Mitigating Dataset Bias by Using Per-sample Gradient

Sumyeong Ahn[*]
Graduate School of AI
KAIST

Seongyoon Kim[*]
Dept. ISysE
KAIST

Se-young Yun
Graduate School of AI
KAIST

Abstract

The performance of deep neural networks is strongly influenced by the training dataset setup. In particular, when attributes having a strong correlation with the target attribute are present, the trained model can provide unintended prejudgments and show significant inference errors (*i.e., the dataset bias problem*). Various methods have been proposed to mitigate dataset bias, and their emphasis is on weakly correlated samples, called *bias-conflicting samples*. These methods are based on explicit bias labels provided by human. However, such methods require human costs. Recently, several studies have tried to reduce human intervention by utilizing the output space values of neural networks, such as feature space, logits, loss, or accuracy. However, these output space values may be insufficient for the model to understand the bias attributes well. In this study, we propose a debiasing algorithm leveraging gradient called PGD (**P**er-sample **G**radient-based **D**ebiasing). PGD comprises three steps: (1) training a model on uniform batch sampling, (2) setting the importance of each sample in proportion to the norm of the sample gradient, and (3) training the model using importance-batch sampling, whose probability is obtained in step (2). Compared with existing baselines for various datasets, the proposed method showed state-of-the-art accuracy for the classification task.

1 Introduction

Dataset bias [63, 57], is a bad training dataset problem that occurs when unintended easier-to-learn attributes *(i.e., bias attributes)*, having a high correlation with the target attribute, are present [56, 2]. This is due to the fact that the model can infer outputs by focusing on the bias features, which could lead to testing failures. For example, most "camel" images include a "desert background," and this unintended correlation can provide a false shortcut for answering "camel" on the basis of the "desert." In [48, 37], samples of data that have a strong correlation (like "desert background" in "camel" class images) are called "bias-aligned samples," while samples of data that have a weak correlation (like "camel on the grass" images) are termed "bias-conflicting samples."

To reduce the dataset bias, initial studies [29, 45, 58, 40] have frequently assumed a case where labels with bias attributes are provided, but these additional labels provided through human effort are expensive. Alternatively, the bias-type, such as "background" is assumed in [38, 17, 7, 10, 13]. However, assuming biased knowledge from humans is still unreasonable since even humans cannot predict the type of bias that may exist in a large dataset [53]. Data for deep learning is typically collected by web-crawling without thorough consideration of the dataset bias problem.

Recent studies [35, 48, 30, 37, 54, 67] have replaced human intervention with DNN results. They have identified bias-conflicting samples by using empirical metrics for output space (*e.g.,* training loss and accuracy). For example, [48] suggested a "relative difficulty" based on per-sample training loss and thought that a sample with a high "relative difficulty" was bias-conflicting sample. Most

[*] Two authors contribute equally

Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

of the previous research has focused on the output space, such as feature space (penultimate layer output) [37, 30, 7, 54, 67], loss [48], and accuracy [35, 43]. However, this limited output space can impose restrictions on describing the data in detail.

Recently, as an alternative, model parameter space (*e.g.,* gradient [23, 28, 47]) has been used to obtain high-performance gains compared to output space approaches for various target tasks. For example, [23] used gradient-norm to detect out-of-distribution detection samples and showed that the gradient of FC layer $\in \mathbb{R}^{h \times c}$ could capture joint information between feature and softmax output, where h and c are the dimension of feature and output vector, respectively. Since the gradients of each data point $\in \mathbb{R}^{h \times c}$ constitute high dimensional information, it is much more informative than the output space, such as logit $\in \mathbb{R}^{c}$ and feature $\in \mathbb{R}^{h}$. However, there is no approach to tackle the dataset bias problem using a gradient norm-based metric.

In this paper, we present a resampling method from the perspective of the per-sample gradient norm to mitigate dataset bias. Furthermore, we theoretically justify that the gradient-norm-based resampling method can be an excellent debiasing approach. Our key contributions can be summarized as follows:

- We propose PGD, **P**er-sample **G**radient-norm based **D**ebiasing, a simple and efficient gradient-norm-based debiasing method. PGD is motivated by prior research demonstrating [47, 23, 28] that gradient is effective at finding rare samples, and it is also applicable to finding the bias-conflicting samples in the dataset bias problem (See Section 3 and Appendix D).
- PGD outperforms dataset bias on various benchmarks, such as biased action recognition (BAR), biased FFHQ (BFFHQ), CelebA and CivilComments-WILD. (See Section 4)

2 Dataset Bias Problem

Classification model. We first describe the conventional supervised learning setting. Let us consider the classification problem when a training dataset $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^{n}$ with input image x_i and corresponding label y_i is given. Assuming that there are $c \in \mathbb{N} \setminus \{1\}$ classes, y_i is assigned to the one element in set $C = \{1, ..., c\}$. Note that we focus on a situation where dataset $\mathcal{D}_n$ does not have noisy samples, for example, noisy labels or out-of-distribution samples (e.g., SVHN samples when the task is CIFAR-10). When input x_i is given, $f(y_i|x_i, \theta)$ represents the softmax output of the classifier for label y_i. It is derived from the model parameter $\theta \in \mathbb{R}^{d}$. The cross-entropy (CE) loss $\mathcal{L}_{\text{CE}}$ is frequently used to train the classifier, and it is defined as $\mathcal{L}_{\text{CE}}(x_i, y_i; \theta) = -\log f(y_i|x_i, \theta)$.

Dataset bias. Let us suppose that a training set $\mathcal{D}_n$ is comprised of images, as shown in Figure 1, and that the objective is to classify the `digits`. Each image can be described by a set of attributes, (*e.g.,* for the first image in Figure 1, it can be {`digit 0`, `red`, `thin`,...}). The purpose of the training classifier is to find a model parameter θ that correctly predicts the target attributes, (*e.g.,* `digit`). Notably, the target attributes are also interpreted as *classes*. However, we focus on a case wherein another attribute that is strongly correlated to the target exists, and we call these attributes *bias attributes*. For example, in Figure 1, the bias attribute is `color`.

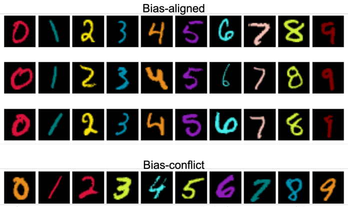


Figure 1: Target and bias attribute: `digit shape`, `color`.

Furthermore, samples whose bias attributes are highly correlated to the target attributes are called *bias-aligned* (top three rows in Figure 1). Conversely, weakly correlated samples are called *bias-conflicting* (see the bottom row of Figure 1). Therefore, our main scope is that the training dataset which have samples whose bias and target attributes are misaligned. [2] According to [48], when the bias attributes are easier-to-learn than the target attributes, dataset bias is problematic as the trained model may prioritize the bias attributes over the target attributes. For example, for a model trained on the images in Figure 1, the model can output class 4 when the (`Orange`, `0`) image (*e.g.,* left bottom image) is given, due to the wrong priority, `color` which is an easier-to-learn attribute [48].

3 PGD: Per-sample Gradient-Norm-Based Debiasing

[2] Note that bias-alignment cannot always be strictly divisible in practice. For ease of explanation, we use the notations bias-conflicting/bias-aligned.

In this section, we propose a novel debiasing algorithm, coined as PGD. PGD consists of two models, biased f_b and debiased f_d with parameters θ_b and θ_d, respectively. Both models are trained sequentially. Obtaining the ultimately trained debiased model f_d involves three steps: (1) train the biased model, (2) compute the sampling probability of each sample, and (3) train the debiased model. These steps are described in Algorithm 1.

Algorithm 1 PGD: Per-sample Gradient-norm based Debiasing

```
Input: dataset $\mathcal{D}_n$, learning rate $\eta$, iterations $T_b$, $T_d$, Batch size $B$, Data augmentation operation $\mathcal{A}(\cdot)$, Initial parameter $\theta_0$, GCE parameter $\alpha$
    /** STEP 1: Train $f_b$ **/
for $t = 1, 2, \cdots, T_b$ do
    Construct a mini-batch $\mathcal{B}_t = \{(x_i, y_i)\}_{i=1}^{B} \sim U$.
    Update $\theta_t$ as:
    $\theta_{t-1} - \frac{\eta}{B}\nabla_\theta \sum_{(x,y)\in B_t} \mathcal{L}_{\text{GCE}}(\mathcal{A}(x), y; \theta_{t-1}, \alpha)$
end for
    /** STEP 2: Calculate $h$ **/
Calculate $h(x_i, y_i)$ for all $(x_i, y_i) \in \mathcal{D}_n$, equation 1.
    /** STEP 3: Train $f_d$ based on $h$ **/
for $t = 1, 2, \cdots, T_d$ do
    Construct a mini-batch $\mathcal{B}'_t = \{(x_i, y_i)\}_{i=1}^{B} \sim h$.
    Update $\theta_{T_b+t}$ as:
    $\theta_{T_b+t-1} - \frac{\eta}{B}\nabla_\theta \sum_{(x,y)\in\mathcal{B}'_t} \mathcal{L}_{\text{CE}}(\mathcal{A}(x), y; \theta_{T_b+t-1})$
end for
```

Step 1: Training the biased model. In the first step, the biased model is trained on the mini-batches sampled from a uniform distribution U, similar to conventional SGD-based training, with data augmentation $\mathcal{A}$. The role of the biased model is twofold: it detects which samples are bias-conflicting and calculates how much they should be highlighted. In doing so, the biased model f_b is trained on the generalized cross-entropy (GCE) loss $\mathcal{L}_{\text{GCE}}$ [48, 37]. For an input image x and the corresponding true class y, $\mathcal{L}_{\text{GCE}}$ is defined as $\mathcal{L}_{\text{GCE}}(x, y; \theta, \alpha) = \frac{1-f(y|x,\theta)^\alpha}{\alpha}$. Note that $\alpha \in (0, 1]$ is a hyperparameter that controls the degree of emphasizing the easy-to-learn samples, namely bias-aligned samples. We set $\alpha = 0.7$ as done by the authors of [68], [48] and [37].

