training set are connected to each other through the spurious attribute, as opposed to the core feature.

To estimate the average connectivity between two groups, denoted as g_1 and g_2, specified by class-attribute pairs (y, a) and (y', a'), we follow the algorithm outlined below:

Initially, we label all training examples belonging to group g_1 or class y and attribute a as 0, and all training examples belonging to group g_2. Next, we train a classifier to distinguish between the two groups. The error of this classifier would be a proxy for "the probability of augmented images being assigned to the other group", or how close they are in the augmentation space. Instead of training a large classifier from scratch for each pair, we use CLIP's representations in Section 4.2, and assume that it is extracting all necessary features for distinguishing between the two groups. In Section 5.2.3, we instead use the representations learned by each SSL model.

We train a linear model on these features to distinguish between each of the two groups. It is important to note that the augmentations used in our experiments are the classical augmentations commonly employed in SimSiam, excluding Gaussian blur. Subsequently, we create the test set following a similar process, where images are labeled based on their group or class-attribute pairs. The trained linear classifier is evaluated on this strongly augmented test set. The test error of the classifier serves as an estimate for the connectivity between the two pairs, providing insights into the degree of connectivity based on the spurious attribute.

By applying this methodology to all four datasets, we obtain results regarding the average spurious connectivity compared to the invariant connectivity. Table 4 summarizes the findings, revealing that, across all datasets, the average spurious connectivity is higher than the invariant connectivity. Furthermore, we validate that both these connectivity values are higher than the probability of simultaneously changing both the spurious attribute and the invariant attribute. These observations indicate that the samples within the training set are more likely to be connected to each other through the spurious attribute, rather than the core feature. This finding suggests a preference of the contrastive loss for alignment based on the spurious attribute rather than class alignment.

F ADDITIONAL RESULTS FOR LATETVG

F.1 LATETVG REDUCES BACKGROUND RELIANCE IN HARD IMAGENET

We evaluate LATETVG on the Hard ImageNet dataset (Moayeri et al., 2022), which consists of 15 challenging ImageNet classes where models rely heavily on spurious correlations. The authors provide spuriousness rankings that enable creating a balanced subset.

In our experiments, we train the SSL model on the full Hard ImageNet train split, and train the linear classifier on the spurious-balanced subset. This tests the model's ability to learn representations without exploiting spurious cues.

We then evaluate the downstream classifier on four different dataset splits as below:

- **None**: Original test split
- **Gray**: The object region is grayed out by replacing RGB values with the mean RGB value. This removes texture/color cues.
- **Gray BBox**: The object region is removed by replacing it with the mean RGB value of the surrounding bounding box region. This ablates shape cues.
- **Tile**: The object region is replaced by tiling the surrounding bounding box region. This also ablates shape cues.

A classifier relying on the spurious (i.e. non-object) features will achieve high performance in all evaluation splits. However, a classifier relying on the invariant features should perform decently on the original test split, but exhibit greatly reduced accuracy on the other splits. Thus, we desire high accuracy for the None split, and low accuracy for the other three splits.

Comparing the results to section 7 from (Moayeri et al., 2022), we find that the gap between None and other three splits is already large in SSL-base, and SSL-LateTVG is further decreasing the accuracy in the spurious datasets. This shows that the SSL-LateTVG encoder relies less on the spurious feature to predict the labels, which degrades the performance on splits that try remove the core feature.

We do not tune the hyperparameters in this experiment, but we find that for all sets of hyperparameters, SSL-LateTVG results in lower downstream accuracy on Gray, Gray BBox, and Tile splits as shown in Table 6.

