SelecMix: Debiased Learning by Contradicting-pair Sampling

Inwoo Hwang[1] **Sangjun Lee**[1] **Yunhyeok Kwak**[1] **Seong Joon Oh**[3]
Damien Teney[4] **Jin-Hwa Kim**[*12] **Byoung-Tak Zhang**[*1]
[1]AI Institute, Seoul National University [2]NAVER AI Lab
[3]University of Tübingen [4]Idiap Research Institute

Abstract

Neural networks trained with ERM (empirical risk minimization) sometimes learn unintended decision rules, in particular when their training data is biased, i.e., when training labels are strongly correlated with undesirable features. To prevent a network from learning such features, recent methods augment training data such that examples displaying spurious correlations (i.e., *bias-aligned* examples) become a minority, whereas the other, *bias-conflicting* examples become prevalent. However, these approaches are sometimes difficult to train and scale to real-world data because they rely on generative models or disentangled representations. We propose an alternative based on mixup, a popular augmentation that creates convex combinations of training examples. Our method, coined SelecMix, applies mixup to *contradicting pairs* of examples, defined as showing either (i) the same label but dissimilar biased features, or (ii) different labels but similar biased features. Identifying such pairs requires comparing examples with respect to unknown biased features. For this, we utilize an auxiliary contrastive model with the popular heuristic that biased features are learned preferentially during training. Experiments on standard benchmarks demonstrate the effectiveness of the method, in particular when label noise complicates the identification of bias-conflicting examples.

1 Introduction

The inductive biases contributing to the success of deep neural networks (DNNs) can sometimes limit their capabilities for out-of-distribution (OOD) generalization. DNNs are prone to learn simple, linear predictive patterns from their training data, sometimes ignoring more complex but important ones [30, 32]. It has been suggested that the simplest correlations in the data are often spurious [7]. A DNN relying on such simple, spurious patterns will therefore display poor OOD generalization. Spurious correlations in a dataset are often the result of a selection bias, and such datasets are therefore said to be *biased*. This paper is about debiased learning, also known as debiasing, i.e., methods that prevent a network from relying on spurious correlations when trained on a biased dataset.

Biased datasets typically contain a majority of so-called *bias-aligned* examples and a minority of *bias-conflicting* ones. In bias-aligned examples, ground truth labels are correlated with both robust and biased features.[2] In bias-conflicting examples, labels are correlated only with robust features. Clearly, the issues of models trained on biased datasets stem from the prevalence of bias-aligned samples. Various approaches for debiased learning encourage models to ignore biased features. Since

[*]Corresponding authors.

[2]A feature is biased if it displays a pattern that is statistically predictive of the labels over the dataset, though not necessarily on every example. For instance, a blue background may be present in most (but not all) images of birds. These images are said to be *bias-aligned*.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

the identification of biased features from i.i.d. data is ill-defined, it requires additional assumptions, or supervision from heterogeneous (non-i.i.d.) training samples [28].

In this work, we approach debiased learning with the assumption that biased features are "easier to learn" than robust ones, meaning that they are incorporated in the model earlier during training [29, 30] [3]. Existing works based on this heuristic typically train two models: (i) an auxiliary model that purposefully relies on biased features, and (ii) the desired debiased model. The auxiliary one guides the training of the debiased one [24, 27]. For example, Nam et al. [24] trains the auxiliary model with a *generalized cross-entropy* (GCE) loss [40] that strengthens its reliance on biased, easy-to-learn features. The training of the debiased model either augments the data with novel bias-conflicting examples [15, 20] or upweights existing ones [22, 24]. On the one hand, upweighting-based methods are simple but their debiasing capabilities are limited when bias-conflicting examples are scarce. On the other hand, augmentation-based methods rely on carefully tuned generative models or disentangled representations that are difficult to apply to real-world data.

We propose a simple and effective method based on mixup [37], a popular data augmentation method that creates convex combinations of randomly-chosen pairs of examples and their labels. Our method, coined SelecMix, is an application of mixup to selected *contradicting pairs* of examples to generate new bias-conflicting examples. We define contradicting pairs as having either (i) the same ground truth label but dissimilar biased features, or (ii) different labels but similar biased features. To compare examples with respect to their biased features and thereby identify the contradicting pairs, we train an auxiliary contrastive model with a novel *generalized supervised contrastive* (GSC) loss that amplifies the reliance on easy-to-learn features. As a result, feature clustering in the embedding space serves as a good indicator of the similarity of the biased features. We train the auxiliary model and the desired debiased model simultaneously. The auxiliary model identifies contradicting pairs, while the debiased model is trained on data augmented by SelecMix. Compared to past approaches, our method generates bias-conflicting examples without generative models or disentanglement, while implicitly upweighting existing ones since they are frequently selected for the mixup.

We evaluate our method on standard benchmarks for debiasing. Experimental results suggest that SelecMix consistently outperforms prior methods, especially when bias-conflicting samples are scarce. In addition, our method maintains its superior performance in the presence of label noise that complicates the identification of bias-conflicting examples. Ablation studies show advantages of both (i) the selective mixup strategy compared to other mixup variants, and (ii) the GSC loss, which strengthens the reliance on biased features and allows measuring the similarity of examples with respect to the biased features.

2 Related work

Debiasing with known forms of bias or bias labels. Early works on debiasing assume some knowledge about the bias. Some methods require that each training example is provided with a *bias label*, i.e., a precise value for its biased features [11, 14, 21, 26, 31]. Other methods use knowledge of the general form of the bias, such as color or texture in images. This information is typically used to design custom architectures [1, 5, 34]. For example, ReBias [1] uses a BagNet [4] as an auxiliary model because it focuses on texture, which is assumed to be the biased feature. The auxiliary model guides the training of a debiased model robust to unusual variations in texture.

Debiasing with the easy-to-learn heuristic. A number of recent works assume that biased features are learned more quickly than robust ones [20, 24]. A popular approach is to train an auxiliary model that intentionally relies primarily on the easy-to-learn biased features, e.g., with the GCE loss [40]. The auxiliary model guides the training of a debiased model that focuses on other, presumably non-biased features. For example, LfF [24] learns biased and debiased models simultaneously. Bias-conflicting training examples are identified based on the relative losses from the biased and debiased models, then upweighted for the training the debiased model. BiaSwap [15] trains an image translation model to generate new bias-conflicting training examples. Building on LfF, DFA [20] disentangles robust and biased features and swaps them randomly for augmentation. Disentanglement is however an ill-posed problem in itself [23] and a challenge with real-world data. In contrast, our

[3] See [38] for a discussion of the disputed relevance of this assumption to real-world data.

