Label Noise Robustness for Domain-Agnostic Fair Corrections via Nearest Neighbors Label Spreading

Nathan Stromberg
Arizona State University
nstrombe@asu.edu

Rohan Ayyagari
Arizona State University
rayyaga2@asu.edu

Sanmi Koyejo
Stanford University
sanmi@cs.stanford.edu

Richard Nock
Google Research
richardnock@google.com

Lalitha Sankar
Arizona State University
lsankar@asu.edu

Abstract

Last-layer retraining methods have emerged as an efficient framework for correcting existing base models. Within this framework, several methods have been proposed to deal with correcting models for subgroup fairness with and without group membership information. Importantly, prior work has demonstrated that many methods are susceptible to noisy labels. To this end, we propose a drop-in correction for label noise in last-layer retraining, and demonstrate that it achieves state-of-the-art worst-group accuracy for a broad range of symmetric label noise and across a wide variety of datasets exhibiting spurious correlations. Our proposed approach uses label spreading on a latent nearest neighbors graph and has minimal computational overhead compared to existing methods.

1 Introduction

In order to ensure fairness between subpopulations, an important metric to consider is the lowest accuracy amongst all subpopulations, often referred to as worst-group accuracy (WGA). State-of-the-art methods for optimizing WGA utilize group membership information to modify the training loss [1, 12, 21, 20, 22] or the distribution of the training data [10, 6, 11] in order to account for imbalance amongst groups and successfully train a classifier which is fair across groups.

While these methods can achieve astounding WGA, many have significant computational and data costs. Last-layer retraining (LLR) has emerged as a popular method to adapt existing deep models with minimal overhead while ensuring fairness [10, 11, 22, 20]. Additionally, the necessity of group membership information has been relaxed as new methods infer group membership by utilizing auxiliary classifiers to detect minority groups (generally known as two-stage methods) [11, 6, 12, 16, 22]. Taken together, it is now possible to correct powerful deep models for worst-group accuracy with low computational and data cost. Two methods using this approach are SELF [11] and RAD [22].

Unfortunately, class labels are often noisy [25] in practice. Further, Oh et al. [16] demonstrated just how fragile most two-stage model corrections are in the face of this issue. While Oh et al. [16] presented a method with improved robustness to label noise, their method requires fully retraining the base model and cannot match the performance of existing methods at low noise levels. To address these issues, we connect the existing literature on label propagation and worst-group accuracy correction, i.e., we utilize label spreading on the latent nearest neighbors graph to correct for noisy labels and two-stage LLR methods to maximize WGA without domain annotations. We list our main contributions below.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-4424

- We highlight the failure of state-of-the-art (SOTA) LLR methods, in particular, SELF [11] and RAD [22], under target label noise.
- We introduce an elegant label correction preprocessing method that significantly improves the performance of the SOTA LLR methods under label noise. The key insight here is that LLR models operate on largely separable embeddings, and therefore, label noise can be potentially corrected using label propagation techniques.
- For various spurious correlation datasets, we highlight the effectiveness of our modular approach when combined with both SELF and RAD and compare it to domain-aware and full retraining methods.

1.1 Related Work

Subgroup Robustness Downsampling has become one of the most common methods of increasing WGA. Kirichenko et al. [10] propose deep feature reweighting (DFR), which downsamples majority groups to the size of the smallest group to rebalance the retraining set. Chaudhuri et al. [4] explore the effects of downsampling theoretically and show that downsampling can increase WGA under certain data distribution assumptions. LaBonte et al. [11] propose using class-balancing when group membership information is unavailable and demonstrate its efficacy.

Upweighting is a popular alternative to downsampling as it utilizes all available data. Idrissi et al. [7] show that upweighting relative to the proportion of groups can achieve strong WGA, and Welfert et al. [26] prove downsampling and this form of upweighting are statistically equivalent. Upweighting has been extended to domain annotation-free settings by Qiu et al. [20] through loss-weighted learning.

Domain annotation-free methods often use a secondary model to identify minority groups. These are referred to as two-stage methods. Qiu et al. [20] use the pretrained model itself but do not explicitly identify minority examples and instead upweight proportionally to the loss. Unfortunately, this ties their identification method to the choice of the loss. Liu et al. [12] consider fully retraining the pretrained model as opposed to only the last layer, but their method of minority identification using an early stopped model is considered in the last layer in LaBonte et al. [11]. LaBonte et al. [11] not only consider early stopping as implicit regularization for their identification model, but also dropout (randomly dropping weights during training). Stromberg et al. [22] consider an explicit ℓ_1 regularizer, building on the work of LaBonte et al. [11] and Kirichenko et al. [10].

Wei et al. [25] show that human annotation of image class labels can be noisy with up to a 40% noise proportion, thus motivating the need for robust methods for WGA. Oh et al. [16] consider robustness to class label noise. Their method, END, utilizes predictive uncertainty from a robust identification model to select an unbiased retraining set. Their method struggles against SOTA methods at low noise levels and is not easily adapted to other error set-selection techniques. Still, it achieves strong performance in the high noise regime where other two-stage full retraining methods fall apart.

Label Propagation The problem of spreading labels from a small, but well-annotated set of reference points to a larger unlabeled set has been extensively explored in the literature. For a survey of basic methods of label propagation on a graph, see Bengio et al. [2] and references therein. The graph on which labels are propagated is critical to the success of these methods. Common graphs are the k-nearest neighbors (kNN) graph, the adjacency matrix of an undirected graph, or an underlying directed acyclic graph (DAG).

When labels are noisy, the objective of label propagation may be ill-equipped to correct these labels, but the general principles can be of use. For a restricted classifier setting, Gao et al. [5] prove the robustness of the kNN learning procedure to label noise and give a sense of how to choose k, the number of nearest neighbors, via an excess risk bound. This inspires our empirical analysis and our results verify the trends suggested by their bound. Iscen et al. [9] utilize neighbor consistency to deal with noisy labels when training deep models and frame their method as an implicit label propagation. Patrini et al. [19] consider label noise and possible corrections via loss and logit reweighting.

Label propagation and spreading have also been used in the semi-supervised learning setting for both general models and those focused on subgroup fairness. Iscen et al. [8] include label propagation in order to learn general deep models in a semi-supervised manner. Nam et al. [15] utilize an auxiliary model inspired by the label propagation objective to predict group labels on unlabeled samples. This

allows for semi-supervised fair training. Unfortunately, their method is very costly as it involves training both a spurious attribute prediction model and running GroupDRO [21].

Our work combines the ideas of two-stage last-layer corrections of LaBonte et al. [11], Stromberg et al. [22] with simple label propagation techniques in the latent space of the original (unfair) model. Because we have a low-dimensional embedding in which clean examples are well-separated by a linear classifier, we can utilize kNN label spreading with many nearest neighbors, which is enough to correct most label noise in the training data. With this preprocessing method, we can then proceed with any two-stage last-layer WGA correction to achieve fairness without domain information. Thus, our method is an elegant plug-and-play addition to last-layer subgroup robustness methods, which achieves SOTA test WGA when training with noisy class labels.