Algorithm	Pruning threshold, percentage	**None** ↑	**Gray** ↓	**Gray BBox** ↓	**Tile** ↓
SSL-BASE	-	79.5	61.6	53.5	58.1
SSL-LATETVG	46, 0.5	78.0	59.5	51.1	52.1
	47, 0.5	76.7	59.1	49.6	54.4
	48, 0.8	73.9	56.1	48.0	51.3
	49, 0.8	68.4	50.7	42.4	44.7

Table 6: We train SimSiam models with a ResNet-50 backbone on unlabeled data from Hard ImageNet containing spurious correlation, we then train the downstream linear classifier on a balanced subset, and evaluate the downstream model on splits containing spurious features – LATETVG degrades the performance on these splits, without hyperparameter tuning

F.2 LATETVG CLOSES THE GAP TO SUPERVISED PRE-TRAINING

Self-supervised pretraining has shown a lot of promise in bridging the gap to supervised approaches in general representation learning. In this section, we explore whether this trend holds true for pre-training with data containing spurious correlations. To perform this analysis, we start with the same encoder model and vary only the pretraining strategy while fixing other aspects of the training, such hyperparameter selection and model selection.

We emphasize that this is an unfair comparison to begin with, since supervised pretraining requires labeled data whereas SSL does not, hence reducing the annotation budget drastically as shown in table 7. However, the goal of this experiment to understand to what extent do SSL models and specifically LATETVG , compare with ERM based supervised pretraining strategies.

Table 7: Accuracy (%) of SSL models pre-trained on each dataset versus features of a supervised model: Representations obtained from the supervised featurizer are more predictive of the core feature than SimCLR and SimSiam featurizers

	Average Accuracy			Worst-Group Accuracy		
	SimCLR	**SimSiam**	**Supervised**	**SimCLR**	**SimSiam**	**Supervised**
celebA	82.1	81.9	**91.9**	76.7	77.5	**81.7**
cmnist	82.5	82.1	**98.4**	81.7	80.7	**94.9**
metashift	55.1	55.8	**89.8**	45.5	42.3	**83.5**
spurcifar10	69.3	75.1	**89.9**	36.5	43.4	**79.6**
waterbirds	47.5	50.7	**67.9**	43.8	**48.3**	41.1

Table 8 shows the results of our experiment – we have compared both average and worst group accuracies for the SSL-based and ERM-based encoders across all our evaluation datasets. In terms of worst group accuracy it is clear that LATETVG narrows the gap between the SSL baseline and the ERM model significantly – 17% relative improvement for cmnist to 50% in the case of spurcifar10. In the case of celebA, we even outperform the ERM baseline. Similar to previous experiments, the relative boost in performance from LATETVG is higher for cases where the base encoder is weaker, indicating the strength of our final layer augmentation in extracting useful signal relevant to the core features during pretraining.

Table 8: LATETVG with SimSiam closes the gap between SSL baseline and supervised pre-training on worst group and average accuracy.

	Average Accuracy			Worst-group Accuracy		
	SSL-BASE	SSL-LATE-TVG	SUPERVISED	SSL-BASE	SSL-LATE-TVG	SUPERVISED
celebA	81.9	88.9	91.9	77.5	83.1	81.7
cmnist	82.1	80.6	98.4	80.7	83.1	94.9
metashift	55.8	70.1	89.8	42.3	79.6	83.5
spurcifar10	75.1	76.1	89.9	43.4	61.4	79.6
waterbirds	50.7	54.8	67.9	48.3	56.3	41.1

F.3 SSL-LATETVG OUTPERFORMS BASELINE ACROSS HYPER-PARAMETER SETTINGS

Disrupting the features and creating new views of the pairs is possible even with small amounts of pruning. We run a grid-search over the last three to five convolutional layers of ResNet models depending on the dataset, and choose pruning percentages varying between $[0.5, 0.7, 0.8, 0.9, 0.95]$. We find that in the metashift dataset, *all* hyperparameter settings improve the worst-group accuracy and outperform the baseline. Average and worst-group accuracies of different pruning hyperparameters on the metashift and celebA datasets is show in in figure 4.