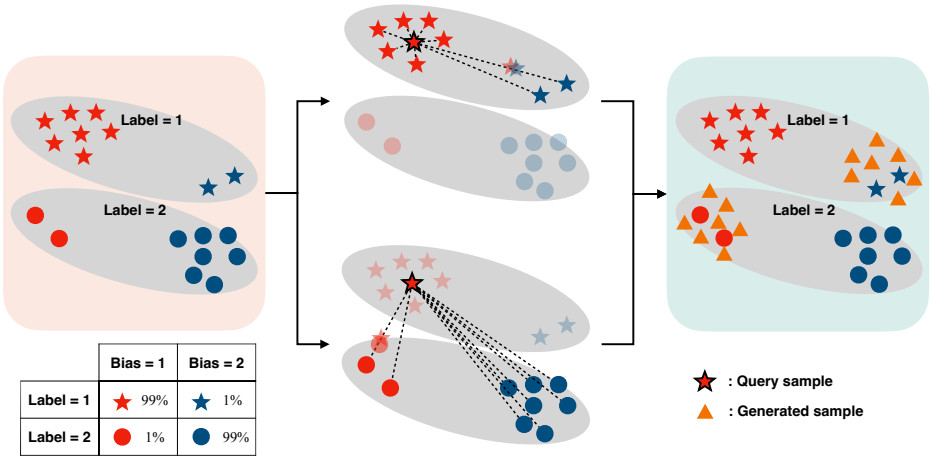


Figure 1: Overview of the proposed SelecMix. A gray ellipse with the symbols represents a sharing label on the embedding space. The selective mixup is applied to the pairs of samples (dashed lines) having **(top)** the same label but dissimilar biased features, and **(below)** the different labels but similar biased features (Sec. 3.3). The orange triangles of the top and below ellipses represent the generated samples by the previous two procedures, respectively. The legend illustrates the distribution of the exemplary dataset of two labels and two bias labels.

method augments bias-conflicting data by mixing existing examples, without generative models nor disentangled representations.

3 Method

We first describe the use of mixup as a debiasing strategy, while assuming that each training example is (unrealistically) provided with bias labels (Sec. 3.1). Then, we move to a realistic scenario where bias labels are not available. We introduce an auxiliary contrastive model that identifies the biased features, which are assumed to be "easier to learn" than robust ones. This allows comparing examples with respect to these biased features (Sec. 3.2). Finally, we describe the complete SelecMix method that combines the selective mixup with our auxiliary biased model (Sec. 3.3).

3.1 Mixup for augmenting bias-conflicting samples

In a highly biased dataset, bias-conflicting samples constitute only a small fraction of the training data. This is the root issue in the cases we consider. The goal is thus to increase the fraction of bias-conflicting examples in the training data, which will then reduce the reliance of the learned model on biased features. Our idea is to use mixup [37] to augment the existing pool of bias-conflicting examples. Mixup is a popular augmentation method [2, 16, 36] known to improve various measures of robustness [25, 39]. It constructs convex combinations of pairs of examples and their labels:

$$(\widetilde{x}_i,\, \widetilde{y}_i) \;\leftarrow\; \big(\lambda\, x_i + (1-\lambda)\, x_j,\;\; \lambda\, y_i + (1-\lambda)\, y_j\big), \tag{1}$$

where (x_i, y_i) and (x_j, y_j) are two original training examples (e.g., image and one-hot label vector) and λ is a random mixing coefficient, $\lambda \sim U[0, 1]$.

Assuming for now that bias labels are available, we could generate bias-conflicting examples by applying mixup on pairs having either (i) the same ground truth label but different bias labels or (ii) different labels but the same bias label. Any such pair includes at least one bias-conflicting example, so that mixup generates additional ones as long as this original example is assigned a higher mixing weight in Eq. (1). An overview of this mixup strategy is shown in Fig. 1. Next, we describe how to identify such desired pairs when bias labels are not available.

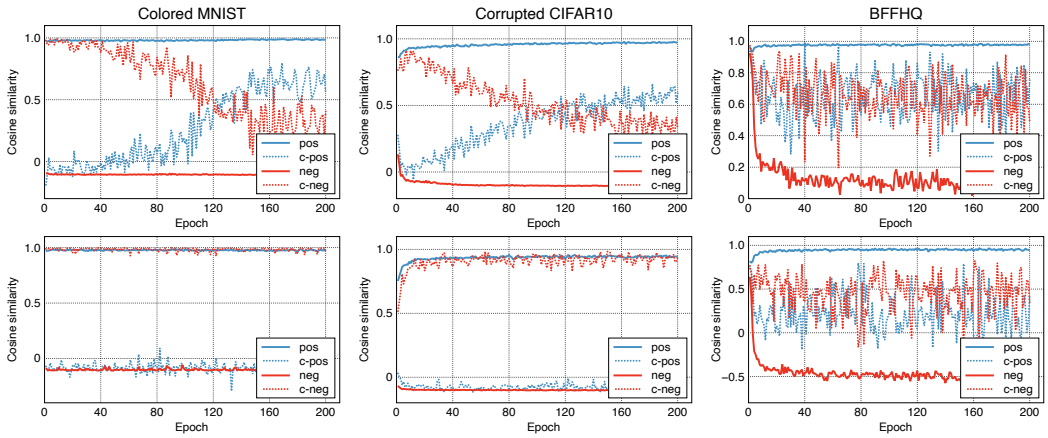


Figure 2: Illustration of the similarity of positive, negative, and contradicting pairs when trained with **(top)** the SC and **(bottom)** the proposed GSC losses. The solid line shows the average cosine similarity of (i) the pairs with the same label (*positives*) and (ii) the pairs with different labels (*negatives*). The dotted line represents (iii) the pairs with the same label but different bias labels (*contradicting positives*) and (iv) the pairs with the different labels but the same bias label (*contradicting negatives*). **Observations:** As training proceeds with the SC loss, the similarity of contradicting positives increases while it decreases for the contradicting negatives. In contrast, the proposed GSC loss amplifies the reliance on the biased features, thus the clustering in embedding space remains a good indicator of the similarity of the biased features during the whole training process.

3.2 Replacing bias labels with an auxiliary model

We utilize an auxiliary model to compare training examples with respect to their biased features. We assume that biased features are easier to learn than robust ones, because they are involved in simpler (e.g., linear) predictive patterns. Our auxiliary model is trained to rely primarily on biased features. We train the auxiliary model with a contrastive objective [6, 10, 13]. This is known to induce a clustering in embedding space (better than standard cross-entropy) that reflects the similarity of training examples in terms of the learned features. These are *biased* features by our assumption, such that the clustering reflects the similarity of examples w.r.t. their (unknown) bias labels. We use the supervised contrastive (SC) loss of Khosla et al. [13]:

$$\mathcal{L}_{SC} = -\sum_{i \in \mathcal{B}} \frac{1}{|\mathcal{P}_i|} \sum_{k \in \mathcal{P}_i} \log p_{i,k}, \quad \text{where} \quad p_{i,k} = \frac{\exp\left(\boldsymbol{z}_i^\intercal \boldsymbol{z}_k / \tau\right)}{\sum_{j \in \mathcal{B} \setminus \{i\}} \exp\left(\boldsymbol{z}_i^\intercal \boldsymbol{z}_j / \tau\right)}, \tag{2}$$

where $\boldsymbol{z}_i$ is the normalized embedding of image x_i, $\mathcal{B} = \{1, 2, ..., B\}$ is the set of indices in the current mini-batch, $\mathcal{P}_i = \{k \in \mathcal{B} \setminus \{i\} \mid y_i = y_k\}$ is the set of positive examples relative to the example i (i.e., with the same label), and the scalar τ is a temperature hyperparameter.

As an experiment to confirm that the clustering of training examples in embedding space is indeed based on biased features, we train the auxiliary model on the Colored MNIST, the Corrupted CIFAR-10, and the BFFHQ datasets (see Appendix A.1) with the SC loss and compute the cosine similarity $\boldsymbol{z}_i^\intercal \boldsymbol{z}_j$ of the embeddings of all pairs of examples. Fig. 2 shows that the examples are clustered according to the biased features early in the training. In other words, predictive patterns involving biased features are learned faster than those involving robust features, as desired. Since the higher cosine similarity $\boldsymbol{z}_i^\intercal \boldsymbol{z}_j$ implies a high probability $p_{i,j}$, we interpret it as the *likelihood* of the pair (i, j) having similar biased features.