Step 2: Compute the gradient-based sampling probability. In the second step, the sampling probability of each sample is computed from the trained biased model. Since rare samples have large gradient norm compared to the usual samples at the biased model [22], the sampling probability of each sample is computed to be proportional to its gradient norm so that bias-conflicting samples are over-sampled. We propose the following sampling probability of each sample $h(x_i, y_i)$ which is proportional to their gradient norm as follows:

$$h(x_i, y_i) = \frac{\|\nabla_\theta \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\|_s^r}{\sum_{(x_i,y_i)\in\mathcal{D}_n} \|\nabla_\theta \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\|_s^r}, \tag{1}$$

where $\|\cdot\|_s^r$ denotes r square of the L_s norm, and θ_b is the result of step 1. Note that computing the gradient for all samples requires huge computing resources and memory. Therefore, we only extract the gradient of the final FC layer parameters. This is a frequently used technique for reducing the computational complexity [6, 47, 28, 26, 27]. In other words, instead of $h(x_i, y_i)$, we empirically utilize $\hat{h}(x_i, y_i) = \frac{\|\nabla_{\theta_{\text{fc}}} \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\|_s^r}{\sum_{(x_i,y_i)\in\mathcal{D}_n} \|\nabla_{\theta_{\text{fc}}} \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\|_s^r}$, where θ_{fc} is the parameters of the final FC layer. We consider $r = 1$ and $s = 2$ (*i.e.,* L_2), and deliver ablation studies on various r and s in Appendix **??**.

Step 3: Ultimate debiased model training. Finally, the debiased model f_d is trained using mini-batches sampled with the probability $h(x_i, y_i)$ obtained in stage 2. However, [37] argued that just oversampling bias-conflicting samples does not successfully debias, and this unsatisfactory result stems from the data diversity, *i.e.,* data augmentation techniques are required. Hence, we used simple randomized augmentation operations $\mathcal{A}$ such as random rotation and random color jitter to oversample the bias-conflicting samples.

4 Experiments

In this section, we demonstrate the effectiveness of PGD for multiple benchmarks compared with previous proposed baselines. Detail analysis not in this section,*e.g.,* training time, unbiased case study, easier to learn target attribute, sampling probability analysis, reweighting with PGD are described in the Appendix D.

4.1 Benchmarks

To precisely examine the debiasing performance of PGD, we used the BFFHQ, BAR, CelebA, and CivilComments-WILDS datasets obtained from the real-world used to observe the situations in which general algorithms have poor performance due to bias attributes. Note that BFFHQ and BAR are biased by using human prior knowledge, while CelebA and CivilComments-WILDS are biased datasets by nature. A detailed explanation of each benchmark are presented in Appendix B.

4.2 Implementation.

Baselines. We select baselines available for the official code from the respective authors among debiasing methods without prior knowledge on the bias. We use eight methods on the various tasks: vanilla network, LfF [48], JTT [43][3], Disen [37], GEORGE [59], EIIL [14], BPA[54] and CNC [67].

Implementation details.

We use two types of networks: ResNet18 [21] and pretrained BERT. For BFFHQ, it uses ResNet18 as a backbone network, and exactly the same setting presented by Disen [37]. For CelebA, we follows experimental setting of [54] which uses ResNet18 as a backbone network. For CivilComments-WILDS, we utilize exactly the same hyperparameters of [43] and utilize pretrained BERT. Detail hyperparameters for CIFAR, BFFHQ, CelebA, CivilComments-WILDS are described in Appendix C. To reduce the computational complexity in extracting the per-sample gradients, we use only a fully connected layer, similar to [6, 47, 28, 26, 27].

4.3 Results

Table 1: Average test accuracy and standard deviation (three runs) for experiments with the raw image benchmarks: BAR and BFFHQ. The best accuracy is indicated in **bold** and for the overlapped best performance case is indicated in <u>Underline</u>.

Dataset	Vanilla	LfF	JTT	Disen	PGD (Ours)
BAR	63.15± 1.06	64.41± 1.30	63.62±1.33	<u>64.70± 2.06</u>	**65.39± 0.47**
BFFHQ	77.77± 0.45	82.13± 0.38	77.93± 2.16	82.77± 1.40	**84.20± 1.15**

Table 2: Average and worst test accuracy with the raw image benchmark: **CelebA** and raw NLP task: **CivilComments-WILDS**. The results of comparison algorithms for † and ‡ are the results reported in [54] and [67], respectively. The best worst accuracy is indicated in **bold**.

		Vanilla	LfF	GEORGE	BPA	EIIL	JTT	CNC	Ours
CelebA†	Avg.	80.52	84.89	83.13	90.18	-	-	-	89.27
	Worst	41.02	57.96	65.45	82.54	-	-	-	**82.73**
CivilComments‡	Avg.	92.1	92.5	-	-	90.5	91.1	81.7	92.1
	Worst	58.6	58.8	-	-	67.0	69.3	68.9	**70.6**

Similar to the results for the bias-feature-injected benchmarks, as shown in Table 4 and Table 2, PGD shows competitive performance among all the debiasing algorithms on the raw image benchmark (BAR, BFFHQ, and CelebA). For example, for the BFFHQ benchmark, the accuracy of PGD is 1.43% better than that of Disen. As in Table 2, PGD outperforms the other baselines on CivilComments-WILDs, much more realistic NLP task. Therefore, we believe PGD also works well with transformer, and it is applicable to the real-world.

4.4 Further analysis

To prove the superiority of PGD, we provide extensive analysis at the supplementary materials. For example, we provide empirical analysis about sampling probability of PGD D.2, Parameter sensitivity D.5, Reweighting using per-sample gradient D.8. Furthermore, we also provide theoretical evidence for understanding PGD at Appendix E.

5 Conclusion

We propose a gradient-norm-based dataset oversampling method for mitigating the dataset bias problem. The main intuition of this work is that gradients contains abundant information about each sample. Since the bias-conflicting samples are relatively more difficult-to-learn than bias-aligned samples, the bias-conflicting samples have a higher gradient norm compared with the others. Through various experiments and ablation studies, we demonstrate the effectiveness of our gradient-norm-based oversampling method, called PGD. We are still working on a future project: case where the given training dataset is corrupted, such as with noisy labels. We hope that this study will help improve understanding of researchers about the dataset bias problem.

[3] In the case of JTT [43], although the authors used bias label for validation dataset (especially, bias-conflicting samples), we tune the hyperparameters using a part of the biased training dataset for fair comparison. Considering that JTT does not show significant performance gain in the results, it is consistent with the existing results that the validation dataset is important in JTT, as described in [24].

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST), 10%] and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [No.2022-0-00641, XVoice: Multi-Modal Voice Meta Learning, 90%]

References

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning*, pages 60–69. PMLR, 2018.

[2] Faruk Ahmed, Yoshua Bengio, Harm van Seijen, and Aaron Courville. Systematic generalisation with group invariant predictions. In *International Conference on Learning Representations*, 2020.

[3] Mohsan Alvi, Andrew Zisserman, and Christoffer Nellåker. Turning a blind eye: Explicit removal of biases and variation from deep neural network embeddings. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[4] Jing An, Lexing Ying, and Yuhua Zhu. Why resampling outperforms reweighting for correcting sampling bias with stochastic gradients. In *International Conference on Learning Representations*, 2020.

[5] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[6] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

[7] 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*, pages 528–539. PMLR, 2020.

[8] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561, 2019.

[9] 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.

[10] Remi Cadene, Corentin Dancette, Matthieu Cord, Devi Parikh, et al. Rubi: Reducing unimodal biases for visual question answering. In *Advances in Neural Information Processing Systems*, pages 839–850, 2019.

[11] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

[12] Kamalika Chaudhuri, Sham Kakade, Praneeth Netrapalli, and Sujay Sanghavi. Convergence rates of active learning for maximum likelihood estimation, 2015.