2 Problem Setup

Let $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^n$ be a dataset of n iid examples with features $X_i \in \mathcal{X}$ and class label $Y_i \in \mathcal{Y}$. Additionally let $\mathcal{G} = \{\mathcal{G}_i\}_{i=1}^k$ be a partition of $\mathcal{D}$, that is each $\mathcal{G}_i \subseteq \mathcal{D}$, corresponding to the k groups of interest. In general, this partition is unknown at training time.

We are given a pre-trained deep model

$$f(x;\theta) = \arg\max_{y\in\mathcal{Y}} \sigma_y(\langle \Phi(x), \theta\rangle), \tag{1}$$

where $\Phi : \mathcal{X} \to \mathbb{R}^d$ is an embedding function and $\sigma : \mathbb{R} \to [0,1]^{|\mathcal{Y}|}$ such that σ_y is the estimate of the probability that X is in class y. An example of σ is the oft-used softmax function. We assume that f is trained on clean data. While this may seem like a strong assumption, it can be relaxed as recent work suggests that feature embeddings are typically robust to label noise [9, 17].

We seek to retrain solely the last layer of f in order to increase the worst-group accuracy of the model. That is we learn a θ_{WGA} such that

$$\theta_{\text{WGA}} = \max_\theta \min_{\mathcal{G}_i\in\mathcal{G}} \frac{1}{|\mathcal{G}_i|} \sum_{(x,y)\in\mathcal{G}_i} \mathbb{1}(f(x;\theta) = y). \tag{2}$$

That is, θ_{WGA} maximizes the worst-group accuracy (WGA). We will use WGA as our metric for all of our discussions moving forward.

2.1 Label Noise Model

We assume that we observe not $\mathcal{D}$, but $\tilde{\mathcal{D}} = \{(X_i, \tilde{Y}_i)\}_{i=1}^n$ where Y has been affected by symmetric label noise with noise level $p \in (0, 1/2)$. In general, this noise could be caused by human error, data corruption, or malevolence. Patrini et al. [19] consider a wider class of class-dependent noises and show that in general, predictors can be made robust to such noise with relatively simple adjustments to the loss function or data distributions. This simple correction is not helpful in two-stage methods as demonstrated by Oh et al. [16]; we expand on this further in Section 2.3.

2.2 Basic Last-Layer Model Corrections for WGA

When we have information about the group membership $\mathcal{G}$, standard methods are to upweight examples proportional to their group size or to remove samples from larger groups. Specifically, group upweighting (GUW) seeks to minimize the following objective:

$$\theta^{(GUW)} = \max_\theta \frac{1}{k} \sum_{\mathcal{G}_i\in\mathcal{G}} \frac{n}{|\mathcal{G}_i|} \sum_{(x,y)\in\mathcal{G}_i} \mathbb{1}(f(x;\theta) = y), \tag{3}$$

where the inner sum is upweighted by a factor inversely proportional to its prevalence in the dataset. Group downsampling (GDS) takes a similar approach, but sub-samples all groups which are larger than the smallest group. Denoting $n_{\text{min}} = \min_{\mathcal{G}_i\in\mathcal{G}} |\mathcal{G}_i|$ as the size of the smallest group, we can write the GDS objective as:

$$\theta^{(GDS)} = \max_\theta \frac{1}{kn_{\text{min}}} \sum_{\mathcal{G}_i\in\mathcal{G}} \sum_{(x,y)\in\overline{\mathcal{G}_i}} \mathbb{1}(f(x;\theta) = y), \tag{4}$$

 https://doi.org/10.52202/079017-4424

where $\overline{\mathcal{G}_i}$ is a downsampled version of $\mathcal{G}_i$ with size n_{min}.

These two approaches are examined theoretically in Chaudhuri et al. [4], Welfert et al. [26], though their performance in the presence of label noise is unclear. Patrini et al. [19] suggest that a related upweighting factor to Equation (3) may be helpful in combating label noise (the so-called backwards correction) suggesting that GUW may be robust to label noise. Welfert et al. [26] prove that GUW and GDS are equivalent in the statistical setting (infinitely many samples), which suggests that GDS should also have some resilience to label noise. We explore this empirically in Section 4.

2.3 Two-Stage Last-Layer Model Correction

When group membership information is unavailable, techniques that build on GUW or GDS have been explored. Broadly they fall into the category of two-stage corrections [6, 12, 16, 14], and more specifically in our setting, last-layer model corrections for worst-group accuracy [11, 22]. These methods make the assumption that the groups of interest are determined by a tuple of class label Y and domain label D. Additionally there is a spurious correlation between Y and D which causes the base model f to perform poorly on minority groups. This same correlation is exploited when correcting the model in that an error set, $\mathcal{E}$, is constructed that is likely to contain minority examples:

$$\mathcal{E} = \{(X, Y) \in \mathcal{D} : f_{\text{bias}}(X) \neq Y\}, \tag{5}$$

where f_{bias} is the base model f for LaBonte et al. [11] and a highly regularized model intended to increase performance in Stromberg et al. [22]. This error set is then used as a proxy for a group-balanced or minority-dominated subset, respectively, for fair retraining. We present a simplified version of the RAD [22] and SELF [11] algorithms in Algorithm 1 and Algorithm 2 respectively. Note that in this work we focus on the misclassification version of SELF, but other variations are discussed by LaBonte et al. [11]. The intuition behind these methods is that the error set will likely contain mostly minority examples as the majority should be easily captured by the biased classifier.

The construction of this error set is precisely why using a noise-robust loss [16] or a corrected loss [19] is not enough to achieve robustness to label noise for two-stage LLR methods. The robust models will still (correctly) misclassify noisy examples, including them in the error set for retraining (see Appendix C for experimental results in this setting). This results in an error set which is dominated by noisy majority points rather than clean minority points. This is exacerbated by SELF which keeps only the examples with highest loss (most likely to be noisy). We present a proof of this failure for RAD in a group-conditional Gaussian setting in Appendix B Proposition 2. Oh et al. [16] circumvent this problem by changing the construction of the error set to depend on the entropy of the predictions rather than their correctness. While this increases robustness to label noise, it comes at the cost of clean performance. We instead focus on cleaning the labels so that we can realize the performance gains of existing two-stage methods.