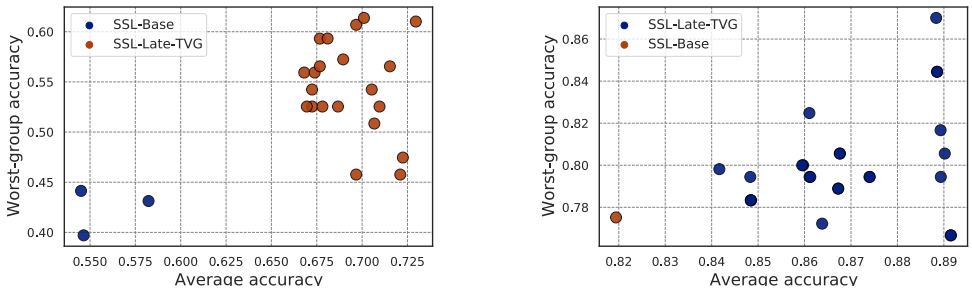

Figure 4: Downstream worst-group accuracy of SSL-Late-TVG on the metashift (left) and celebA (right) datasets as we vary the model pruning hyperparameters.

Additionally, instead of choosing the best-performing model, we consider top 5 models across different pruning hyperparameters, and report the performance in Table 9. Even in this scenario, we observe large performance gains with LATETVG .

Table 9: LATETVG improves baseline worst group and average accuracy of SSL models.

	celebA	cmnist	metashift	spurcifar10	waterbirds
Worst-group Accuracy					
SSL-BASE	77.52	**80.7±2.71**	42.33±2.32	43.44±8.87	48.3±1.82
SSL-LATE-TVG	**81.83±1.75**	77.18±1.59	**60.34±0.97**	**54.58±1.74**	**51.87±2.37**
Average Accuracy					
SSL-BASE	81.94	**82.08±1.17**	55.8±2.11	75.05±0.19	50.68±1.27
SSL-LATE-TVG	**87.32±1.46**	79.74±1.19	**69.7±2.09**	**75.68±0.72**	**55.36±0.72**

F.4 WHAT FEATURES DOES LATETVG LEARN?

Recall that we motivated LATETVG by explaining that more difficult features could be learned in the *later* layers of an encoder, and by removing the spurious feature from the encoder, we force the model to learn more complex features. In this section, we use Grad-CAM Selvaraju et al. (2016) to compare the SSL-base and SSL-LATETVG . We consider the representations that SSL-base and SSL-LATETVG learn for `metashift`, and use that to visualize the final layer of the encoder. We choose the best-performing LATETVG model based on downstream worst-group accuracy. We visualize the parts of the image that both SSL-Base and LATETVG attend to, in majority 5, and minority 6 groups.

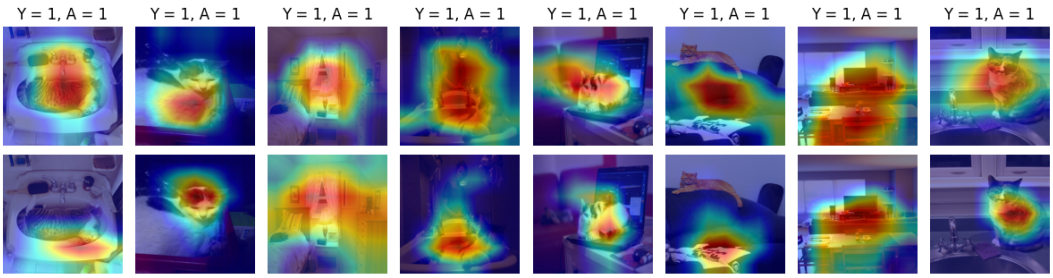

Figure 5: We use Grad-CAM to explain the ResNet-18 SSL-base (top), and SSL-LATETVG model (bottom) for majority examples