[13] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4069–4082, 2019.

[14] Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In *International Conference on Machine Learning*, pages 2189–2200. PMLR, 2021.

[15] Luke Darlow, Stanisław Jastrzębski, and Amos Storkey. Latent adversarial debiasing: Mitigating collider bias in deep neural networks. *arXiv preprint arXiv:2011.11486*, 2020.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[17] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2018.

[18] Ankit Goyal, Kaiyu Yang, Dawei Yang, and Jia Deng. Rel3d: A minimally contrastive benchmark for grounding spatial relations in 3d. *Advances in Neural Information Processing Systems*, 33, 2020.

[19] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913, 2017.

[20] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[21] 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.

[22] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020.

[23] Rui Huang, Andrew Geng, and Yixuan Li. On the importance of gradients for detecting distributional shifts in the wild. *Advances in Neural Information Processing Systems*, 34, 2021.

[24] Badr Youbi Idrissi, Martin Arjovsky, Mohammad Pezeshki, and David Lopez-Paz. Simple data balancing achieves competitive worst-group-accuracy. In *Conference on Causal Learning and Reasoning*, pages 336–351. PMLR, 2022.

[25] Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[26] Krishnateja Killamsetty, S Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR, 2021.

[27] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. *arXiv preprint arXiv:2012.10630*, 2020.

[28] Krishnateja Killamsetty, Xujiang Zhao, Feng Chen, and Rishabh Iyer. Retrieve: Coreset selection for efficient and robust semi-supervised learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[29] 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 Conference on Computer Vision and Pattern Recognition*, pages 9012–9020, 2019.

[30] Eungyeup Kim, Jihyeon Lee, and Jaegul Choo. Biaswap: Removing dataset bias with bias-tailored swapping augmentation. *arXiv preprint arXiv:2108.10008*, 2021.

[31] Nayeong Kim, Sehyun Hwang, Sungsoo Ahn, Jaesik Park, and Suha Kwak. Learning debiased classifier with biased committee. *arXiv preprint arXiv:2206.10843*, 2022.

[32] Arvindkumar Krishnakumar, Viraj Prabhu, Sruthi Sudhakar, and Judy Hoffman. Udis: Unsupervised discovery of bias in deep visual recognition models. In *British Machine Vision Conference (BMVC)*, volume 1, page 3, 2021.

[33] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[34] Oran Lang, Yossi Gandelsman, Michal Yarom, Yoav Wald, Gal Elidan, Avinatan Hassidim, William T Freeman, Phillip Isola, Amir Globerson, Michal Irani, et al. Explaining in style: Training a gan to explain a classifier in stylespace. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 693–702, 2021.

[35] Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. Adversarial filters of dataset biases. In *International Conference on Machine Learning*, pages 1078–1088. PMLR, 2020.

[36] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[37] 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.

[38] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

[39] Erich L. Lehmann and George Casella. *Theory of Point Estimation*. Springer-Verlag, New York, NY, USA, second edition, 1998.

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

[41] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 513–528, 2018.

[42] Zhiheng Li and Chenliang Xu. Discover the unknown biased attribute of an image classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14970–14979, 2021.

[43] 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.

[44] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 3730–3738. IEEE Computer Society, 2015.

[45] Daniel McDuff, Shuang Ma, Yale Song, and Ashish Kapoor. Characterizing bias in classifiers using generative models. In *Advances in Neural Information Processing Systems*, pages 5404–5415, 2019.

[46] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018. cite arxiv:1802.03426Comment: Reference implementation available at http://github.com/lmcinnes/umap.

[47] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.

[48] Jun Hyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. In *34th Conference on Neural Information Processing Systems (NeurIPS) 2020*. Neural Information Processing Systems, 2020.

[49] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *arXiv preprint arXiv:2007.00653*, 2020.

[50] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. *Advances in neural information processing systems*, 30, 2017.

[51] Vikram V Ramaswamy, Sunnie SY Kim, and Olga Russakovsky. Fair attribute classification through latent space de-biasing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9301–9310, 2021.

[52] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[53] Roland Schäfer. On bias-free crawling and representative web corpora. In *Proceedings of the 10th web as corpus workshop*, pages 99–105, 2016.

[54] Seonguk Seo, Joon-Young Lee, and Bohyung Han. Unsupervised learning of debiased representations with pseudo-attributes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16742–16751, June 2022.

[55] Robert J. Serfling. *Approximation Theorems of Mathematical Statistics*. John Wiley & Sons, 1980.

[56] 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.

[57] Robik Shrestha, Kushal Kafle, and Christopher Kanan. An investigation of critical issues in bias mitigation techniques. *arXiv preprint arXiv:2104.00170*, 2021.

[58] Krishna Kumar Singh, Dhruv Mahajan, Kristen Grauman, Yong Jae Lee, Matt Feiszli, and Deepti Ghadiyaram. Don't judge an object by its context: Learning to overcome contextual bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11070–11078, 2020.

[59] Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *Advances in Neural Information Processing Systems*, 33:19339–19352, 2020.

[60] Jamshid Sourati, Murat Akcakaya, Todd K. Leen, Deniz Erdogmus, and Jennifer G. Dy. Asymptotic analysis of objectives based on fisher information in active learning, 2016.

[61] 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.

[62] 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. *arXiv preprint arXiv:2105.05612*, 2021.

[63] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[64] 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.

[65] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer, New York, 2004.

[66] Blake E. Woodworth, Suriya Gunasekar, Mesrob I. Ohannessian, and Nathan Srebro. Learning non-discriminatory predictors. In *COLT*, 2017.

[67] Michael Zhang, Nimit S Sohoni, Hongyang R Zhang, Chelsea Finn, and Christopher Ré. Correct-n-contrast: A contrastive approach for improving robustness to spurious correlations. *arXiv preprint arXiv:2203.01517*, 2022.

[68] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

– Appendix –
Mitigating Dataset Bias by Using Per-sample Gradient

A Related Work

Debiasing with bias label. In [19, 18], a debiased dataset was generated using human labor. Various studies [3, 29, 45, 58, 62] have attempted to reduce dataset bias using *explicit bias labels*. These studies [3, 29, 45, 58, 41, 40], used bias labels for each sample to reduce the influence of the bias labels when classifying target labels. Furthermore, [61] proposed the EnD regularizer, which entangles target correlated features and disentangles biased attributes. Several studies [3, 29, 62] have designed DNNs as a shared feature extractors and multiple classifiers. In contrast to the shared feature extractor methods, [45] and [51] fabricated a classifier and conditional generative adversarial networks, yielding test samples to determine whether the classifier was biased. Furthermore, [58] proposed a new overlap loss defined by a class activation map (CAM). The overlap loss reduces the overlapping parts of the CAM outputs of the two bias labels and target labels. The authors of [40, 41] employed bias labels to detect bias-conflicting samples and to oversample them to debias. In [43], a reconstructing method based on the sample accuracy was proposed. The authors of [43] used bias labels in the validation dataset to tune the hyper-parameters. On the other hand, there has been a focus on fairness within each attribute [20, 66, 50, 1]. Their goal is to prevent bias attributes from affecting the final decision of the trained model.

Debiasing with bias context. In contrast to studies assuming the explicit bias labels, a few studies [17, 64, 38, 7, 10, 13] assumed that the bias context is known. In [17, 64, 38], debiasing was performed by directly modifying known context bias. In particular, the authors of [17] empirically showed that CNNs trained on ImageNet [16] were biased towards the image texture, and they generated stylized ImageNet to mitigate the texture bias, while [38] and [64] inserted a filter in front of the models so that the influence of the backgrounds and colors of the images could be removed. On the other hand, some studies [7, 13, 10], mitigated bias by reweighting bias-conflicting samples: [7] used specific types of CNNs, such as BagNet [9], to capture the texture bias, and the bias was reduced using the Hilbert-Schmidt independence criterion (HSIC). In the visual question answering (VQA) task, [13] and [10] conducted debiasing using the entropy regularizer or sigmoid output of the biased model trained on the fact that the biased model was biased toward the question.

Debiasing without human supervision. Owing to the impractical assumption that bias information is given, recent studies have aimed to mitigate bias without human supervision [35, 48, 15, 30, 37]. [35] identified bias-conflicting samples by sorting the average accuracy of multiple train-test iterations and performed debiasing by training on the samples with low average accuracy. In [2], each class is divided into two clusters based on IRMv1 penalty [5] using the trained biased model, and train the debiased model so that the output of two clusters become similar. Furthermore, [30] used Swap Auto-Encoder [49] to generate bias-conflicting samples, and [15] proposed the modification of the latent representation to generate bias-conflicting samples by using an auto-encoder. [37] and [48] proposed a debiasing algorithm weighted training by using a relative difficulty score, which is measured by the per-sample training loss. Specifically, [37] used feature mixing techniques to enrich the dataset feature information. [54] and [59] proposed unsupervised clustering based debiasing method. Recently, contrastive learning based method [67] and self-supervised learning method [31] are proposed. On the other hand, there have been studies [42, 34, 32] that identify the bias attribute of the training dataset without human supervision.