Algorithm 1 RAD [22] Algorithm

Input: $\mathcal{D} = (x_i, y_i)$ for $i \in [n]$, c: penalty factor, λ: upweight factor
$\mathcal{E} \leftarrow \emptyset$
Learn θ_{bias} minimizing

$$\sum_{(x,y)\in\mathcal{D}} \ell(x, y; \theta) + c\|\theta\|_1 \tag{6}$$

$\hat{\boldsymbol{y}} \leftarrow f(\boldsymbol{x}; \theta_{\text{bias}})$
if $\hat{y}_i \neq y_i$ **then**
  $\mathcal{E} \leftarrow \mathcal{E} \cup (x_i, y_i)$
end if
Learn θ_{RAD} minimizing

$$\sum_{(x,y)\in\mathcal{D}\setminus\mathcal{E}} \ell(x, y; \theta) + \lambda \sum_{(x,y)\in\mathcal{E}} \ell(x, y; \theta) \tag{7}$$

Return: θ_{RAD}

Algorithm 2 SELF [11] Algorithm

Input: $\mathcal{D} = (x_i, y_i)$ for $i \in [n]$, n_{sub}: size of reweighting set
$\mathcal{E} \leftarrow \emptyset$
$\hat{\boldsymbol{y}} \leftarrow f(\boldsymbol{x}; \theta)$
if $\hat{y}_i \neq y_i$ **then**
  $\mathcal{E} \leftarrow \mathcal{E} \cup (x_i, y_i)$
end if
Subsample $\mathcal{E}$ to select n_{sub} examples with highest loss
Subsample $\mathcal{E}$ so that each class is equally represented
Learn θ_{SELF} minimizing, starting from θ

$$\sum_{(x,y)\in\mathcal{E}} \ell(x, y; \theta) \tag{8}$$

Return: θ_{SELF}

3 Label Spreading for Robust Worst-Group Accuracy

To correct for label noise in the training data, we will consider the labels of each point's nearest neighbors. We denote the k nearest neighbors matrix as V_k where $V_k(i,j)$ is 1 if X_j is one of the k nearest neighbors of X_i in ℓ_2 distance and 0 otherwise. Thus $\frac{1}{k}V_k$ is a row-stochastic matrix with a uniform distribution on each point's k nearest neighbors. Note that V_k is not necessarily symmetric and so defines a directed graph.

We update the estimated labels iteratively by taking a uniformly-weighted majority vote of each point's nearest neighbors. The idea is that noise is added at random to each class, thus it is likely that for noise level p, $1-p$ proportion of a given query point's neighbors have clean labels. It is intuitive to select a higher k for increasing p in order to minimize the effect of local noise density. Thus most noisy points will be corrected after just one round of spreading, and noise will continue to diminish unless there is a small cluster which is especially noisy. See Gao et al. [5] for a more complete theoretical treatment of kNN classification with label noise.

To better understand how the choice of k is affected by the noise parameter p consider the following:

Proposition 1 (Theorem 2 from Gao et al. [5])**.** *For $k \geq 8$ and symmetric label noise level p*

$$\mathbb{E}_{\mathcal{D}}[\mathcal{R}_k] \leq \mathcal{R}^* + \frac{2\mathcal{R}^*}{\sqrt{k}} + \frac{p}{(1-2p)\sqrt{k}} + 5\max\{L, \sqrt{L}\}\sqrt{d}\left(\frac{k}{n}\right)^{1/(1+d)} \tag{9}$$

where $\mathcal{R}^*$ is the Bayes optimal risk, $\mathcal{R}_k$ is the risk of kNN, d is the data feature dimensions, and L is the Lipschitz constant of the Bayes optimal classifier.

For $d \gg 1$, minimizing the upper bound in Equation (9) leads to the dependence of k on p as

$$\mathcal{O}\left(\left(\mathcal{R}^* + \frac{p}{1-2p}\right)^2\right), \tag{10}$$

corroborating the intuitive choice of larger k for larger amounts of noise.

3.1 A Note on Domain Label Spreading

Because of the effectiveness of simple domain-aware LLR methods like GDS and GUW, it may be tempting to consider kNN label spreading for domain noise as well as target noise. In this way, we could correct for the failures of GUW and GDS seen by Stromberg et al. [22] which lead them to propose RAD. The effectiveness of kNN label spreading for target labels is mainly due to the fact that the clean embeddings are close to linearly separable (that is the original base model had high accuracy). In practice, not only are classes linearly separable in the latent space, but are tightly clustered. This has been observed empirically and partially explained theoretically under the framework of neural collapse [18]. Such collapse has not been seen for subgroups when this information is not explicitly utilized in the training of the base model. Thus it is unreasonable to expect that domains should be as well separated as classes. This limits the effectiveness of kNN label spreading for domain labels, but as we demonstrate in Section 4, kNN-RAD can achieve SOTA WGA without using domain labels at all in the training phase. Nam et al. [15] bypass this issue by training a separate neural network to predict group membership using semi-supervised data, effectively causing neural collapse for groups rather than classes. However, their method suffers from poor computational performance and struggles to deal with label noise as demonstrated by Oh et al. [16].

3.2 kNN-RAD and kNN-SELF

In order to retrain the base model, we first perform T rounds of kNN label spreading (in practice we take $T = 1$) and then pass the cleaned labels to any last layer method. We denote the combination of these procedures **kNN-RAD**, Algorithm 3, or **kNN-SELF**, Algorithm 4, depending on the two-stage method used after label spreading. While we focus on using embeddings from a model trained on

 https://doi.org/10.52202/079017-4424

clean data, prior work suggests that the embeddings should be relatively robust to label noise [9, 17].

Algorithm 3 kNN-RAD Algorithm

Input: $\mathcal{D} = (x_i, \tilde{y}_i)$ for $i \in [n]$, k: number of neighbors, c: penalty factor, λ: upweight factor

Calculate kNN graph V_k
$\hat{\boldsymbol{y}}^{(0)} \leftarrow \tilde{\boldsymbol{y}}$
for $t = 1, \ldots, T$ **do**
  $\hat{\boldsymbol{y}}^{(t)} \leftarrow \mathbb{1}(V_k \hat{\boldsymbol{y}}^{(t-1)} \geq \frac{k}{2})$
end for
$\theta_{\text{RAD}} \leftarrow \text{RAD}(\boldsymbol{x}, \hat{\boldsymbol{y}}^{(T)}, c, \lambda)$ (Algorithm 1)
Return: θ_{RAD}

Algorithm 4 kNN-SELF Algorithm

Input: $\mathcal{D} = (x_i, \tilde{y}_i)$ for $i \in [n]$, k: number of neighbors, n_{sub} size of reweighting set

Calculate kNN graph V_k
$\hat{\boldsymbol{y}}^{(0)} \leftarrow \tilde{\boldsymbol{y}}$
for $t = 1, \ldots, T$ **do**
  $\hat{\boldsymbol{y}}^{(t)} \leftarrow \mathbb{1}(V_k \hat{\boldsymbol{y}}^{(t-1)} \geq \frac{k}{2})$
end for
$\theta_{\text{SELF}} \leftarrow \text{SELF}(\boldsymbol{x}, \hat{\boldsymbol{y}}^{(T)}, n_{\text{sub}})$ (Algorithm 2)
Return: θ_{SELF}

4 Experiments

We present worst-group accuracies for several representative methods across four large publicly available datasets. Specifically, we compare kNN-RAD and kNN-SELF to END [16] which aims to provide robustness to label noise for two-stage full retraining methods. As baselines we include group upweighting (GUW) and group downsampling (GDS) which are simple and effective, but require domain annotations which are not available to other methods. Additionally, we demonstrate that uncorrected RAD [22] and SELF [11] suffer large performance degradation under label noise.