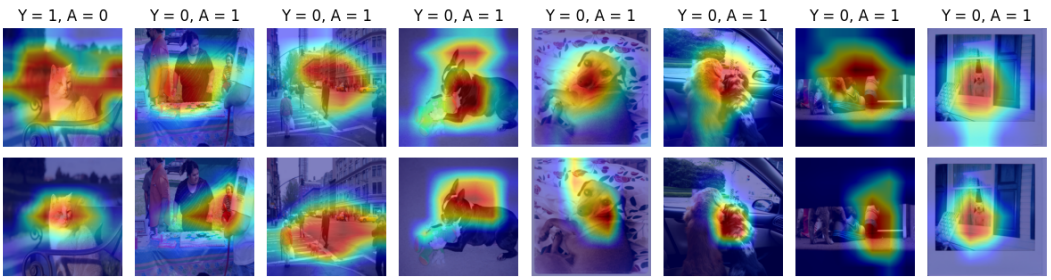

Figure 6: We use Grad-CAM to explain the ResNet-18 SSL-base (top), and SSL-LATETVG model (bottom) for minority examples

F.5 ADDITIONAL DOWNSTREAM IMBALANCE RESULTS

For both the best downstream linear model chosen based on worst-group accuracy, and linear models with no regularization, we observe the same trend for the datasets shown in Figure 7.

G SPURIOUS LEARNING IN SELF-SUPERVISED REPRESETATIONS

G.1 ADDITIONAL RE-SAMPLING RESULTS

We present the complete table from experiment in section 4.4.

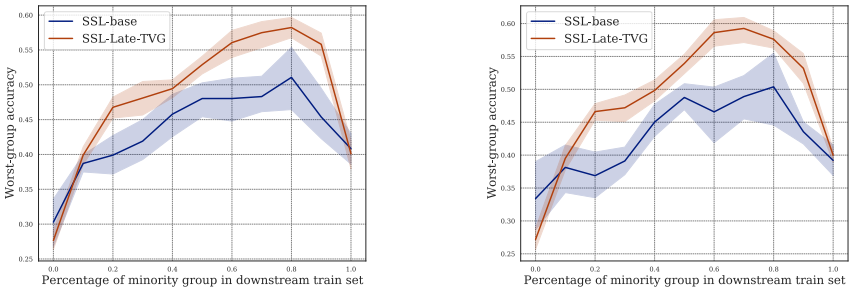

Figure 7: Effect of changing minority weight in downstream training set on `metashift`. Left (no regularization), Right (Downstream hyperparameters tuned)

Table 10: Downstream performance Accuracy (%) of linear models; For each dataset, we pre-train the model on up-sampled, down-sampled, and balanced training sets

Dataset	SSL Train Set	Average	Worst Group
celebA	Balanced	86.4	75.8
	Downsampled	83.2	77.8
	Original	81.9	77.5
	Upsampled	86.3	81.6
cmnist	Balanced	75.4	72.0
	Downsampled	74.7	70.1
	Original	82.1	80.7
	Upsampled	77.7	75.4
metashift	Balanced	60.7	38.5
	Downsampled	55.7	46.2
	Original	55.8	42.3
	Upsampled	64.4	45.1
spurcifar10	Balanced	68.7	35.2
	Downsampled	53.1	29.0
	Original	75.0	43.4
	Upsampled	57.4	24.1
waterbirds	Balanced	53.1	51.3
	Downsampled	51.0	48.8
	Original	50.7	48.3
	Upsampled	55.2	48.0

G.2 IMAGENET PRE-TRAINED SELF-SUPERVISED MODELS

We obtain pre-trained ResNet50 encoders with SimSiam, SimCLR, and CLIP training strategies, and evaluate the accuracy of core feature prediction similar to the previous sections.

Table 11: Worst-group Accuracy (%) of ImageNet pre-trained models when evaluated on each dataset using a linear probe.

Dataset	CLIP	ERM_{in}	$\text{SimCLR}_{\text{in}}$	$\text{SimSiam}_{\text{in}}$
celebA	87.2	79.4	87.2	84.4
cmnist	88.9	85.4	85.5	86.4
metashift	83.1	83.1	79.7	69.5
waterbirds	81.1	84.3	78.8	79.3