B Benchmarks

B.1 Synthetic datasets (Additional synthetic datasets)

To precisely examine the debiasing performance of PGD, we additionally checked the Colored MNIST, Multi-bias MNIST, and Corrupted CIFAR datasets as synthetic datasets, which assume situations in which the model learns bias attributes first. We reconstruct MNIST variants, while the others are directly downloaded from the official repositories[4], [5] and run without any modification.

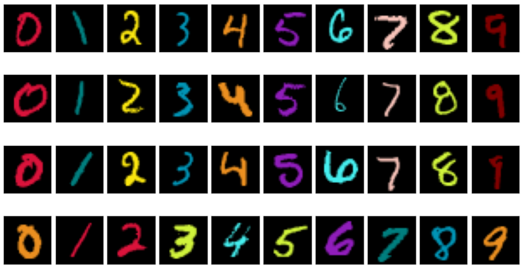


Figure 1: Colored MNIST: Single bias attribute, color, and target attribute shape. Top 3 rows represent bias-aligned samples, the other bottom row samples are bias-conflicting examples.

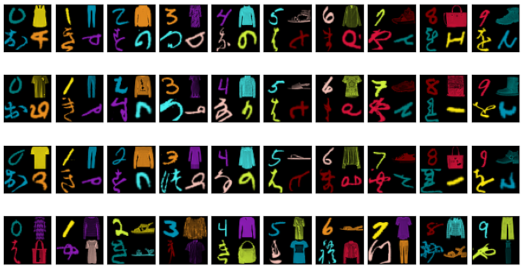


Figure 2: Biased MNIST: Multiple bias attributes, colors and objects, and target attribute shape. Top 3 rows represent bias-aligned samples, the other bottom row samples are bias-conflicting examples.

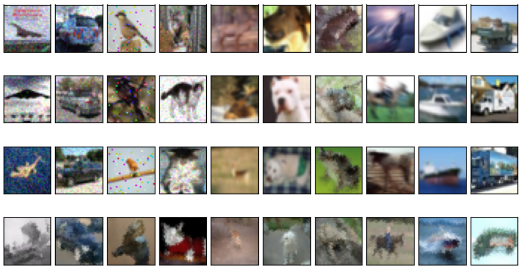


Figure 3: Corrupted CIFAR: single bias attribute, noise, and target attribute object. Top 3 rows represent bias-aligned samples, the other bottom row samples are bias-conflicting examples.

Colored MNIST The MNIST dataset [36] is composed of 1-dimensional gray hand-written images. The size of the image is 28×28. We inject color into these gray images to give them two main attributes, color and digit shape. This benchmark comes from related works [48, 30, 37, 7]. First, we choose ten uniformly sampled RGB colors from $\{C_i\}_{i\in[10]} \in \mathbb{R}^{3\times 10}$. Each sample (x, y) is colored by the following steps. (1) Select bias-confilicting or bias-aligned: random sample $u \sim \mathcal{U}(0, 1)$, and set the sample to bias-conflicting samples when $u < \rho$, otherwise as bias-aligned. Note that ρ is a ratio of bias-conflicting samples and we use $\{0.5\%, 1\%, 5\%\}$.
(2) Coloring: color the image with $c \sim \mathcal{N}(C_i, \sigma I)$, where $i \neq y$ for bias-conflict, or $i = y$ for bias-aligned. To generate color variation, we inject color for each sample with added noisy *i.e.,* $C_i + \mathcal{N}(0, 0.0001)$. We use $55,000$ samples for training, $5,000$ samples for validation (*i.e.,* 10%) and $10,000$ samples for test. Remark that test samples are unbiased test set, which means $\rho = 90\%$.

Multi-Bias MNIST The image size of Multi-bias MNIST is 56×56. This dataset aims to test the case where there are multiple biased attributes. To do so, we inject totally 7 bias attributes, {digit

[4] https://github.com/alinlab/BAR

[5] https://drive.google.com/drive/folders/1JEOqxrhU_IhkdcRohdbuEtFETUxfNmNT

color, object 1 (fashion), object 1 color, object 2 (Japanese character), object 2 color, object 3 (English character), object 3 color}, while the target attribute is digit shape. We inject each bias independently into each sample, as with the colored MNIST case (*i.e.,* sampling, and injecting bias). To generate unbiased test set, we also set $\rho = 90\%$ for all bias attributes. As same with Colored MNIST, we use $55,000$ samples for training, $5,000$ samples for validation and $10,000$ samples for test.

Corrupted CIFAR This dataset is generayted by injecting filters to the CIFAR10 dataset [33]. This benchmark is motivated by the works [48, 37]. In this benchmark, the biased attribute and the target attribute are object and corruption, respectively. Corruption examples are {Snow, Frost, Fog, Brightness, Contrast, Spatter, Elastic, JPEG, Pixelate, Saturate}. We download this benchmark from the official repository of [37]. This dataset is composed of $45,000$ training dataset and $5,000$ validation set, and $10,000$ for test. As same with prior datasets, test dataset is composed of unbiased samples.

B.2 Datasets in Section 4

We will explain the datasets utilized in Section 4.

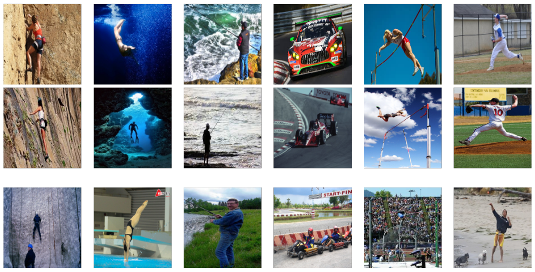


Figure 4: Biased Action Recognition: Target attribute: Action, while biased attribute is background. Top 2 rows represent bias-aligned samples. On the other hand, the bottom row represent bias-conflict samples.

Biased Action Recognition (BAR) This dataset comes from the paper [48] for real-world image test. The goal of this benchmark is classifying 6 actions {Climbing, Diving, Fishing, Racing, Throwing, Vaulting}, while the places are biased. Target and bias attributes pairs are (Climbing,RockWall), (Diving,Underwater), (Fishing, WaterSurface), (Racing, APavedTrack), (Throwing, PlayingField), and (Vaulting, Sky). Bias-conflict samples, for example, is (Climbing, IceCliff), (Diving, Indoor), (Fishing, Forest), (Racing, OnIce), (Throwing, Cave), (Vaulting, Beach). The number of samples for training is $1,941$, and test is $6,54$. To split the training and validation samples, we use 10% validation samples, *i.e.,* $1,746$ images for training and 195 for validation. We download training dataset from the official repository.

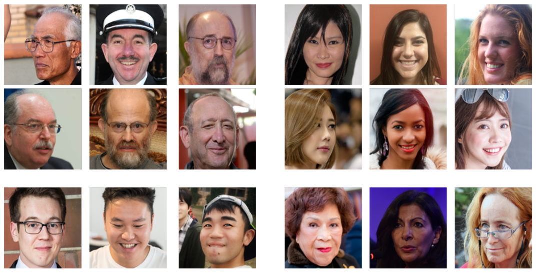


Figure 5: BFFHQ: Biased FFHQ. Target attribute: Gender, Biased attribute: age. Top 2 rows represent bias-aligned samples, oppositely, bottom row represent bias-conflict samples.

Biased FFHQ This BFFHQ benchmark is conducted in [37, 30]. Target and biased attributes for bias-aligned samples are (Female, Young), and (Male, Old). Specifically, 'Young' represents, age ranging 10 to 29, and 'old' are from 40 to 59. Oppositely, the bias-conflict samples are (Female, Old) and (Male, Young). The number of training samples are $19,200$, validation samples are $1,000$, and test samples are $1,000$.

CelebA CelebA [44] is a common real world face classification dataset and each image has 40 attributes. The goal is classifying the hair color(“blond" and “not blond") of celebrities which has a spurious correlation with the gender (“male" or “female") attribute. In fact, only 6% of blond faces are male, so ERM shows poor performance on bias conflict subclass. We report the average accuracy and the worst-group accuracy on the test dataset.

CivilComments-WILDS CivilComments-WILDS [8] is a dataset to classify whether an online comment is toxic or non-toxic. Each data is a real online comment, curated on the Civil Comments platform that is a comment plug-in for independent news sites. The mentions of certain demographic identities (male, female, White, Black, LGBTQ, Muslim, Christian, and other religion) cause the supurious correlation with the label. Table 1 indicates the portion of toxic comments for each demographic identities [54].