To further amplify the reliance of the auxiliary model on the biased features, we define the *generalized SC* (GSC) loss as follows:

$$\mathcal{L}_{GSC} = -\sum_{i \in \mathcal{B}} \frac{1}{|\mathcal{P}_i|} \sum_{k \in \mathcal{P}_i} \hat{p}_{i,k}^{q} \log p_{i,k}, \tag{3}$$

Algorithm 1 SELECMIX

Input: batch $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{B}$, biased model g_ϕ
Output: batch $\widetilde{\mathcal{B}} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{B}$
Sample $p \sim \mathrm{U}[0,1]$, $\widetilde{\mathcal{B}} = \varnothing$
for $i = 1, \cdots, B$ **do**
  Sample $\lambda \sim \mathrm{U}[0,1]$ and $\lambda \leftarrow \min(\lambda, 1-\lambda)$
  if $p > 0.5$: $k = \operatorname*{argmin}_{j \in \mathcal{P}_i} g_\phi(x_i)^\intercal g_\phi(x_j)$
  else: $k = \operatorname*{argmax}_{j \in \mathcal{N}_i} g_\phi(x_i)^\intercal g_\phi(x_j)$
  $(\widetilde{x}_i, \widetilde{y}_i) \leftarrow (\lambda x_i + (1-\lambda) x_k, y_k)$
  $\widetilde{\mathcal{B}} \leftarrow \widetilde{\mathcal{B}} \cup \{(\widetilde{x}_i, \widetilde{y}_i)\}$
end for

Algorithm 2 Training with SELECMIX

Input: a dataset $\mathcal{D} = \{(x_i, y_i)\}$, a model f_θ, a biased model g_ϕ, the number of iterations T
Output: a debiased model f_θ
Initialize θ and ϕ
for $t = 1, \cdots, T$ **do**
  Draw a batch $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{B}$ from $\mathcal{D}$
  $\widetilde{\mathcal{B}} = \text{SELECMIX}(\mathcal{B}, g_\phi)$
  Update θ with $\mathcal{L}_{\text{CE}}(\widetilde{\mathcal{B}})$
  Update ϕ with $\mathcal{L}_{GSC}(\mathcal{B})$ {Eq. (3)}
end for

where $\hat{p}^q_{i,k}$ is a scalar having the same value as $p^q_{i,k}$, meaning that the gradient is not back-propagated through it. The term $\hat{p}^q_{i,k}$ assigns a higher weight to sample pairs with a high probability $p_{i,k}$ and thus amplifies the reliance on the biased features. We discuss the relationship between the GCE and GSC losses in Appendix C.2.

3.3 Complete proposed method: selective mixup with biased embedding space

We now have an auxiliary model for quantifying the similarity of the training examples in terms of the biased features. We use it to apply mixup on *contradicting pairs*, which have either (i) the same label but dissimilar biased features (*contradicting positives*) or (ii) different labels but similar biased features (*contradicting negatives*).

Contradicting positives. For each instance (x_i, y_i) in the current mini-batch (i.e., the "query"), we pick another one (x_k, y_k) with the lowest similarity (measured in the space of their embeddings produced by the auxiliary model) among the set of positive examples (i.e., with the same label as x_i):

$$k = \operatorname*{argmin}_{j \in \mathcal{P}_i} p_{i,j} = \operatorname*{argmin}_{j \in \mathcal{P}_i} \boldsymbol{z}_i^\intercal \boldsymbol{z}_j, \quad \text{where } \mathcal{P}_i = \{j \in \mathcal{B} \setminus \{i\} \mid y_i = y_j\}, \tag{4}$$

where $\mathcal{B} = \{1, 2, ..., B\}$ is the set of the sample indices in the current mini-batch, and $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ are the normalized embeddings produced by our auxiliary model g_ϕ, i.e., $\boldsymbol{z}_i = g_\phi(x_i)$. Since we select the pair among the set of positives, the training CE loss of the mixed example is $l(\widetilde{x}_i, \widetilde{y}_i) = l(\lambda x_i + (1-\lambda)x_k, \lambda y_i + (1-\lambda)y_k) = l(\lambda x_i + (1-\lambda)x_k, y_k)$. Considering that most examples are bias-aligned in the training set, the query (x_i, y_i) is also likely to be bias-aligned. In addition, since the query x_i and the selected example x_k have the same label but dissimilar biased features, it is also likely that the biased features of x_k are not correlated with the label, i.e., the selected example (x_i, y_i) is likely to be bias-conflicting. Thus, to effectively generate an example that contradicts the prediction based on biased features, we sample $\lambda \sim U[0,1]$ and assign the smaller value among λ and $1-\lambda$ to x_i and the larger one to x_k.

Contradicting negatives. For each query (x_i, y_i), we pick another one with the highest similarity among the set of negative examples:

$$k = \operatorname*{argmax}_{j \in \mathcal{N}_i} p_{i,j} = \operatorname*{argmax}_{j \in \mathcal{N}_i} \boldsymbol{z}_i^\intercal \boldsymbol{z}_j, \quad \text{where } \mathcal{N}_i = \{j \in \mathcal{B} \mid y_i \neq y_j\}. \tag{5}$$

Similarly, we sample $\lambda \sim U[0,1]$ and let $\lambda \leftarrow \min(\lambda, 1-\lambda)$. In standard mixup, the training loss of the mixed sample is: $l(\widetilde{x}_i, \widetilde{y}_i) = l(\lambda x_i + (1-\lambda)x_k, \lambda y_i + (1-\lambda)y_k) = \lambda \cdot l(\lambda x_i + (1-\lambda)x_k, y_i) + (1-\lambda) \cdot l(\lambda x_i + (1-\lambda)x_k, y_k)$. However, considering that (i) the query (x_i, y_i) is likely to be bias-aligned, and (ii) the pair (x_i, x_k) shares similar biased features, the first term $\lambda \cdot l(\lambda x_i + (1-\lambda)x_k, y_i)$ acts as bias-aligned example since the biased feature shared with x_i and x_k is predictive of the label y_i. Thus, rather than interpolating the label, we simply assign $\widetilde{y}_i \leftarrow y_k$. The pseudo-code of the proposed algorithm is presented in Alg. 1 and Alg. 2.

Intuitive interpretation. Unlike standard mixup, we assign the label of the generated sample to the label of the selected sample, which is likely to be bias-conflicting. Therefore, the proposed method

can be viewed as **generating new bias-conflicting samples by injecting bias-aligned samples as noise to existing bias-conflicting samples**. The method can also be interpreted as implicitly upweighting existing bias-conflicting samples, since they are frequently chosen for the mixup pair.

4 Experiments

We now validate the effectiveness of the proposed method on debiasing benchmarks. We also evaluate it under the presence of label noise, which is a challenging but realistic scenario, yet less explored so far (Sec. 4.1). We also provide a detailed analysis of each component of our method: (i) the auxiliary contrastive model (Sec. 4.2) and (ii) the selective mixup strategy (Sec. 4.3).