4.1 Experimental Details

We perform our experiments with several common datasets in the literature of worst-group accuracy, including three vision datasets and one text dataset. Following prior work [10, 11], we use half of the validation as retraining data (with noisy target labels) and half as a *clean* holdout. More details can be found in Appendix A.

CMNIST [1] is a variant of the MNIST handwritten digit dataset in which digits 0-4 are labeled $y = 0$ and digits 5-9 are labeled $y = 1$. Further, 90% of digits labeled $y = 0$ are colored green and 10% are colored red. The reverse is true for those labeled $y = 1$. Thus, we can view color as a domain, and we can see that the color of the digit and its label are correlated.

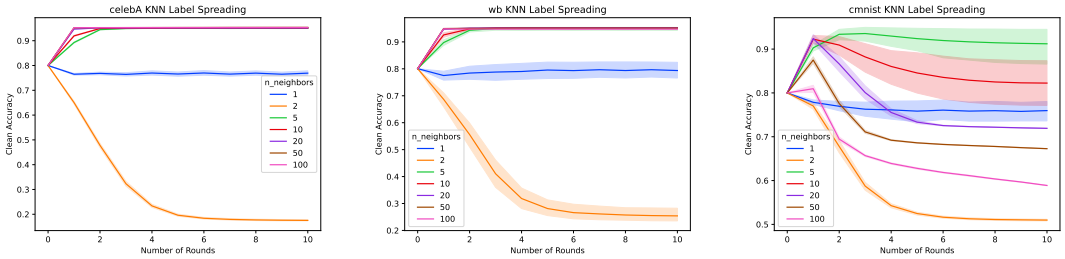
CelebA [13] is a dataset of celebrity faces. For this data, we predict hair color as either blonde ($y = 1$) or non-blonde ($y = 0$) and use gender, either male ($d = 1$) or female ($d = 0$), as the domain label. There is a correlation in the dataset between hair color and gender because of the prevalence of blonde female celebrities.

Waterbirds [21] is a semi-synthetic dataset which places images of land birds ($y = 1$) or sea birds ($y = 0$) on land ($d = 1$) or sea ($d = 0$) backgrounds. There is a correlation between background and the type of bird in the training data but this correlation is removed in the validation data.

CivilComments [3] is a text corpus dataset of public comments on news websites. Comments are labeled either as toxic ($y = 1$) or civil ($y = 0$) and the spurious attribute is the presence ($d = 1$) or absence ($d = 0$) of a minority identifier (e.g. LGBTQ, race, gender). There is a strong class imbalance (most comments are civil), though the domain imbalance is modest.

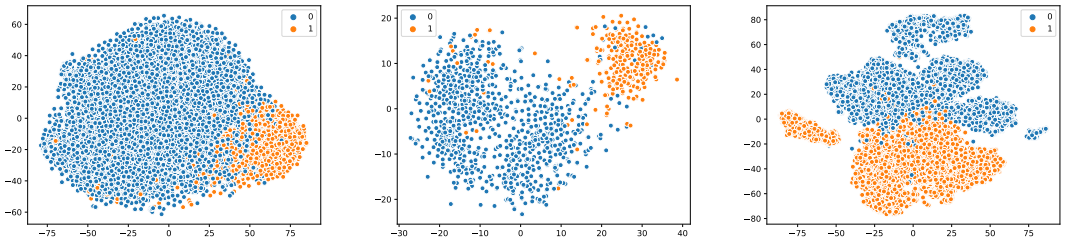
4.2 Empirical Evidence for Label Spreading

To empirically evaluate the effectiveness of kNN label spreading, we add symmetric label noise to embeddings from three image datasets and then perform label spreading over 10 rounds. We see in Figure 1 that kNN label spreading can correct most of the noise present in the training data with only one round of spreading. This is likely due to the well-separated nature of the embeddings. CMNIST suffers from decreasing performance with too many rounds or neighbors as the data is not perfectly clustered into classes as it is in the other image datasets. Here, it is likely that the nearest neighbors for the small clusters to the side in Figure 2 are of the opposite class, allowing noisy points to spread more easily than clean points. For this reason, we employ a smaller value of k for CMNIST than for other



(a) CelebA (b) Waterbirds (c) CMNIST

Figure 1: Accuracy (and 95% confidence intervals over 10 runs) of predicted labels from kNN under 20% symmetric label noise. CelebA and Waterbirds achieve strong performance with a large number of nearest neighbors, but CMNIST struggles as the number of neighbors or rounds grows too large.



(a) CelebA (b) Waterbirds (c) CMNIST

Figure 2: tSNE projection of the 2048 dimensional latent embeddings into a 2 dimensional space for visualization. We see that CelebA and Waterbirds show clear class separation while CMNIST has more hierarchical clustering. This could lead to decreased performance of label spreading.

datasets. Distance-weighted voting could be another way to combat this issue. Additionally, because the label spreading process is so quick, we spread for only 1 round in practice. A distance-weighted approach may require more rounds for clean labels to spread throughout the dataset.

4.3 Robustness of kNN to Noisy Embeddings

While we assume that the embedding model Φ is train on clean (but imbalanced) data, it is not always the case in practice. While Iscen et al. [9] suggest that embeddings are relatively robust to noise, they find that the classification head is not. Both RAD and SELF rely on the embeddings and SELF on the classification head as well. Additionally, the kNN label spreading proposed in this work assumes clean embeddings with only symmetric label noise.

To explore how robust our method is to violation of these assumptions, we train base models for both CelebA and Waterbirds on data which contained 20% SLN. We then test the finetuning procedure with varying levels of label noise. We see in Table 1 that both RAD and SELF suffer when using noisy embeddings, and this is only exacerbated by noise in the finetuning set. Still, kNN label cleaning is able to provide a significant boost in performance despite having relatively low-quality embeddings. We see a similar trend for Waterbirds in Table 2 which shows even stronger performance gains for kNN over vanilla two-stage methods.

It is also notable that SELF has significantly worse performance than RAD when using noisy embeddings, likely because SELF additionally reuses the final classification layer of the base model. Previous work [8] has shown that most of the drop in performance when training deep models with noisy data comes from a poor classification head.

4.4 Results

We report the worst-group accuracy of each approach and its standard deviation over 10 noise seeds. Note that for END [16], we report the results of Oh et al. [16]. For SELF [11] and RAD [22], we implement their algorithms ourselves to ensure a fair comparison. Each table involves the following:

https://doi.org/10.52202/079017-4424

Table 1: **CelebA WGA (std. dev.)** using embeddings from a noisy base model. We see that kNN label correction still offers a significant increase in performance over vanilla two-stage methods.