Identity	male	female	White	Black	LGBTQ	Muslim	Christian	other religions
portion(%) of toxic	14.9	13.7	28.0	31.4	26.9	22.4	9.1	15.3

Table 1: portion of toxic comments in the CivilComments-Wilds for each demographic identity

C Experiment details

C.1 Settings

Architecture. For the colored MNIST, and Multi-Bias MNIST datasets, we use simple convolutional networks consisting of three CNN layers with kernel size 4, and channel size $\{8, 32, 64\}$ for each layer. Also, we utilize average pooling at the end of each layer. Batch normalization and dropout techniques are used for regularization. Detailed network configurations are below. For corrupted CIFAR, BAR, and BFFHQ, we utilize ResNet-18 which is provided by the open source library, `torchvision`. For CelebA, we follows experimental setting of [54] which uses ResNet18 as a backbone network. For CivilComments-WILDS, we utilize exactly the same hyperparameters of [43] and utilize pretrained BERT.

SimConv-1.

```
(conv1): Conv2d(3, 8, kernel_size=(4, 4), stride=(1, 1))
(bn1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(relu1): ReLU()
(dropout1): Dropout(p=0.5, inplace=False)
(avgpool1): AvgPool2d(kernel_size=2, stride=2, padding=0)
(conv2): Conv2d(8, 32, kernel_size=(4, 4), stride=(1, 1))
(bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(relu2): ReLU()
(dropout2): Dropout(p=0.5, inplace=False)
(avgpool2): AvgPool2d(kernel_size=2, stride=2, padding=0)
(conv3): Conv2d(32, 64, kernel_size=(4, 4), stride=(1, 1))
(relu3): ReLU()
(bn3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(dropout3): Dropout(p=0.5, inplace=False)
(avgpool3): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=64, out_features=$num_class, bias=True)
```

SimConv-2.

```
(conv1): Conv2d(3, 8, kernel_size=(7, 7), stride=(1, 1))
(bn1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(relu1): ReLU()
(dropout1): Dropout(p=0.5, inplace=False)
(avgpool1): AvgPool2d(kernel_size=3, stride=3, padding=0)
```

```
(conv2): Conv2d(8, 32, kernel_size=(7, 7), stride=(1, 1))
(bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(relu2): ReLU()
(dropout2): Dropout(p=0.5, inplace=False)
(avgpool2): AvgPool2d(kernel_size=3, stride=3, padding=0)
(conv3): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
(relu3): ReLU()
(bn3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(dropout3): Dropout(p=0.5, inplace=False)
(conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
(relu4): ReLU()
(bn4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(dropout4): Dropout(p=0.5, inplace=False)
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=128, out_features=$num_class, bias=True)
```

C.2 Baselines

(1) LfF [48] trains the debiased model by weighting the bias-conflicting samples based on the "relative difficulty" which is computed by the two loss values from the biased model and debiased model. To emphasize bias-conflict samples, the authors utilize generalized cross entropy loss with parameter $\alpha = 0.7$. We implement the LfF algorithm following the official code offered by the authors. Loss function that this work proposed are as follows. Details are in the original paper.

$$\mathcal{L}_{\text{LfF}} = W(z)\mathcal{L}_{\text{CE}}(C_d(z,y) + \lambda\mathcal{L}_{\text{GCE}}(C_b(z,y)),$$
$$W(z) = \frac{\mathcal{L}_{\text{CE}}(C_b(z),y)}{\mathcal{L}_{\text{CE}}(C_b(z),y) + \mathcal{L}_{\text{CE}}(C_d(z),y)}.$$

Note that $W(z)$ is a relative difficulty and that GCE is a generalized cross-entorpy. z denotes feature, which is the output of the penultimate layer, and $C_{\cdot}$ is fully connected layer.

(2) JTT [43] aims to debiasing by spliting dataset into correctly learned and failed to learned samples. To do so, JTT trains the biased model first and split the given training dataset as follows:

$$\mathcal{D}_{\text{error-set}} = \{(x,y) \text{ s.t. } y_{\text{given}} \neq \arg\max_{c} f_b(x)[c]\}, \tag{1}$$

Then, train the ultimate debiased model by oversampling $\mathcal{D}_{\text{error-set}}$ with λ_{up} times. We set λ_{up} as $1./\rho$ for all experiments. We reproduce the results by utilizing the official code offered by the authors. The main strength of PGD compared to JTT is that PGD does not need to set a hyperparameter.

(3) Disen [37] aims to debias by generating abundant features from mixing features between samples. To do so, the authorus train the biased and debiased model by aggregating features from both networks. This work also utilized "relative difficulty" that is proposed in LfF [48]. We reproduce the results utilizing the official code offered by the authors. The loss function proposed in this work is as follows. Details are provided in the original paper.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{dis}} + \lambda_{\text{swap}}\mathcal{L}_{\text{swap}},$$

where

$$\mathcal{L}_{\text{swap}} = W(z)\mathcal{L}_{\text{CE}}(C_d(z_{\text{swap}},y) + \lambda_{\text{swap}_b}\mathcal{L}_{\text{GCE}}(C_b(z_{\text{swap}},\tilde{y}))$$
$$\mathcal{L}_{\text{dis}} = W(z)\mathcal{L}_{\text{CE}}(C_d(z,y) + \lambda_{\text{dis}}\mathcal{L}_{\text{GCE}}(C_b(z,y)),$$
$$W(z) = \frac{\mathcal{L}_{\text{CE}}(C_b(z),y)}{\mathcal{L}_{\text{CE}}(C_b(z),y) + \mathcal{L}_{\text{CE}}(C_d(z),y)}.$$

All terms are identical with LfF paper, except for swapped feature z_{swap}.

(4) GEORGE [59] aims to debias by measuring and mitigating hidden stratification without requiring access to subclass labels. Suppose there are given n datapoints, $x_1, \cdots, x_n \in \chi$ and associated

superclass (target) labels $y_1, \cdots, y_n \in \{1, \cdots, C\}$. In addition, associated with each datapoint x_i has latent (unobserved) subclass label $z_i \in \{1, \cdots, K\}$.
GEORGE consists of 3 steps. To do so, the authorus train the biased based on ERM first. Next, to estimate approximate subclass (latent) label, apply UMAP dimensionality reduction [46] to the features of given training dataset at ERM model and cluster the output of the reduced dimension for the data of each superclass into K clusters, where K is chosen automatically. The detail description of clustering process is provided in the original paper. Lastly, to improve performance on these estimated subclass, they minimize the maximum per-cluster average loss(i.e. $(x, y) \sim \hat{P}_{\tilde{z}}$), by using the clusters as groups in the GDRO objective [52]. The loss function proposed in this work is as follows:

$$\underset{L, f_\theta}{\text{minimize}} \max_{1 \leq \tilde{z} \leq K} \mathbb{E}_{(x,y) \sim \hat{P}_{\tilde{z}}}[l(L \circ f_\theta(x), y)]$$

where f_θ and L are parameterized feature extractor and classifier, respectively.

(5) BPA[54] aims to debias by using the technique of feature clustering and cluster reweighting. It consists of 3 steps. To do so, the authorus train the biased model $\tilde{\theta}$ based on ERM first. Next, at the biased model $\tilde{\theta}$, cluster all training examples into K clusters based on the feature derived from the ERM model $\tilde{\theta}$, where K is hyperparameter. Here, $h(x, y; \tilde{\theta}) \in \mathcal{K} = \{1, \cdots, K\}$ denote the cluster mapping function of data (x, y) derived by $\tilde{\theta}$. At the last step, they calculate the proper importance weight, w_k, to the k-th cluster, whekre $k \in \mathcal{K}$ and the final objective of debiasing framework is given by minimizing a weighted empirical risk as follows:

$$\underset{\theta}{\text{minimize}} \left\{ \mathbb{E}_{(x,y) \sim P} \left[w_{h(x,y;\tilde{\theta})}(\theta) l(x, y; \theta) \right] \right\},$$

where $w_{h(x,y;\tilde{\theta})}(\theta)$ denote the importance weight to the cluster $h(x, y; \tilde{\theta})$ at the model θ. Concretely, for arbitrary iteration index T, $w_{h(x,y;\tilde{\theta})}(\theta_T)$ is derived from the momentum method based on the history set $\mathcal{H}_T$, defined by

$$\mathcal{H}_T = \left\{ 1 \leq t \leq T \mid \frac{\mathbb{E}_{(x,y) \sim P_k} \left[l((x, y); \theta_t) \right]}{N_k} \right\},$$

where N_k is number of the data belong to k-th cluster. Details are provided in the original paper.