Datasets. The Colored MNIST is a modified MNIST [19] that consists of colored images of ten digits where each digit is correlated with the color (e.g., most of the images of "0" are colored with red). Here, the label is a digit (i.e., $0 \sim 9$) and the biased feature is color. The Corrupted CIFAR10 is constructed by applying different types of corruptions to the corresponding objects in the original CIFAR-10 [18] dataset (e.g., most of the images of dogs are corrupted with GAUSSIAN BLUR noise). The Biased FFHQ (BFFHQ) [20] is constructed based on the real-world dataset FFHQ [12], where the label is age and the biased feature is gender. The ratio of bias-conflicting samples in the training set is $\alpha \in \{0.5\%, 1\%, 2\%, 5\%\}$ for {Colored MNIST, Corrupted CIFAR10} and $\alpha = 0.5\%$ for BFFHQ. All datasets are available in the official repository of DFA [20]. We defer the detailed description of the datasets in Appendix A.1.

Baselines. To evaluate the effectiveness of our method in debiasing, we compare it with the prior methods LfF [24] and DFA [20] which also rely on the easy-to-learn property of the biased features. LfF trains an auxiliary model with the GCE loss to amplify its reliance on the biased features, then reweights examples for training a debiased model. DFA disentangles the biased and robust features with a similar principle as LfF, then augments the data for training a debiased model by swapping the biased features across examples. We also include EnD [31], ReBias [1], and HEX [34]. EnD leverages explicit bias labels. ReBias and Hex are designed for a specific, known form of biased features such as color and texture in images.

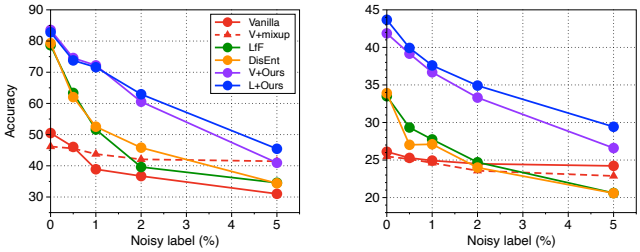
Table 1: Main results. (*) Denotes methods tailored to predefined forms of bias, ($^\circ$) methods using bias labels, and (†) methods relying on the easy-to-learn heuristic. Numbers for HEX are from [20].

Dataset	Ratio (%)	Vanilla	HEX * [34]	ReBias * [1]	EnD $^\circ$ [31]	LfF † [24]	DFA † [20]	V+Ours †	L+Ours †
Colored MNIST	0.5	35.71±0.83	30.33±0.76	**71.42**±1.41	56.98±4.85	63.86±2.81	67.37±1.61	70.47±1.66	70.00±0.52
	1.0	50.51±2.17	43.73±5.50	**86.50**±0.97	73.83±2.09	78.64±1.51	80.20±1.86	83.55±0.42	82.80±0.71
	2.0	65.40±1.63	56.85±2.58	**92.95**±0.21	82.28±1.08	84.95±1.71	85.61±0.76	87.03±0.58	87.16±0.62
	5.0	82.12±1.52	74.62±3.20	**96.92**±0.09	89.26±0.27	89.42±0.65	89.86±0.80	91.56±0.17	91.57±0.20
Corrupted CIFAR-10	0.5	23.26±0.29	13.87±0.06	22.13±0.23	22.54±0.65	29.36±0.18	30.04±0.66	38.14±0.15	**39.44**±0.22
	1.0	26.10±0.72	14.81±0.42	26.05±0.10	26.20±0.39	33.50±0.52	33.80±1.83	41.87±0.14	**43.68**±0.51
	2.0	31.04±0.44	15.20±0.54	32.00±0.81	32.99±0.33	40.65±1.23	42.10±1.04	47.70±1.35	**49.70**±0.54
	5.0	41.98±0.12	16.04±0.63	44.00±0.66	44.90±0.37	50.95±0.40	49.23±0.63	54.00±0.38	**57.03**±0.48
BFFHQ	0.5	56.20±0.35	52.83±0.90	56.80±1.56	56.53±0.61	65.60±1.40	61.60±1.97	**71.60**±1.91	70.80±2.95

4.1 Main results

We apply our method to a vanilla ResNet18 (V+Ours) and to LfF (L+Ours). We use the Colored MNIST, Corrupted CIFAR10, and BFFHQ datasets. As shown in Table 1, our method consistently outperforms baselines, except ReBias [1] on the Colored MNIST. Their bias-capturing model BagNet [4] is specifically tailored to color and texture as biased features by relying on local image patches as input. Both DFA and our method augment the pool of bias-conflicting training examples, but DFA's reliance on disentangled representations seems problematic on the more complex datasets. In contrast, our method performs well on all datasets, owing to the simplicity of the mixup strategy. In particular, our method outperforms DFA by a large margin on BFFHQ while the gap is smaller on Colored MNIST where disentanglement is easier. See Appendix A.2 for experimental details.

Debiasing under the presence of label noise. Label noise can negatively affect debiasing methods that need to identify bias-conflicting examples. This is because training examples with noisy (i.e., incorrect) labels are difficult to distinguish from the bias-conflicting examples that we wish to identify. Fig. 3 shows that our method maintains better performance under the presence of label noise than



(a) Colored MNIST (b) Corrupted CIFAR-10

Figure 3: Unbiased accuracy under the presence of label noise.

Table 2: Ablation study of various metrics on the biased embedding space. GT indicates that we used ground truth bias label for SelecMix. The cos and l_2 denote cosine distance and l_2 distance on the embedding space of the biased model trained with GCE loss, respectively. Ours denote the cosine distance on the embedding space of the proposed biased contrastive model.

SelecMix	GT	cos	l_2	KL-divergence	Ours
Contradicting positives	41.71±0.85	40.31±0.63	38.56±1.25	40.82±0.94	**42.94**±0.44
Contradicting negatives	41.70±0.74	33.69±0.97	34.15±0.50	30.99±0.93	**41.82**±0.98
Both	43.35±0.67	38.05±0.83	38.90±0.96	39.94±1.48	**43.68**±0.51

baselines. We hypothesize that the robustness of our method comes from the nature of mixup, which is known to improve robustness [39].

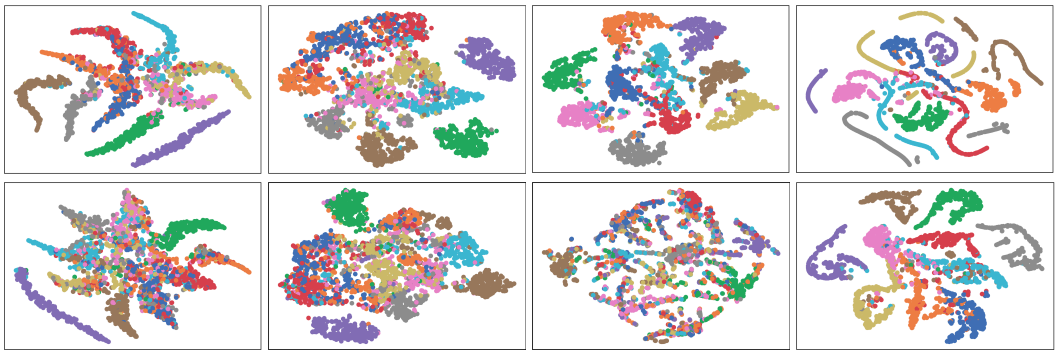
4.2 Detailed analysis of the biased contrastive model

Comparison of the biased embedding spaces for measuring bias similarity. Our auxiliary contrastive model trained with the GSC loss learns the embedding space which reflects the similarity of the examples w.r.t. their biased features. It is then used for SelecMix to identify the contradicting pairs by measuring the cosine similarity. We replace our auxiliary model with the biased model of LfF [24] which is trained with the GCE loss, and use it for SelecMix. For the similarity measures, we used cosine distance and l_2 distance in the embedding space learned by their biased model, and the KL-divergence of the softmax outputs of the classification head. As shown in Table 2, our auxiliary contrastive model achieves the best performance. Especially, the performance gap is significant for the contradicting negatives. This supports our claim in Sec. 3.2 that the proposed GSC loss induces the feature clustering in the embedding space w.r.t. the biased features. In contrast, the GCE loss learns the decision boundary and does not explicitly cluster the features, thus selecting the contradicting negatives, i.e., the pairs with different labels but similar biased features, seems to underperform ours.