Method	0%	10%	20%	30%
RAD	80.55 (0.02)	74.78 (2.32)	0 (0)	0 (0)
SELF	31.86 (2.32)	40.55 (0.66)	17.68 (16.71)	50.55 (1.36)
KNN - RAD	80.55 (0.02)	77.38 (1.7)	76.56 (2.24)	69.94 (3.07)
KNN - SELF	31.01 (0.42)	41.9 (3.84)	44.8 (4.64)	46 (7.20)

Table 2: **Waterbirds WGA (std. dev.)** using embeddings from a noisy base model. We see that kNN strongly outperforms vanilla two-stage methods at every noise level.

Method	0%	10%	20%	30%
RAD	86.12 (0.05)	52.49 (2)	32.26 (2.99)	16.94 (2.5)
SELF	67.43 (4.48)	21.67 (3.12)	16.71 (1.30)	6.43 (19.30)
KNN - RAD	84.04 (0.04)	74.28 (2.63)	73.78 (3.29)	61.07 (9.68)
KNN - SELF	68.96 (3.79)	68.02 (1.9)	74.90 (2.73)	49.41 (9.89)

- Each table is broken into three parts: the first are simple baselines with access to *clean* domain annotations, the second are SOTA two-stage LLR methods for WGA, and the third are robust two-stage methods including our own (last layer retraining) kNN-RAD and kNN-SELF.
- We write GUW* and GDS* to denote the worst-group accuracies of upweighting and downsampling, respectively, that can be achieved with oracle access to *clean* domain labels at training time. Note that such access is *not available* to the two-stage methods.
- We highlight in **bold** both the best domain annotation-free method for each noise level and the method within one standard deviation of the best.
- The "Domain Annotation" column denotes whether a method requires access to domain annotation only at validation (model selection) time or over both training and validation phases. The "Layer" column denotes whether a given method retrains only the *last* layer or the *full* model.

For the CMNIST dataset, in Table 3 we see that kNN label spreading gives small gains over vanilla RAD, but even vanilla RAD is not significantly diminishing in performance with increasing noise levels. This is likely due to the ease of the dataset overall. As noted in Section 3, CMNIST is not as strongly clustered as other image datasets, so it is necessary to use a smaller number of nearest neighbors. Here, kNN-SELF provides small gains over vanilla SELF as both are fairly robust.

For the CelebA dataset, Table 4 shows that kNN-RAD achieves dramatically improved performance relative to vanilla RAD and also over END. Although it retrains only the last layer, kNN-RAD achieves SOTA performance for domain annotation-free methods at every noise level for this dataset. In fact, kNN-RAD is competitive with domain-aware methods such as GDS and GUW across noise levels. Here kNN-SELF falls behind kNN-RAD and END, but still achieves significantly more robustness than vanilla SELF. Note the relatively higher variance of WGA for SELF and kNN-SELF; this is likely due to the downsampling step that induces different data distributions for different seeds.

For the Waterbirds dataset, we see in Table 5 that kNN-RAD strongly outperforms END and is competitive with domain-aware methods across noise levels, improving the performance of even the oracle methods GDS and GUW. Vanilla RAD experiences a catastrophic failure at 20% label noise and above which kNN-RAD is able to correct. SELF performs well at 0% label noise, but quickly degrades with label noise. kNN-SELF achieves dramatic gains over vanilla SELF and maintains an edge over END at every noise level.

For the CivilComments dataset, we see in Table 6 that kNN-RAD is able to match the performance of the domain-aware methods without having access to domain information. Additionally, kNN label spreading demonstrates significant gains over vanilla RAD. SELF struggles on CivilComments even in the experiments of LaBonte et al. [11], so it is no surprise that it performs poorly here. The heavy class imbalance is likely the culprit combined with the class balancing performed in Algorithm 2.

Table 3: **CMNIST WGA (std. dev.)**: GUW* and GDS* denote the worst-group accuracies of upweighting and downsampling, respectively, achieved with oracle access to *clean* domain labels which aren't available to the two-stage methods. We list both the best domain annotation-free method for each noise level and the method within one standard deviation of the best in **bold**. We see that CMNIST is a relatively easy dataset in general, so label noise does not cause dramatic performance loss. Yet, kNN provides some additional robustness for both RAD and SELF. CMNIST is not considered in [16] and thus, results for END are not reported.

			Label Noise (%)			
Method	Domain Annotation	Layer	0	10	20	30
GUW*	Training	Last	95.27 (0.07)	95.17 (0.53)	93.11 (0.94)	92.05 (1.23)
GDS*	Training	Last	95.37 (0.22)	94.19(1.02)	94.35 (1.29)	93.31 (0.98)
RAD [22]	Val	Last	**93.41** (0.79)	89.62 (0.72)	89.52 (0.65)	88.78 (0.89)
SELF [11]	Val	Last	92.04 (0.20)	89.98 (1.16)	88.05 (2.35)	**90.58** (1.12)
END [16]	Val	Full	-	-	-	-
kNN-RAD	Val	Last	**93.42** (0.63)	**92.46** (0.92)	**91.95** (0.86)	**90.50** (3.59)
kNN-SELF	Val	Last	91.77 (0.21)	**92.81** (0.88)	91.06 (1.06)	**90.16** (3.67)

Table 4: **CelebA WGA (std. dev.)**: We see that RAD and SELF achieve strong performance at 0% noise, but are not robust at larger noise levels. Here kNN-RAD and kNN-SELF maintain strong performance relative to their vanilla counterparts up to 30% noise and kNN-RAD strongly outperforms END at all noise levels.

			Label Noise (%)			
Method	Domain Annotation	Layer	0	10	20	30
GUW*	Training	Last	86.67 (0)	84.58 (1.21)	82.74 (1.42)	82.08 (2.50)
GDS*	Training	Last	85.72 (1.65)	84.63 (1.58)	84.06 (2.40)	83.12 (1.91)
RAD [22]	Val	Last	**83.89** (0)	81.80 (0.37)	0 (0)	0 (0)
SELF [11]	Val	Last	83.48 (0)	60.48 (6.09)	59.99 (4.52)	60.71 (3.70)
END [16]	Val	Full	82.6 (2)	79.7 (1)	**81.1** (2)	**77.8** (3)
kNN-RAD	Val	Last	**83.89** (0)	**83.20** (1.53)	**82.29** (1.25)	**79.5** (1.91)
kNN-SELF	Val	Last	83.48 (0)	72.96 (22.96)	75.77 (8.02)	73.98 (4.40)

5 Discussion and Limitations

We observe that kNN label spreading dramatically increases the robustness of both RAD and SELF and additionally find several interesting trends that warrant discussion. First, it is clear that for larger noise levels, a larger number of nearest neighbors is needed to correct for the noise. This is suggested by Gao et al. [5] for large k, but is empirically verified with our experiments. The empirically optimal k value increases with each increase in label noise, which allows the label spreading to average over a larger number of nearest neighbors. For CMNIST, we must use a smaller, but still increasing, k to account for the smaller clusters within classes. A limitation of our method is that k is chosen as a hyperparameter, but these insights help to suggest likely ranges for the optimal. Indeed, if an estimate of the noise prevalence is available, the choice of k nearest neighbors could reasonably be estimated without hyperparameter tuning at all.