(6) CNC [67] aims to debias by learning representations such that samples in the same class but different groups are close to each other. CNC is composed of 2 steps: (1) Inferring pseudo group labels, (2) Supervised contrastive learning. First, get ERM based model $f_{\hat{\theta}}$ and get pseudo prediction $\hat{y}$, standard argmax over the final-layer outputs of $f_{\hat{\theta}}$. Next, training the debiased model based on supervised contrastive learning using pseudo prediction $\hat{y}$. The detail process of contrastive learning for each iteration is like below:

- From the selected batch, sample the one anchor data (x,y).
- Construct the set of positives samples $\{(x_m^+, y_m^+)\}$ which is belong to the batch, satisfying $y_m^+ = y$ and $\hat{y}_m^+ \neq \hat{y}$.
- Similarly, construct the set of negative samples $\{(x_n^-, y_n^-)\}$ which is belong to the batch, satisfying $y_n^- \neq y$ and $\hat{y}_n^- = \hat{y}$.
- With loss of generality, assume cardinality of positive set and negative set are M and N, respectively.
- Weight update based on the gradient of the loss function $\hat{L}(f_\theta; x, y)$, the detail is like below:

  $$\hat{L}(f_\theta; x, y) = \lambda \hat{L}_{\text{con}}^{\text{sup}}(x, \{x_m^+\}_{m=1}^M, \{x_n^-\}_{n=1}^N; f_{\text{enc}}) + (1 - \lambda) \hat{L}_{\text{cross}}(f_\theta; x, y).$$

  Here, $\lambda \in [0, 1]$ is a hyperparameter and $\hat{L}_{\text{cross}}(f_\theta; x, y)$ is an average cross-entropy loss over x, the M positives, and N negatives. Moreover, f_{enc} is the feature extractor part of f_θ and the detail formulation of $\hat{L}_{\text{con}}^{\text{sup}}(x, \{x_m^+\}_{m=1}^M, \{x_n^-\}_{n=1}^N; f_{\text{enc}})$ is like below:

$$-\frac{1}{M} \sum_{r=1}^{M} \log \frac{\exp(f_{\text{enc}}(x)^T f_{\text{enc}}(x_r^+)/\tau)}{\sum_{m=1}^{M} \exp(f_{\text{enc}}(x)^T f_{\text{enc}}(x_m^+)/\tau) + \sum_{n=1}^{N} \exp(f_{\text{enc}}(x)^T f_{\text{enc}}(x_n^-)/\tau)}.$$

(7) EIIL [14] proposes a novel invariant learning framework that does not require a prior environment knowledge. EIIL is composed of 3-step process: (1) ERM training (2) Environment inference (EI) (3) Invariant Learning (IL).
First, get a biased model $w \circ \Phi$ by minimizing ERM. Note that Φ and w are feature extractor and classifier, respectively.
Next, based on feature extractor Φ, optimize the EI objective to infer environments of each training data. The object of EI is sorts training examples $\mathcal{D}$ into $\mathcal{D}_1 \cup \mathcal{D}_2$ that maximally separate the spurious features so that facilitate effective invariant learning. Concretely, get a $q^\star = \text{argmax}_q \; \|\nabla_{\tilde{w}=1} \tilde{R}^e(\tilde{w} \circ \Phi)\|$, where $\tilde{R}^e(\tilde{w} \circ \Phi) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} q_i(e) l(\tilde{w} \circ \Phi(x_i), y_i)$. Note that $e \in \{1, 2\}$ and $q_i(1) + q_i(2) = 1$ for all $i \in \{1, \cdots, |\mathcal{D}|\}$. Based on $q^\star$, get a bernoulli sample $\hat{q} \sim \text{Bernoulli}(q^\star)$, and split $\mathcal{D}$ in to D_1 and D_2 based on result of binary value $\hat{q}$.
Lastly, to get a debiased model, optimize the classifier w and feature extractor Φ by minimizing invariant learning objective (e.g. IRM or GroupDRO) from the gained two pseudo environment $\mathcal{D}_1$ and $\mathcal{D}_2$.

C.3 Implementation Details

Image processing We train and evaluate with a fixed image size. For colored MNIST case, 28×28 and Multi-bias MNIST , 56×56, Corrupted CIFAR, 32×32, and real-world datasets of 224×224. For all cases, we utilize random rotation and color jitter to avoid overfitting. We use normalizing with mean $0.4914, 0.4822, 0.4465)$, and standard deviation $(0.2023, 0.1994, 0.2010)$.

Implementation For Table 3 and Table 4 reported in Section 4, we reproduce all experimental results referring to other official repositories: [6] [7] [8] [9].

Except for JTT, all hyperparameters for corrupted CIFAR, BAR, and BFFHQ follow previously reported parameters in repositories. We grid-search for other cases, MNIST variants. We set the only hyperparameter of PGD, q, as 0.7, which is proposed by the original paper [68]. A summary of the hyperparameters that we used is reported in Table 2.

	Colored MNIST	Multi-bias MNIST	Corrupted CIFAR	BAR	Biased FFHQ	CelebA	CivilComments-WILDS
Optimizer	SGD	SGD	Adam	SGD	Adam	Adam	SGD
Batch size	128	32	256	16	64	256	16
Learning rate	0.02	0.01	0.001	0.0005	0.0001	0.0001	0.00001
Weight decay	0.001	0.0001	0.001	1e-5	0.0	0.01	0.01
Momentum	0.9	0.9	-	0.9	-	-	0.9
Lr decay	0.1	0.1	0.5	0.1	0.1	Cosine annealing	0.1
Decay step	40	-	40	20	32	-	-
Epoch	100	100	200	100	160	100	5
GCE α	0.7	0.7	0.7	0.7	0.7	0.7	0.7

Table 2: Hyperparameter details

For Table 2 reported in Section 4, we follow the implementation settings of CelebA and CivilComments-WILDS, suggested by [54] and [43], respectively. A summary of the hyperparameters that we used is reported in Table 2. We conduct our experiments mainly using a single Titan XP GPU for all cases.

[6] https://github.com/alinlab/LfF

[7] https://github.com/clovaai/rebias

[8] https://github.com/kakaoenterprise/Learning-Debiased-Disentangled

[9] https://github.com/anniesch/jtt

D Additional experiments

D.1 Unbiased test accuracy on synthetic datasets

Table 3: Average test accuracy and standard deviation (three runs) for experiments with the MNIST variants under various bias conflict ratios. The best accuracy is indicated in **bold** for each case.

Dataset	ρ	Vanilla	LfF	JTT	Disen	PGD (Ours)
CMNIST	0.5%	60.94± 0.97	91.35± 1.83	85.84± 1.32	94.56± 0.57	**96.88± 0.28**
	1%	79.13± 0.73	96.88± 0.20	95.07± 3.42	96.87± 0.64	**98.35± 0.12**
	5%	95.12± 0.24	98.18± 0.05	96.56± 1.23	98.35± 0.20	**98.62± 0.14**
MBMNIST	10%	25.23± 1.16	19.18± 4.45	25.34± 1.45	25.75± 5.38	**61.38± 4.41**
	20%	62.06± 2.45	65.72± 6.23	68.02± 3.23	61.62± 2.60	**89.09± 0.97**
	30%	87.61± 1.60	89.89± 1.76	85.44± 3.44	88.36± 2.06	**90.76± 1.84**
CCIFAR	0.5%	23.06± 1.25	28.83± 1.30	25.34± 1.00	29.96± 0.71	**30.15± 1.22**
	1%	25.94± 0.54	33.33± 0.15	33.62± 1.05	36.35± 1.69	**42.02± 0.73**
	5%	39.31± 0.66	50.24± 1.41	45.13± 3.11	51.19± 1.38	**52.43± 0.14**

Table 4: Average test accuracy and standard deviation (three runs) for experiments with the raw image benchmarks: BAR and BFFHQ. The best accuracy is indicated in **bold** and for the overlapped best performance case is indicated in Underline.

Dataset	Vanilla	LfF	JTT	Disen	PGD (Ours)
BAR	63.15± 1.06	64.41± 1.30	63.62±1.33	64.70± 2.06	**65.39± 0.47**
BFFHQ	77.77± 0.45	82.13± 0.38	77.93± 2.16	82.77± 1.40	**84.20± 1.15**

Accuracy results. In Table 3, we present the comparisons of the image classification accuracy for the unbiased test sets. The proposed method outperforms the baseline methods for all benchmarks and for all different ratios. For example, our model performance is 35.94% better than that of the vanilla model for the colored MNIST benchmark with a ratio $\rho = 0.5\%$. For the same settings, PGD performs better than Disen by 2.32%.

As pointed out in [57], colored MNIST is too simple to evaluate debiasing performance on the basis of the performance of baselines. In Multi-bias MNIST case, other models fail to obtain higher unbiased test results, even though the ratio is high, *e.g.,* 10%. In this complex setting, PGD shows superior performance over other methods. For example, its performance is higher by 36.15% and 35.63% compared with the performance of vanilla model and Disen for the ratio of 10%.

Similar to the results for the bias-feature-injected benchmarks, as shown in Table 4 and Table 2, PGD shows competitive performance among all the debiasing algorithms on the raw image benchmark (BAR and BFFHQ). For example, for the BFFHQ benchmark, the accuracy of PGD is 1.43% better than that of Disen.