Table 3: Bias label prediction accuracy.

Biased model	Corrupted CIFAR-10				BFFHQ
	0.5%	1.0%	2.0%	5.0%	0.5%
LfF [24]	77.44±0.94	73.01±1.70	67.00±0.87	55.58±0.06	51.07±3.06
DFA [20]	79.03±1.15	72.30±0.71	64.71±0.24	52.01±0.70	46.20±1.00
Ours	**95.45**±0.05	**93.39**±0.02	**92.89**±0.06	**87.28**±0.20	**59.13**±0.23

Bias label prediction of the biased model. Table 3 shows the accuracy of the bias label prediction of the auxiliary biased model for each method. Our biased contrastive model does not have a classification head since it is trained with a contrastive loss. Thus, we attach a linear classifier on top of the model and fine-tune it. Note that the bias labels are used only for the evaluation. As shown in Table 3, the biased contrastive model shows the best performance. As the ratio of the bias-conflicting samples increases, they degrade the performance. Notice our biased contrastive model is robust to this effect compared with the prior methods relying on the GCE loss.



(a) LfF [24] (b) DFA [20] (c) SC loss [13] (d) GSC loss

Figure 4: Visualization with t-SNE of features extracted from (a) the biased model of LfF [24], (b) the biased model of DFA [20], (c) the model trained with the SC loss, and (d) our auxiliary contrastive model trained with the GSC loss. **(Top)** α=1%. **(Bottom)** α=5%.

Visualization. Fig. 4 illustrates t-SNE [33] plot of the features extracted from the corresponding models trained on the Corrupted CIFAR10. As shown in Fig. 4(d), which corresponds to the proposed biased contrastive model, we observe that the samples are well clustered on the embedding space of the biased contrastive model. Similar to the results in Table 3, the GCE-based biased models, i.e., LfF and DFA, suffer from the increasing number of bias-conflicting samples.

Pretraining vs. simultaneous training of the auxiliary model. For our method in the main experiments, the auxiliary biased model is simultaneously trained with the debiased model. To analyze the performance of the pretrained biased model, we equip SelecMix with the pretrained biased contrastive model and evaluate the performance of our method with the varying total number of epochs for pretraining. The solid line represents the unbiased accuracy of our method with the pretrained biased model. The dotted line represents the unbiased accuracy of our method, i.e., simultaneous training of the auxiliary biased model. As shown in Fig. 5, vanilla SC loss exhibits the high variance of the performance with respect to the number of the epochs for pretraining, compared to the proposed GSC loss.

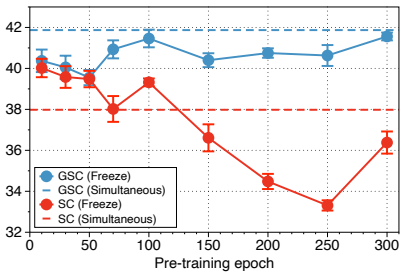


Figure 5: Comparison of simultaneous training vs. pretraining of the auxiliary model.

4.3 Detailed analysis of the selective mixup strategy of SelecMix

Ablation study. As shown in Table 4, we obtain the best performance when utilizing both the contradicting positives (A) and the contradicting negatives (B) all together in most cases. This is because it generates more diverse bias-conflicting examples compared to when using only one of them alone. On the other hand, standard mixup degrades the generalization capability in many cases which implies that the selection of the pairs is crucial.

Comparison with a mixup variant when bias label is available. We compare SelecMix with the recently proposed LISA [35], the mixup strategy for addressing domain generalization and subpopulation shift problems. LISA similarly applies the vanilla mixup on the pair of (i) the samples with the same label but different domains, and (ii) the samples with different labels but the same domain. Here, the domain label corresponds to the bias label in our experiment. The key differences between SelecMix and LISA are (i) ours does not require the bias label, (ii) we do not interpolate the labels of the generated samples, and (iii) we sample λ and let $\lambda \leftarrow \min(\lambda, 1-\lambda)$, i.e., we always assign the higher weight to the selected sample and the lower weight to the query sample (Sec. 3.3). Table 5 empirically confirms the better performance of SelecMix, using the explicit bias label for a fair comparison, over LISA, implying that augmenting the samples close to the bias-conflicting

Table 4: Detailed analysis of SelecMix. (A) denotes SelecMix with only contradicting positives. (B) denotes SelecMix with only contradicting negatives. (AB) corresponds to the proposed SelecMix.

Dataset	Colored MNIST				Corrupted CIFAR-10				BFFHQ
Ratio (%)	0.5	1.0	2.0	5.0	0.5	1.0	2.0	5.0	0.5
Vanilla	35.71±0.83	50.51±2.17	65.40±1.63	82.12±1.52	23.26±0.29	26.10±0.72	31.04±0.44	41.98±0.12	56.20±0.35
+ mixup	36.90±1.87	46.19±1.93	59.44±3.61	76.28±3.51	23.53±0.80	25.48±0.50	31.88±0.42	43.41±0.56	52.80±0.40
+ Ours (A)	58.49±0.77	74.59±0.77	83.16±1.14	89.32±0.71	37.16±0.09	41.38±0.41	**48.00**±0.64	**54.40**±0.26	66.20±1.80
+ Ours (B)	**71.23**±1.30	81.87±0.94	85.91±0.51	90.53±0.81	35.17±0.47	37.74±0.17	42.77±0.75	49.59±0.05	70.33±0.46
+ Ours (AB)	70.47±1.66	**83.55**±0.42	**87.03**±0.58	**91.56**±0.17	**38.14**±0.15	**41.87**±0.14	47.70±1.35	54.00±0.38	**71.60**±1.91
LfF	63.86±2.81	78.64±1.51	84.95±1.71	89.42±0.65	29.36±0.18	33.50±0.52	40.65±1.23	50.95±0.40	65.60±1.40
+ mixup	44.30±1.03	58.22±2.71	72.44±2.10	85.28±1.39	22.71±0.60	26.32±1.10	32.67±0.46	45.16±0.92	57.53±0.64
+ Ours (A)	57.28±1.96	74.44±0.36	84.20±0.48	90.33±0.22	38.46±0.40	42.94±0.44	49.32±0.28	56.11±0.83	67.60±2.20
+ Ours (B)	**70.26**±1.70	**83.14**±0.86	86.44±0.49	91.49±0.31	37.15±1.31	41.82±0.98	47.01±0.34	53.68±0.02	70.40±2.23
+ Ours (AB)	70.00±0.52	82.80±0.71	**87.16**±0.62	**91.57**±0.20	**39.44**±0.22	**43.68**±0.51	**49.70**±0.54	**57.03**±0.48	**70.80**±2.95

Table 5: Comparison with the mixup variants when the bias label is accessible.