We also note that the quality of the embeddings is crucial in the selection of the number of neighbors and the number of spreading rounds. CMNIST is the prime example of a dataset that is generally well-separated but has not yet experienced "neural collapse," so each class has several clusters of points. The danger of this is that some clusters may be nearer to the opposite class than their own, requiring fewer nearest neighbors to prevent noisy labels from spreading. The base model accuracy can help indicate how extensively it is trained, though embeddings can also be analyzed qualitatively. As pointed out in Section 3, a limitation of our method is that we assume that embeddings are from a model trained on clean data, but this need not be the case. Indeed Iscen et al. [9] use the robustness of embeddings to label noise in order to train robust deep learning models and we verify the robustness of kNN-RAD in particular in Section 4.3.

 https://doi.org/10.52202/079017-4424

Table 5: **Waterbirds WGA (std. dev.)**: We see that both kNN-RAD and kNN-SELF strongly outperform END even though they update only the last layer. Non-robust methods fail quickly.

Method	Domain Annotation	Layer	Label Noise (%) 0	10	20	30
GUW*	Training	Last	91.60 (0.05)	90.90 (0.78)	88.25 (1.91)	84.78 (3.53)
GDS*	Training	Last	92.32 (0.58)	89.53(1.66)	86.93 (2.40)	78.09 (3.36)
RAD [22]	Val	Last	91.23 (0.06)	79.33 (1.38)	50.74 (2.23)	19.52 (1.91)
SELF [11]	Val	Last	**92.83** (0.49)	7.58 (2.77)	1.38 (0.20)	0.66 (0.14)
END [16]	Val	Full	82.8 (1)	84.2 (1)	83.2 (1)	81.8 (1)
kNN-RAD	Val	Last	90.92 (0.08)	**90.72** (0.63)	**89.93** (1.10)	**86.90** (3.07)
kNN-SELF	Val	Last	**92.65** (0.47)	89.55 (0.65)	88.24 (2.93)	82.44 (10.43)

Table 6: **Civil Comments WGA (std. dev.)**: SELF struggles on this highly class-imbalanced dataset, but kNN-RAD is competitive with domain-aware methods even for large noise. Oh et al. [16] do not consider this dataset and thus, results for END are not reported.

Method	Domain Annotation	Layer	Label Noise (%) 0	10	20	30
GUW*	Training	Last	80.25 (0.03)	80.04 (0.43)	81.58 (0.18)	79.39 (0.70)
GDS*	Training	Last	79.67 (0.48)	80.16(0.63)	81.18 (0.53)	81.21 (0.32)
RAD [22]	Val	Last	**80.99** (0.03)	79.25 (0.44)	77.45 (0.71)	54.36 (1.30)
SELF [11]	Val	Last	60.61 (0.04)	59.92 (0.03)	59.92 (0.04)	59.95 (0.03)
END [16]	Val	Full	-	-	-	-
kNN-RAD	Val	Last	**81.0** (0.03)	**81.15** (0.60)	**80.70** (0.62)	**78.56** (1.52)
kNN-SELF	Val	Last	60.65 (0.05)	72.30 (1.58)	64.64 (2.72)	61.30 (0.72)

Finally, we note that while kNN label spreading works well with both SELF and RAD, the down-sampling to balance the class aspect of the SELF algorithm combined with label spreading leads to a much larger variance than with kNN-RAD. That said, preprocessing with a kNN label spreading module still achieves significant gains in worst-group accuracy relative to vanilla SELF. A coupled label cleaning and error set selection process may be an interesting direction for future work. By leaving our preprocessing generic, we can ensure our method is applicable when future two-stage methods are developed.

Our work allows for the robustness of two-stage fairness corrections, which could improve fairness for minority groups in a wide variety of models. Unfortunately, no two-stage last layer correction can provide guarantees on the fairness of the resulting model in the general setting. This could lead to practitioners assuming fairness without auditing it.

6 Acknowledgements

Nathan Stromberg, Rohan Ayyagari, and Lalitha Sankar acknowledge support by NSF CIF-2007688, SCH-2205080, PIPP-2200161, and a Google AI for Social Good grant. Sanmi Koyejo acknowledges support by NSF Career Award 2046795 and SCH-2205329, Stanford HAI, and Google Inc.

References

[1] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[2] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. 2006.

[3] D. Borkan, L. Dixon, J. Sorensen, N. Thain, and L. Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561, 2019. URL `http://arxiv.org/abs/1903.04561`.

[4] K. Chaudhuri, K. Ahuja, M. Arjovsky, and D. Lopez-Paz. Why does throwing away data improve worst-group error? In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[5] W. Gao, B.-B. Yang, and Z.-H. Zhou. On the resistance of nearest neighbor to random noisy labels. (arXiv:1607.07526), Sept. 2018. URL `http://arxiv.org/abs/1607.07526`. arXiv:1607.07526 [cs].

[6] G. Giannone, S. Havrylov, J. Massiah, E. Yilmaz, and Y. Jiao. Just mix once: Mixing samples with implicit group distribution. In *NeurIPS 2021 Workshop on Distribution Shifts*, 2021.

[7] B. Y. Idrissi, M. Arjovsky, M. Pezeshki, and D. Lopez-Paz. Simple data balancing achieves competitive worst-group-accuracy. In *Proceedings of the First Conference on Causal Learning and Reasoning*, 2022.

[8] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5070–5079, 2019.

[9] A. Iscen, J. Valmadre, A. Arnab, and C. Schmid. Learning with neighbor consistency for noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4672–4681, 2022.

[10] P. Kirichenko, P. Izmailov, and A. G. Wilson. Last layer re-training is sufficient for robustness to spurious correlations. In *The Eleventh International Conference on Learning Representations*, 2023.

[11] T. LaBonte, V. Muthukumar, and A. Kumar. Towards last-layer retraining for group robustness with fewer annotations. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[12] E. Z. Liu, B. Haghgoo, A. S. Chen, A. Raghunathan, P. W. Koh, S. Sagawa, P. Liang, and C. Finn. Just train twice: Improving group robustness without training group information. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[13] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[14] J. Nam, H. Cha, S. Ahn, J. Lee, and J. Shin. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33:20673–20684, 2020.

[15] J. Nam, J. Kim, J. Lee, and J. Shin. Spread spurious attribute: Improving worst-group accuracy with spurious attribute estimation. *arXiv preprint arXiv:2204.02070*, 2022.

[16] D. Oh, D. Lee, J. Byun, and B. Shin. Improving group robustness under noisy labels using predictive uncertainty. *ArXiv*, abs/2212.07026, 2022.