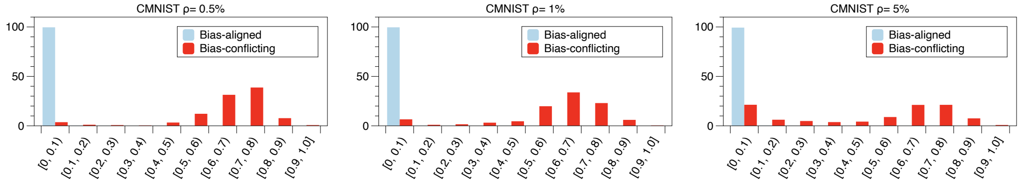
D.2 Correlation between gradient norm and bias-alignment of the CMNIST

To check if the per-sample gradient norm efficiently separates the bias-conflicting samples from the bias-aligned samples, we plot the gradient norm distributions of the colored MNIST (CMNIST). For comparison, we normalized the per-sample gradient norm as follows: $\frac{\|\nabla_\theta \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\|}{\max_{(x_i, y_i) \in \mathcal{D}_n} \|\nabla_\theta \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\|}$. As in Figure 6, the bias-aligned sample has a lower gradient norm (blue bars) than the bias-conflicting samples (red bars).

D.3 PGD does not learn only the second-easiest feature

We provide the results of the following experimental setting: the target feature is color and the bias feature is digit shape, i.e., the task is to classify the color, not the digit shape. Let us give an example of this task. When one of the target classes is Red, this class is aligned with one of the digits (e.g., “0”). In other words, bias-aligned samples in this class are (Red, “0”), and the bias-conflicting samples are (e.g., (Red, “1”), (Red, “2”), ..., (Red,”9”)).

Note that, as shown in LfF [48], color is empirically known to be easier to learn than digit shape, we think that the above scenario reflects the concern: whether PGD is only targeting to learn the second-easiest feature (digit shape). Therefore, if the concern is correct, PGD may fail in this Color



(a) Colored MNIST $\rho = 0.5\%$ (b) Colored MNIST $\rho = 1\%$ (c) Colored MNIST $\rho = 5\%$

Figure 6: Histogram of per-sample gradient norm.

target MNIST scenario since the model will learn digit shape. However, as shown in the table below, vanilla, PGD, and LfF perform well in that case.

	Vanilla (Digit)	**Vanilla (Color)**	LfF (Digit)	**LfF (Color)**	PGD (Digit)	**PGD (Color)**
$\rho = 0.5\%$	60.94	**90.33**	91.35	**91.16**	96.88	**98.92**
$\rho = 1\%$	79.13	**92.53**	96.88	**96.12**	98.35	**99.58**
$\rho = 5\%$	95.12	**96.96**	98.18	**99.11**	98.62	**99.7**

Table 5: Digit target MNIST vs Color target MNIST

We can also support this result by seeing the distribution of the normalized gradient norms, $\|\nabla_\theta \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\| / \max_{(x_i,y_i)\in\mathcal{D}_n} \|\nabla_\theta \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\| \in [0, 1]$, extracted from the biased model θ_b (trained in the Step 1 in Algorithm 1 of the Section 3). If PGD aims to learn the second-easiest feature (digit shape), it will highlight bias-conflicting samples that have abundant digit shapes, so that the ultimate debiased model can consider them important.

	[0.0,0.1)	[0.1,0.2)	[0.2,0.3)	[0.3,0.4)	[0.4,0.5)	[0.5,0.6)	[0.6,0.7)	[0.7,0.8)	[0.8,0.9)	[0.9,1.0]
Bias-aligned	53504	88	40	21	13	**18**	**16**	**17**	**8**	**4**
Bias-conflicting	212	18	7	10	8	**6**	**5**	**4**	**0**	**1**

Table 6: Number of samples at each bin: Color target MNIST ($\rho = 0.5\%$)

The numbers filled in the Table 6 is the number of data belonging to each bin category. We can check that *there are no bias-conflicting samples whose gradient norm is significantly larger than the bias-aligned samples.* In other words, PGD *does not force the debiased model to learn the digit shape (i.e., the second-easiest feature) in this scenario.* This scenario brings the similar performance comparing to Vanilla.

D.4 PGD on unbiased datasets

Vanilla	LfF	Disen	Ours
$\rho = 90\%$ Colored MNIST			
99.04	98.75	99.31	98.43
Natrual CIFAR10			
94.24	-	-	94.79

Table 7: Results on unbiased colored MNIST and natural CIFAR10 cases.

In addition, we check whether PGD fails in the unbiased and conventional public datasets. In this case, target is the first-easiest feature because no specific attribute makes bias. To verify this, we report two types of additional results: (1) unbiased colored MNIST ($\rho = 90\%$), and (2) conventional public datasets, *i.e.,* CIFAR 10. We follow experimental settings of colored MNIST for the unbiased colored MNIST. On the other hand, we train ResNet18 [21] for CIFAR 10 with the SGD optimizer, 0.9 momentum, $5e-4$ weight decay, 0.1 learning rate, CosineAnnealing LR decay scheduler. As in Table 7, PGD does not suffer a significant performance degradation in unbiased Colored MNIST. Furthermore, it performs better than the vanilla model on the CIFAR 10 dataset. This means that the training distribution that PGD changes does not cause significant performance degradation. In other words, *PGD can only balance the training data set, regardless of whether the training dataset is balanced or unbalanced.*

D.5 Ablation study on GCE parameter α

Colored MNIST	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$
		Debiased model		
$\rho = 0.5\%$	89.93	94.70	96.88	96.79
$\rho = 1\%$	96.32	97.27	97.35	97.59
$\rho = 5\%$	98.80	98.82	98.62	98.78
		Biased model		
$\rho = 0.5\%$	19.86	18.70	18.12	17.39
$\rho = 1\%$	22.40	21.04	19.71	19.12
$\rho = 5\%$	53.24	49.46	43.97	39.40

Table 8: Ablation study on GCE parameter α.

The only hyper-parameter used in PGD is the GCE parameter α. We experimented with this value at 0.7 according to the protocoal of LfF [48]. However, we need to compare the various cases α on the colored MNIST benchmark for analysis. As in Table 8, when the GCE parameter is 0.9, the debiased model performs best. This is because the biased model is fully focused on the bias feature, rather than seeing the target feature, which can be seen from the unbiased test accuracy of the biased model (see the bottom of Table 8).

D.6 Computation cost

	Vanilla	LfF	Disen	ours
Computation time	14m 59s	21m 35s	23m 18s	33m 31s
	Step 1	Step 2	Step 3	
Computation time	15m 19s	1m 26s	16m 46s	

Table 9: Computation cost

Debiasing requires more modules, which increases the computational cost. The training time for the colosed MNIST $\rho = 0.5\%$ is reported in Table 9. As in the top of Table 9, we report the computational cost of four algorithms, vanilla, LfF, Disen, and PGD. Here, PGD spends a longer training time. To check which parts generate longer time, we measure part-by-part cost, at the bottom of Table 9. Note that Steps 1, 2, and 3 represent training the biased model, computing per-sample gradient norm, and training the debiased model, respectively. We can conclude the following two facts. (1) Step 2 (computing per-sample gradient norm) spends 4.3% of training time. (2) Resampling requires an additional cost of 1 m 27s.

D.7 Multi-stage vs Single-stage

	Training time		Test Acc.	
Colored MNIST	Single-stage	Multi-stage	Single-stage	Multi-stage
$\rho = 0.5\%$	2h 53m 40s	33m 39s	92.39	96.88
$\rho = 1\%$	2h 54m 45s	32m 28s	97.10	97.35
$\rho = 5\%$	2h 49m 31s	34m 13s	98.51	98.62

Table 10: Multi-stage versus Single-stage

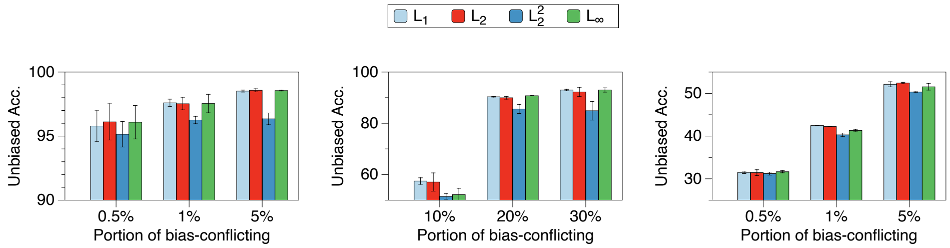
PGD computes per-sample gradient only once, between training the biased model and the debiased model. However, an update of the per-sample gradient can be performed repeatedly at each epoch (*i.e.,* Single-stage). In other words, the PGD can be modeled to run the following loop: updating the biased model → updating the sampling probability → updating the debiased model. In this section, we justify why we compute per-sample gradient at once (*i.e.,* Multi-stage). We report the performance of Multi-stage and Single-stage of PGD on the colored MNIST dataset. As in Table 10, the single-stage method has two characteristics: (1) It takes a long training time compared to the multi-stage method. (2) It has lower unbiased test accuracy compared to the multi-stage method. The reason why it takes longer training time is that computing the per-sample gradient norm requires huge computation resources. On the other hand, the sampling probability of the single-stage method changes training distribution over epochs, the debiased model suffers unstable training, so that it loses the debiasing performance.