Dataset	Colored MNIST				Corrupted CIFAR-10				BFFHQ
Ratio (%)	0.5	1.0	2.0	5.0	0.5	1.0	2.0	5.0	0.5
Vanilla	35.71±0.83	50.51±2.17	65.40±1.63	82.12±1.52	23.26±0.29	26.10±0.72	31.04±0.44	41.98±0.12	56.20±0.35
+ LISA (A)	48.95±0.75	67.10±0.77	78.28±0.99	87.04±0.72	33.29±0.65	38.62±0.31	45.79±0.13	53.41±0.27	63.20±0.92
+ Ours (A)	58.07±1.94	72.25±0.60	82.35±0.82	90.11±0.14	36.10±0.21	40.55±0.45	46.70±0.73	53.90±0.59	67.67±1.33
+ LISA (B)	52.32±2.30	72.97±1.22	79.74±2.25	86.44±1.10	29.56±0.93	34.23±1.34	40.47±0.67	47.61±0.64	58.93±0.50
+ Ours (B)	72.02±1.39	81.94±1.62	86.04±0.77	88.68±2.32	35.47±0.54	39.27±0.87	43.13±1.29	48.44±0.96	77.40±2.09
+ LISA (AB)	60.85±1.72	74.42±2.27	83.20±0.52	88.73±0.39	32.71±1.09	38.18±0.90	44.15±0.39	51.57±0.45	65.20±0.53
+ Ours (AB)	70.57±2.86	83.38±0.53	87.22±0.70	90.23±0.58	37.02±1.05	41.66±1.10	48.35±0.99	53.47±0.53	75.00±0.53

samples is crucial on the biased datasets. More discussions on the comparison between the SelecMix (using the explicit bias label) and LISA is provided in Appendix C.3.

5 Conclusions

We presented SelecMix, a method for debiased learning that augments bias-conflicting examples using mixup on contradicting pairs of examples. The selection of these pairs is the critical part of the method. It relies on an auxiliary model trained with a contrastive loss designed to amplify reliance on the biased features. The biased features are assumed to be "easy to learn" and incorporated earlier than others during training by SGD. SelecMix outperforms baselines on debiasing benchmarks and remains effective under the presence of label noise.

Limitations. The "easy-to-learn" assumption may not be correct, i.e., biased features are not always guaranteed to be learned faster than robust ones. Our method should only be used when this assumption holds, but it is unclear how to determine such cases a priori, and how frequent they are in real-world data [38]. In the worst case, our method could increase reliance on biased features and *worsen* robustness. Our experiments use semi-synthetic datasets where the assumption is made valid by construction. Thus, our results are not a sign of the broad real-world applicability of the method.

Societal impact. Our contributions should have a positive impact since our aim is to make machine learning systems more reliable. However, our method is only one step in this direction and the problems addressed should not be considered as solved.

Acknowledgments and Disclosure of Funding

We would like to thank Sangdoo Yun for the useful discussions and Yeonji Song for the suggestions on the writing. We also like to thank the anonymous reviewers for their constructive comments.

This work was supported by the SNU-NAVER Hyperscale AI Center and the Institute of Information & Communications Technology Planning & Evaluation (2015-0-00310-SW.StarLab/10%, 2019-0-01371-BabyMind/10%, 2021-0-02068-AIHub/10%, 2021-0-01343-GSAI/10%, 2022-0-00951-LBA/10%, 2022-0-00953-PICA/50%) grant funded by the Korean government.

References

[1] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *International Conference on Machine Learning (ICML)*, 2020.

[2] Duhyeon Bang, Kyungjune Baek, Jiwoo Kim, Yunho Jeon, Jin-Hwa Kim, Jiwon Kim, Jongwuk Lee, and Hyunjung Shim. Logit mixing training for more reliable and accurate prediction. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2812–2819, 2022.

[3] Yujia Bao, Shiyu Chang, and Regina Barzilay. Learning stable classifiers by transferring unstable features. In *International Conference on Machine Learning*, pages 1483–1507. PMLR, 2022.

[4] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. In *International Conference on Learning Representations*, 2018.

[5] Remi Cadene, Corentin Dancette, Matthieu Cord, Devi Parikh, et al. Rubi: Reducing unimodal biases for visual question answering. *Advances in neural information processing systems*, 32, 2019.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[7] Nikolay Dagaev, Brett D Roads, Xiaoliang Luo, Daniel N Barry, Kaustubh R Patil, and Bradley C Love. A too-good-to-be-true prior to reduce shortcut reliance. *arXiv preprint arXiv:2102.06406*, 2021.

[8] Giorgio Giannone, Serhii Havrylov, Jordan Massiah, Emine Yilmaz, and Yunlong Jiao. Just mix once: Mixing samples with implicit group distribution. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[11] Youngkyu Hong and Eunho Yang. Unbiased classification through bias-contrastive and bias-balanced learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[12] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[13] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[14] Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning not to learn: Training deep neural networks with biased data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9012–9020, 2019.

[15] Eungyeup Kim, Jihyeon Lee, and Jaegul Choo. Biaswap: Removing dataset bias with bias-tailored swapping augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14992–15001, 2021.

[16] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pages 5275–5285. PMLR, 2020.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[18] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[19] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

[20] Jungsoo Lee, Eungyeup Kim, Juyoung Lee, Jihyeon Lee, and Jaegul Choo. Learning debiased representation via disentangled feature augmentation. *Advances in Neural Information Processing Systems*, 34, 2021.

[21] Yi Li and Nuno Vasconcelos. Repair: Removing representation bias by dataset resampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9572–9581, 2019.

[22] Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pages 6781–6792. PMLR, 2021.

[23] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

[24] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33:20673–20684, 2020.

[25] Chanwoo Park, Sangdoo Yun, and Sanghyuk Chun. A unified analysis of mixed sample data augmentation: A loss function perspective. *arXiv preprint arXiv:2208.09913*, 2022.

[26] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2019.

[27] Victor Sanh, Thomas Wolf, Yonatan Belinkov, and Alexander M Rush. Learning from others' mistakes: Avoiding dataset biases without modeling them. In *International Conference on Learning Representations*, 2020.

[28] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[29] Luca Scimeca, Seong Joon Oh, Sanghyuk Chun, Michael Poli, and Sangdoo Yun. Which shortcut cues will DNNs choose? a study from the parameter-space perspective. In *International Conference on Learning Representations*, 2022.

[30] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.

[31] Enzo Tartaglione, Carlo Alberto Barbano, and Marco Grangetto. End: Entangling and disentangling deep representations for bias correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13508–13517, 2021.

[32] Damien Teney, Ehsan Abbasnejad, Simon Lucey, and Anton van den Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[33] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[34] Haohan Wang, Zexue He, Zachary C Lipton, and Eric P Xing. Learning robust representations by projecting superficial statistics out. In *International Conference on Learning Representations*, 2018.