[17] D. Ortego, E. Arazo, P. Albert, N. E. O'Connor, and K. McGuinness. Multi-objective interpolation training for robustness to label noise. (arXiv:2012.04462), Mar. 2021. URL `http://arxiv.org/abs/2012.04462`. arXiv:2012.04462 [cs].

[18] V. Papyan, X. Han, and D. L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40): 24652–24663, 2020.

 https://doi.org/10.52202/079017-4424

[19] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[20] S. Qiu, A. Potapczynski, P. Izmailov, and A. G. Wilson. Simple and fast group robustness by automatic feature reweighting. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[21] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020.

[22] N. Stromberg, R. Ayyagari, M. Welfert, S. Koyejo, and L. Sankar. Robustness to subpopulation shift with domain label noise via regularized annotation of domains, 2024.

[23] T. Sypherd, M. Diaz, J. K. Cava, G. Dasarathy, P. Kairouz, and L. Sankar. A tunable loss function for robust classification: Calibration, landscape, and generalization. *IEEE Transactions on Information Theory*, 68(9):6021–6051, 2022.

[24] T. Sypherd, R. Nock, and L. Sankar. Being properly improper. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20891–20932. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/sypherd22a.html`.

[25] J. Wei, Z. Zhu, H. Cheng, T. Liu, G. Niu, and Y. Liu. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations*, 2022.

[26] M. Welfert, N. Stromberg, and L. Sankar. Theoretical guarantees of data augmented last layer retraining methods, 2024.

[27] H. Yao, Y. Wang, S. Li, L. Zhang, W. Liang, J. Zou, and C. Finn. Improving out-of-distribution robustness via selective augmentation. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

A Experimental Details

The upstream base models for all the datasets which are considered are taken from [22]. This includes all the hyperparameters, data augmentations and training procedures that we use for the upstream models. We use these base models to generate the embeddings for the predefined validation and test splits of the datasets. In our experimental setup, we assume access to a small set of clean data which we use for validation and hyperparameter selection. We split the validation data into two halves. One half is used as the aforementioned clean set for validation and the other half is used for training during hyperparameter selection. During the final testing, we use the entire (noisy) validation split for retraining and the test split for testing. Once we do the hyperparameter selection, we run the algorithms over 10 different noise seeds and report the accuracy and variance over these 10 runs. This procedure is repeated for each dataset and each noise level.

A.1 Last Layer Retraining Methods

For the methods that involve only last-layer retraining (either with downsampling or upweighting), we only tune `c`, which is the inverse of the λ, the ℓ_1 regularization strength. For all datasets, we tune over 10 values equally spaced on the log scale ranging from 1e-4 to 1.

A.2 RAD

For each dataset, we fix the `c` values for both the identification and retraining models. We get these values from [22]. We tune the upweight factor for each dataset and each noise level. The hyperparameter values and ranges used are given in Table 7. The identification model is implemented using the `PyTorch` library and the retraining model is implemented using the `sklearn.linear_model.LogisticRegression` package. The `learning rate (id)` is the learning rate of the identification model and the `epochs (id)` is the number of epochs used to train the identification model.

A.3 kNN-RAD

We follow the same procedure for kNN-RAD as we did for RAD but with the added kNN based label spreading preprocessing step. In addition to the upweight factor, we also tune `n_neighbors`, which is the number of nearest neighbours used in the kNN algorithm. When there is no noise, we fix the `n_neighbors` value at 1. We use the `sklearn.neighbors.KNeighborsClassifier` package to perform the label spreading step. The ranges and hyperparameter values used are given in Table 8.

A.4 SELF

We implemented misclassification-SELF using code adapted from LaBonte et al. [11] so that it would be compatible with our setup where we use pre-generated embeddings from the base models. We fix the `finetuning steps` which the number of steps of fine-tuning we perform once we construct the class balanced error set. We tune the learning rate and the number of points that are selected for class balancing. The hyperparameter values and ranges used are given in Table 9

A.5 kNN-SELF

The procedure for kNN-SELF is the same as SELF but with the added kNN based label spreading preprocessing step. We fix the hyperparameters using the values selected in the hyperparameter selection in SELF. We additionally tune `n_neighbors`. We use the `sklearn.neighbors.KNeighborsClassifier` package to perform the label spreading step. The ranges and hyperparameter values used are given in Table 10

 https://doi.org/10.52202/079017-4424

Table 7: **RAD Hyperparameters**

Dataset	c (id)	c (retraining)	LR (id)	epochs (id)	upweight factor range
CelebA	6.16e-4	0.007848	1e-5	6	[5, 10, 25, 50]
Waterbirds	3.0e-6	0.143845	1e-5	60	[5, 10, 25, 50]
CMNIST	33.6	0.007848	1e-5	6	[1, 3, 20, 30]
Civilcomments	6.95e-07	0.001833	1e-5	6	[1, 3, 6, 10]

Table 8: **KNN - RAD Hyperparameters**

Dataset	c (id)	c (retraining)	LR (id)	epochs (id)	num neighbors range	upweight factor range
CelebA	6.16e-4	0.007848	1e-5	6	[5, 11, 21]	[10, 25, 50, 75]
Waterbirds	3.0e-6	0.143845	1e-5	60	[5, 11, 21, 31]	[10, 25, 50, 75]
CMNIST	33.6	0.007848	1e-5	6	[3, 5, 7]	[1, 3, 20, 30]
Civilcomments	6.95e-07	0.001833	1e-5	6	[5, 11, 21]	[6, 10, 25, 50]

Table 9: **SELF Hyperparameters**

Dataset	fine-tuning steps	learning rate range	num points range
CelebA	500	[1e-6, 1e-5, 1e-4]	[2, 20, 100]
Waterbirds	500	[1e-4, 1e-3, 1e-2]	[20, 100, 500]
CMNIST	500	[1e-5, 1e-4, 1e-3]	[100, 500, 700]
Civilcomments	200	[1e-6, 1e-5, 1e-4]	[20, 100, 500]

Table 10: **KNN - SELF Hyperparameters**

Dataset	fine-tuning steps	learning rate	num points	num neighbors
CelebA	500	1e-5	2	[11, 25, 37]
Waterbirds	500	1e-4	500	[5, 11, 21]
CMNIST	250	1e-5	500	[7, 21, 41]
Civilcomments	200	1e-6	500	[11, 31, 41]

A.6 Compute Resources

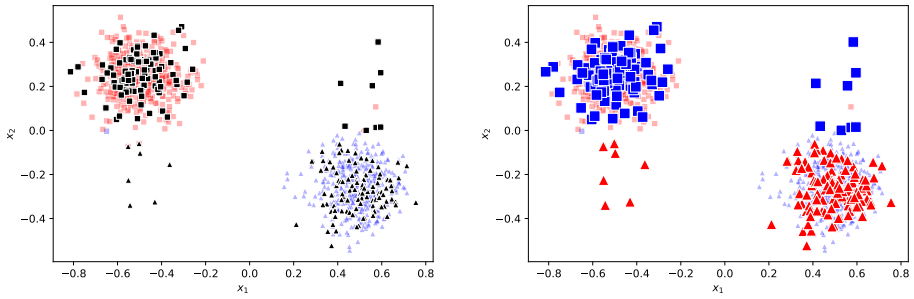
Experiments in Section 4 and Appendix C were conducted on a supercomputing cluster using NVIDIA GPUs for hardware acceleration. Most compute time is spent training base models which needs to be done just once per dataset. Beyond that, experiments finish within hours.