[35] Huaxiu Yao, Yu Wang, Sai Li, Linjun Zhang, Weixin Liang, James Zou, and Chelsea Finn. Improving out-of-distribution robustness via selective augmentation. *arXiv preprint arXiv:2201.00299*, 2022.

[36] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[37] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[38] Jianyu Zhang, David Lopez-Paz, and Léon Bottou. Rich feature construction for the optimization-generalization dilemma. *arXiv preprint arXiv:2203.15516*, 2022.

[39] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? In *International Conference on Learning Representations*, 2020.

[40] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experimental details

A.1 A detailed description of datasets

For the main experiments, we evaluate our method on the Colored MNIST, Corrupted CIFAR-10, and BFFHQ. The example images of each datasets are shown in Figs. 6 to 8. The Corrupted CIFAR-10 is a modified version of the CIFAR-10 dataset [17] which is constructed by applying different types of corruptions. Specifically, each class is corrupted with one specific type of corruption from the following: BRIGHTNESS, CONTRAST, DEFOCUS BLUR, ELASTIC TRANSFORM, FROST, GAUSSIAN BLUR, GAUSSIAN NOISE, IMPULSE NOISE, PIXELATE, and SATURATE. For the Colored MNIST and Corrupted CIFAR-10, we report the accuracy on the unbiased test set. The BFFHQ [15] is a subset of the FFHQ dataset [12] constructed for evaluating debiasing methods. For the BFFHQ, we follow prior work [20] and report the accuracy on the test set consisting of bias-conflicting examples. The datasets are available in the official repository of DFA [20].

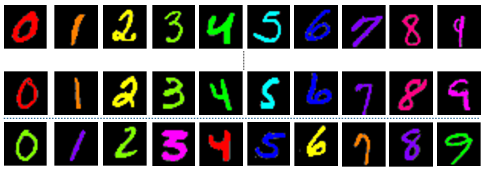


Figure 6: Example images of the Colored MNIST.

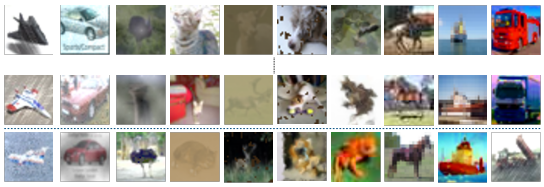


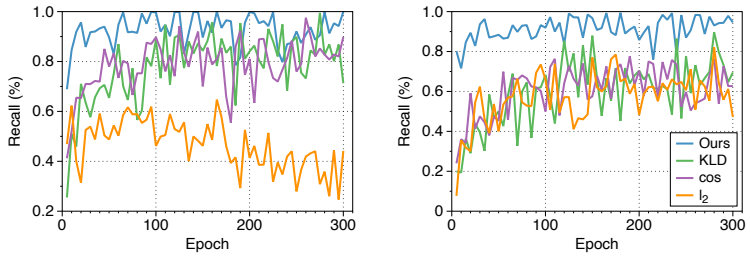
Figure 7: Example images of Corrupted CIFAR10.

Figure 8: Example images of BFFHQ.

A.2 Experimental details

Setup. We use a three-layer MLP for the Colored MNIST and a ResNet18 [9] for {Corrupted CIFAR-10, BFFHQ} as the backbone architecture for all methods. We set the batch size to 256 and 64 for {Colored MNIST, Corrupted CIFAR-10} and {BFFHQ}, respectively. We train the models for 200 and 300 epochs for {Colored MNIST, BFFHQ} and {Corrupted CIFAR-10}, respectively. To train our debiased model, we use the Adam optimizer with learning rates of 0.005, 0.001, and 0.0001 for the Colored MNIST, Corrupted CIFAR-10, and BFFHQ, respectively. To train our auxiliary contrastive model, we use the Adam optimizer with learning rates of 0.01 and 0.001 for the Colored MNIST and Corrupted CIFAR-10, respectively, and use the SGD with a learning rate of 0.4 for the BFFHQ. We did not use any additional augmentations to train the auxiliary model. Most of the experiments were run on a single RTX 3090 GPU.

Implementation details. In the experiments, we combine our method with the vanilla ResNet18 (V+Ours) and LfF (L+Ours). To train the debiased model f_θ, we update θ with $\mathcal{L}_{\text{total}} = \lambda_{\text{base}}\mathcal{L}_{\text{base}}(\mathcal{B}) + \lambda_{\text{ours}}\mathcal{L}_{\text{CE}}(\widetilde{\mathcal{B}})$, where $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{B}$ is the batch, $\widetilde{\mathcal{B}} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{B}$ is the batch of the mixup samples (See Alg. 1), and $\mathcal{L}_{\text{base}}$ is the update rule of the base algorithm for the debiased model (e.g., CE loss for the vanilla ResNet18). We used $(\lambda_{\text{base}}, \lambda_{\text{ours}}) = (0, 1)$ for V+Ours, which corresponds to line 7 in Alg. 2 (i.e., update θ with $\mathcal{L}_{\text{CE}}(\widetilde{\mathcal{B}})$) and $(\lambda_{\text{base}}, \lambda_{\text{ours}}) = (1, 0.1)$ for L+Ours. Note that the training of the auxiliary biased model is agnostic of the base algorithm (e.g., Vanilla or LfF). We fixed the temperature $\tau = 0.2$ (hyperparameter of the contrastive loss) and $q = 0.7$ (hyperparameter of the GCE loss and GSC loss) for all experiments. For the major hyperparameters (i.e., λ_{base}, λ_{ours}, and τ) of our method, we used the fixed values across the datasets and varying ratios. For the hyperparameters of the baselines, we used their default choice, even if the optimal values may vary for the different ratios in the same dataset. Since there is no provision for an unbiased validation set



(a) Contradicting positives (b) Contradicting negatives

Figure 9: Accuracy of discovering contradicting pairs. High recall implies that most of the contradicting pairs in the batch are discovered by SelecMix. The cos and l_2 denote respectively cosine and l_2 distance on the embedding space learned with the GCE loss. KLD denotes the KL-divergence of the softmax outputs of the prediction head trained with the GCE loss.

in most existing benchmarks, we follow the evaluation protocol of prior works [1, 20, 24] and report the best accuracy (i.e., an "oracle" model selection). Experimental results on the Corrupted CIFAR10 and the BFFHQ have averaged over 3 independent trials. For the Colored MNIST, we report the average test accuracy over 5 independent trials.

Label noise experiment. For the label noise experiments in Fig. 3, we modify the Colored MNIST ($\alpha = 1\%$) and the Corrupted CIFAR-10 ($\alpha = 1\%$) by replacing the label with a random one (i.e., label noise) for a portion $\beta \in \{0.5\%, 1\%, 2\%, 5\%\}$ of the training examples. We used the same hyperparameters for the label noise experiments and the main experiments.

B Additional experiments

B.1 Identifying contradicting pairs

We compare our biased model with the model trained with the GCE loss, in terms of the accuracy of identifying contradicting positives and negatives on the Corrupted CIFAR-10 dataset. As shown in Fig. 9, our biased contrastive model better identifies the contradicting pairs, especially for discovering contradicting negatives. This implies that the GSC loss effectively clusters the examples with respect to the biased features, which corroborates the results of Table 2.

B.2 Bias-conflict accuracy

In this subsection, we provide the bias-conflict accuracy (i.e., the accuracy of the bias-conflicting examples) of the baselines and our method. As shown in Table 6, our method outperforms the baselines in terms of the bias-conflict accuracy as well.

Table 6: Bias-conflict accuracy of the baselines and our method.

Dataset	Vanilla	Mixup [37]	EnD [31]	LfF [24]	DFA [20]	V+Ours	L+Ours
Colored MNIST (1.0%)	44.14±2.71	38.80±3.85	62.92±2.56	71.40±2.63	76.06±3.15	**77.38**±1.87	74.62±0.94
Colored MNIST (2.0%)	59.32±1.52	51.88±3.56	76.31±1.16	80.82±2.75	81.39±1.97	**83.32**±1.57	82.70±0.73
Corrupted CIFAR10 (1.0%)	16.30±0.50	14.77±0.50	16.22±0.53	24.08±0.97	25.00±2.55	35.89±1.05	**36.17**±1.25
Corrupted CIFAR10 (2.0%)	21.40±0.74	22.23±0.89	23.77±0.22	34.00±1.06	31.84±1.09	**44.24**±1.07	43.88±1.41

B.3 Comparison with JTT

In this subsection, we provide a comparison with JTT [22]. Table 7 shows the test accuracy of JTT and of our method. We used grid-search to choose the hyperparameters of JTT, i.e., the first stage training epoch T and the upsampling ratio λ as follows: $T \in \{1, 5, 10, 20, 30, 50\}$ and $\lambda \in \{5, 10, 30, 50, 100\}$.

Table 7: Comparison with JTT [22].

Dataset	Vanilla	JTT [22]	V+Ours
Colored MNIST (1.0%)	50.51±2.17	62.35±3.30	**83.55**±0.42
Colored MNIST (2.0%)	65.40±1.63	74.04±1.33	**87.03**±0.58
Corrupted CIFAR10 (1.0%)	26.10±0.72	28.55±0.27	**41.87**±0.14
Corrupted CIFAR10 (2.0%)	31.04±0.44	33.03±0.52	**47.70**±1.35
BFFHQ	56.20±0.35	58.40±0.35	**71.60**±1.91

B.4 Experiments on the UTKFace dataset

Dataset. The UTKFace dataset is composed of various human face images. The dataset contains labels for race, age, and gender. Fig. 10 shows the example images of the UTKFace dataset. Following the prior work [11], we divide the samples into two groups for each label and compose a biased sub-dataset. We defer the details to [11].

Figure 10: Example images of the UTKFace dataset.

Training details. For the debiased model, we use a ResNet18 pretrained on the ImageNet dataset as a backbone architecture and use the Adam optimizer with a weight decay of 0.0001. We set an initial learning rate of 0.001 when the label is either gender or race, and 0.0005 when the label is age. We use the mixup of the contradicting positives for SelecMix. We use $(\lambda_{\text{base}}, \lambda_{\text{ours}}) = (1, 0.1)$. The learning rate is decayed by a factor of 10 at 1/3 and 2/3 of the total training epochs, following the prior work [11]. For the auxiliary contrastive model, we use ResNet18 with the SGD optimizer. We train the models for 40 epochs with a batch size of 128. We use the data augmentations of random resized crop and random horizontal flip.

Experimental results. As shown in Table 8, our method consistently outperforms the baselines for various compositions of target and bias features. This implies that the proposed method successfully scales to the complex real-world datasets which the prior works struggles on.

Table 8: Unbiased accuracy of the baselines and our method trained on UTKFace dataset.

(Target, Bias)	Vanilla	LfF [24]	DFA [20]	V+Ours
(Gender, Age)	74.09±0.99	67.93±2.48	70.34±0.85	**77.00**±0.66
(Age, Gender)	92.00±0.35	84.90±1.72	84.23±1.66	**92.07**±0.10
(Race, Gender)	77.73±0.23	70.72±2.46	70.56±0.89	**78.58**±0.49
(Gender, Race)	88.68±0.21	77.79±1.43	82.03±2.36	**88.92**±0.11

C Additional discussions

C.1 Extended related work

JTT [22] proposed a 2-stage training framework to improve the group robustness without the group information. In the first stage, they train the vanilla network with ERM and construct the error set consists of the misclassified samples. In the second stage, they oversample examples of the error set to train the second network. JM1 [8] also constructs the error set and utilizes class-conditional mixup to further improve the group robustness. In the transfer learning setting, Bao et al. [3] learns spurious features from multiple environments and constructs the metric space based on the spurious features. In contrast, our method exploits the easy-to-learn heuristic and learns the hypersphere embedding space with the proposed GSC loss.

We now discuss the difference between JM1 [8] and our method, in particular, SelecMix (A). While class-conditional mixup is utilized in both our method and JM1, the key difference is in the sampling

of the pairs for the mixup. They combine the misclassified sample from the vanilla model with the sample that has the same label randomly chosen from the rest. On the other hand, SelecMix (A) combines the contradicting pair having the same label but *the most dissimilar* biased features, by measuring the similarity with the biased model.

C.2 Discussion on the relationship between GCE and GSC losses

In this subsection, we discuss the relationship between the generalized cross-entropy (GCE) loss [40] and the proposed GSC loss. To begin with, we first explain how the GCE loss amplifies the reliance of the auxiliary model on biased features, compared to the standard cross-entropy (CE) loss.

The GCE loss is defined as $\mathcal{L}_{\text{GCE}}(\boldsymbol{p}, y) = (1 - \boldsymbol{p}_y^q) \,/\, q$ where $\boldsymbol{p}$ is the softmaxed vector of predictions from the model and $\boldsymbol{p}_y$ its yth component, y is the ground truth class ID, and $q \in (0, 1]$ a scalar hyperparameter.

The GCE loss simplifies to the CE loss as $q \to 0$. Assuming that the predictions are produced by a model of parameters $\boldsymbol{\theta}$, the gradients of the GCE and CE losses are related as follows:

$$\frac{\partial}{\partial \theta} \mathcal{L}_{\text{GCE}_\theta}(\boldsymbol{p}, y) \;=\; \boldsymbol{p}_y^q \cdot \frac{\partial}{\partial \theta} \mathcal{L}_{\text{CE}_\theta}(\boldsymbol{p}, y). \tag{6}$$

Here, the term $\boldsymbol{p}_y^q$ assigns the higher weight to the samples with a high probability $\boldsymbol{p}_y$, thus upweights the "easy" samples and amplifies the reliance on the biased features. While the model trained with the CE loss also focuses on the biased features since they are learned first, the GCE loss was shown to be more effective in identifying bias-conflicting examples by Nam et al. [24]. Similar to the GCE/CE, our GSC loss improves over the SC loss [13] in encouraging the model to rely primarily on biased features. The term $\hat{p}_{i,k}^q$ in Eq. (3) plays the same role as the term $\boldsymbol{p}_y^q$ in the GCE.

C.3 A discussion comparing SelecMix with explicit bias labels and LISA

As described in Sec. 4.3, LISA [35] is a similar mixup strategy that applies mixup on contradicting positives and negatives. When the explicit bias label is available, SelecMix also directly picks the contradicting pairs and applies mixup. Here, the key difference is that (i) SelecMix does not mix the labels, and (ii) SelecMix always assigns the higher weight to the selected sample. Thus, SelecMix more explicitly augment various bias-conflicting samples, compared to LISA. As shown in Table 5, the performance gap between SelecMix and LISA is significant for the small α. This also implies that SelecMix is effective for debiased learning on the highly biased dataset.