B RAD Fails Under Label Noise

Consider the data distribution introduced in Yao et al. [27] (also used in[26]) with group-conditional Gaussians aligned such that the difference between means within a class is given by $[C_1, 0, \ldots, 0] \in \mathbb{R}^d$ and the difference between means across classes is given by $[0, C_2, 0, \ldots, 0] \in \mathbb{R}^d$. That is, we have axis-aligned core and spurious features. Refer to Figure 3 for a visual.

Proposition 2 (Axis-Aligned Gaussian Mixture)**.** *Let RAD use the fixed upweighting scheme* $\frac{|\mathcal{E}^c|}{|\mathcal{E}|}$*. For the dataset described above, with* $C_2 > C_1 > C_0$ *for sufficiently large* C_0*, RAD fails entirely when learning with symmetric label noise* $p > 0$*. Otherwise, RAD fails if* $\frac{1}{2} > p > 2\pi_0$*, achieving lower worst-group accuracy than ERM.*

Proof. We assume that λ, the ℓ_1 regularization strength of the biased classifier, is large enough that only the first (spurious) or second (core) feature is learned. Consider the two cases separately:



(a) Error Set (b) Upweighted points

Figure 3: Label noise causes the RAD algorithm to select, and then upweight, the incorrect error set. We show that RAD fails under suffient SLN for imbalanced settings.

Spurious Feature is Learned Consider the error set $\mathcal{E}$. It is made of up clean minority samples and noisy majority samples. When upweighting, the effective noise level in the retraining set will increase for the majority and decrease for the minority. For $2\pi_0 < p < \frac{1}{2}$, as π_0 goes to 0, the effective noise in the upweighted majority is 50% and 0% in the upweighted minority. In this case, the optimal separator will split the majorities and correctly classify the minorities. By assumption, $C_2 > C_1$ so the ℓ_1 penalty in the retraining step will force the spurious feature to be learned again. Note that ERM would learn a classifier biased to the majorities, but not be restricted to only the spurious feature. Thus RAD performs worse than ERM.

Core Feature is Learned The error set $\mathcal{E}$ now consists of entirely noisy samples which are upweighted by a factor of $\frac{1-p}{p}$. This results in an effective noise level at retraining of 50%, making the retraining task impossible. Thus RAD fails. □

C α-RAD, Why Robust Losses are Not Enough

It may be tempting to believe that simply using a robust loss should be enough to make two-step methods robust to label noise. The issue is that while robust losses can learn a classifier which performs well on clean data, it will (correctly) misclassify noisy examples. This leads to noisy points (which will much more likely to be from a majority group) being selected for the error set which is used to retrain the final classifier. Thus the final classifier will not be trained on data from minority groups as intended, but on mostly majority points, exacerbating unfairness. To demonstrate this, we train RAD [22] using α-loss in the identification step. α-loss [23] has been demonstrated to be robust to symmetric label noise both theoretically [24] and empirically [24, 23].

We first examine the types of points which are selected by RAD in the error set. In Figure 4 we see that for all noise levels, true minority examples are misclassified (and therefore selected) but as the noise increases, the amount of noisy (former) majority points increases as well. These points drown out the effect of the true minority points, causing failure of RAD at large noise levels. We see in the Table 11 that even when training with this robust loss, worst-group accuracy experiences a dramatic dropoff at larger noise levels. This is the same failure mode that is demonstrated with vanilla cross entropy loss in the main text Section 4.

Table 11: **α-RAD**

	Label Noise (%)				
Dataset	0	5	10	15	20
CelebA	79.33 (0.48)	83.78 (1.28)	80.67 (1.78)	58.07 (0.1)	46.45 (0.15)
CMNIST	91.68 (0.37)	88.03 (1.47)	90.56 (1.08)	80.68 (1.08)	47.17 (2.56)

https://doi.org/10.52202/079017-4424

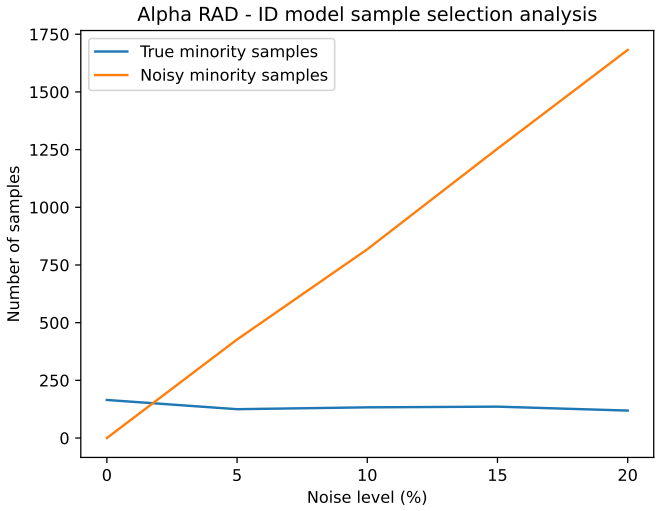


Figure 4: RAD trained with α-loss is able to capture minority points at all noise levels, but an increasing number of noisy majority points are selected as noise increases. This leads to poor downstream fairness

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We propose a drop-in correction which is able to achieve SOTA worst-group accuracy across several datasets as claimed in the abstract.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We point out that we use clean embeddings and have a clean holdout dataset, which is common in the literature. We additionally provide references that suggest clean embeddings may not be necessary.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings,

model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: No theoretical results are presented

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Full experimental details are provided in the appendix including datasets, model details, and hyperparameters. Algorithms are included in the main body. Code will be released publically on GitHub and is included as supplemental material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

 https://doi.org/10.52202/079017-4424

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is included in supplemental materials along with documentation. Code will additionally be released on GitHub after acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Data splits discussed in main body and training details are available in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Standard deviation of presented results are given over 10 independent runs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: These details are provided in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

https://doi.org/10.52202/079017-4424

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: Every step is taken to ensure that the code of ethics is adhered to, including providing resources for reproducibility and discussions on limitations and societal impact.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Impacts and limitations are discussed in the "Discussion and Limitations" section. Included are both positive and negative societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: No models are released and all data used is from publicly available datasets.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets are given appropriate citations in the experimental section of the main body.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Anonymized code is provided along with documentation on its execution.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No human subjects were used.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

https://doi.org/10.52202/079017-4424

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human subjects were used.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.