D.8 Resampling versus Reweighting

Colored MNIST	Resampling	Reweighting
$\rho = 0.5\%$	96.88	94.70
$\rho = 1\%$	97.35	97.20
$\rho = 5\%$	98.62	98.51

Table 11: Reweighting vs resampling

To support our algorithm design, we provide further experimental analysis, *i.e.,* resampling versus reweighting. Reweighting [48, 37] and resampling [43] are the two main techniques to debias by up-weighting bias-conflicting samples. PGD is an algorithm that modifies the sampling probability by using the per-samplge gradient norm. To check whether PGD works with reweighting, we examine the results of PGD with reweighting on colored MNIST datasets and report in Table 11. We compute the weight for each sample as follows: $w(x_i, y_i) = |\mathcal{D}_n| \times \frac{\|\nabla_\theta \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\|}{\sum_{(x_i, y_i) \in \mathcal{D}_n} \|\nabla_\theta \mathcal{L}_{\text{CE}}(x_i, y_i; \theta_b)\|}$. As in Table 11, PGD with resampling slightly outperforms PGD with reweighting. As argued in [4], his gain from resampling can be understood by the arguments that resampling is more stable and better than reweighting.



(a) Colored MNIST (b) Multi-bias MNIST (c) Corrupted CIFAR

Figure 7: Average PGD results for various of norms, $\{L_1, L_2, L_2^2, L_\infty\}$, for the feature-injected benchmarks. The error bars represent the standard deviation of three independent trials.

D.9 Unbiased test accuracy on various norms.

Since, gradient norm can have various candidates, such as order of the norm, we report four configurations of gradient norms. As shown in Figure 7, all norms have significant unbiased test performance. Amongst them, the L_2-norm square case shows lower unbiased performance than the other cases. Therefore, it is recommended that any first power of $L_{\{1,2,\infty\}}$-norms be used in PGD for overcoming the dataset bias problem. This is quite different from the results with [23], which suggested that L_1-norm is the best choice in the research field of out-of-distribution detection.

D.10 Ablation study

Table 12 shows the importance of each module in our method: generalized cross entropy, and data augmentation modules. We set the ratio ρ as 0.5% and 10% for colored MNIST and multi-bias MNIST, respectively. We observe that GCE is more important that data augmentation for colored MNIST. However, data augmentation shows better performance than GCE for multi-bias MNIST. For all cases, the case where both are utilized outperforms the other cases.

Table 12: Ablation studies on GCE and data augmentation (✓for applied case).

GCE	Aug.	Colored (0.5%)	Multi-bias (10%)
		84.93 ± 0.79	40.58 ± 3.39
✓		93.18 ± 1.07	45.70 ± 2.91
	✓	91.19 ± 0.97	46.70 ± 1.10
✓	✓	**96.88** ± **0.28**	**61.38** ± **4.41**

E Mathematical Understanding of PGD

This section provides a theoretical analysis of per-sample gradient-norm based debiasing. We first briefly summarize the maximum likelihood estimator (MLE) and Fisher information (FI) which are ingredients of this section. We then interpret the debiasing problem as a min-max problem and deduce that solving the min-max problem can be phrased as minimizing the trace of the inverse FI. Since handling the trace of the inverse FI is very difficult owing to its inverse computation, we look at a glance by relaxing it into a one-dimensional toy example. In the end, we conclude that the gradient-norm based re-sampling method is an attempt to solve the dataset bias problem.

E.1 Preliminary

Training and test joint distributions. The general joint distribution $\mathcal{P}(x, y|\theta)$ is assumed to be factored into the parameterized conditional distribution $f(y|x, \theta)$ and the marginal distribution $\mathcal{P}(x)$, which is independent of the model parameter θ, *i.e.,* $\mathcal{P}(x, y|\theta) = \mathcal{P}(x)f(y|x, \theta)$. We refer to the model $f(y|x, \theta^\star)$ that produces the exact correct answer, as an oracle model, and to its parameter $\theta^\star$ as the oracle parameter. The training dataset $\mathcal{D}_n$ is sampled from $\{(x_i, y_i)\}_{i=1}^n \sim p(x)f(y|x, \theta^\star)$, where the training and test marginal distributions are denoted by $p(x)$ and $q(x)$, respectively. Here, we assume that both marginal distributions are defined on the marginal distribution space $\mathcal{M} = \{\mathcal{P}(x) | \int_{x\in\mathcal{X}} \mathcal{P}(x)\, dx = 1\}$, where $\mathcal{X}$ means the input data space, *i.e.,* $p(x),\ q(x) \in \mathcal{M}$.

The space $\mathcal{H}$ of sampling probability h. When the training dataset $\mathcal{D}_n$ is given, we denote the sampling probability as $h(x)$ which is defined on the probability space $\mathcal{H}$[10]:

$$\mathcal{H} = \{h(x)\,|\, \textstyle\sum_{(x_i,y_i)\in\mathcal{D}_n} h(x_i) = 1\,,\ h(x_i) \geq 0 \quad \forall (x_i, y_i) \in \mathcal{D}_n\}. \tag{2}$$

Maximum likelihood estimator (MLE). When $h(x)$ is the sampling probability, we define MLE $\hat{\theta}_{h(x),\mathcal{D}_n}$ as follows:

$$\hat{\theta}_{h(x),\mathcal{D}_n} = \arg\min{}_\theta \ -\textstyle\sum_{(x_i,y_i)\in\mathcal{D}_n} h(x_i) \log f(y_i|x_i, \theta).$$

Note that MLE $\hat{\theta}_{h(x),\mathcal{D}_n}$ is a variable controlled by two factors: (1) a change in the training dataset $\mathcal{D}_n$ and (2) the adjustment of the sampling probability $h(x)$. If $h(x)$ is a uniform distribution $U(x)$, then $\hat{\theta}_{U(x),\mathcal{D}_n}$ is the outcome of empirical risk minimization (ERM).

Fisher information (FI). FI, denoted by $I_{\mathcal{P}(x)}(\theta)$, is an information measure of samples from a given distribution $\mathcal{P}(x, y|\theta)$. It is defined as:

$$I_{\mathcal{P}(x)}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{P}(x)f(y|x,\theta)}[\nabla_\theta \ \log\ f(y|x, \theta)\nabla_\theta^\top \ \log\ f(y|x, \theta)]. \tag{3}$$

FI provides a guideline for understanding the test cross-entropy loss of MLE $\hat{\theta}_{U(x),\mathcal{D}_n}$. When the training set is sampled from $p(x)f(y|x, \theta^\star)$ and the test samples are generated from $q(x)f(y|x, \theta^\star)$, we can understand the test loss of MLE $\hat{\theta}_{U(x),\mathcal{D}_n}$ by using FI as follows.

Theorem 1. *Suppose Assumption 1 in Appendix F and Assumption 2 in Appendix G hold, then for sufficiently large $n = |\mathcal{D}_n|$, the following holds with high probability:*

$$\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[-\log f(y|x, \hat{\theta}_{U(x),\mathcal{D}_n})\right]\right] \leq \tfrac{1}{2n}\,\mathrm{Tr}\left[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}\right]\mathrm{Tr}\left[I_{q(x)}(\theta^\star)\right]. \tag{4}$$

Here is the proof sketch. The left-hand side of equation 4 converges to the Fisher information ratio (FIR) $\mathrm{Tr}\left[I_{p(x)}(\theta^\star)^{-1} I_{q(x)}(\theta^\star)\right]$ related term. Then, FIR can be decomposed into two trace terms with respect to the training and test marginal distributions $p(x)$ and $q(x)$. Finally, we show that the term $\mathrm{Tr}[I_{p(x)}(\theta^\star)^{-1}]$ which is defined in the oracle model parameter can be replaced with $\mathrm{Tr}[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}]$. The proof of Theorem 1 is in Appendix D. Note that Theorem 1 means that the upper bound of the test loss of MLE $\hat{\theta}_{U(x),\mathcal{D}_n}$ can be minimized by reducing $\mathrm{Tr}[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}]$.

Empirical Fisher information (EFI). In practice, the exact FI equation 3 cannot be computed since we do not know the exact data generation distribution $\mathcal{P}(x)f(y|x, \theta)$. For practical reasons, the

[10] Note that for simplicity, we abuse the notation $h(x, y)$ used in Section 3 as $h(x)$. This is exactly the same for a given dataset $\mathcal{D}_n$ situation.

empirical Fisher information (EFI) is commonly used [25, 11] to reduce the computational cost of gathering gradients for all possible classes when x is given. In this study, we used a slightly more generalized EFI that involved the sampling probability $h(x) \in \mathcal{H}$ as follows:

$$\hat{I}_{h(x)}(\theta) = \textstyle\sum_{(x_i,y_i)\in\mathcal{D}_n} h(x_i)\nabla_\theta \log\, f(y_i|x_i,\theta)\nabla_\theta^\top \log\, f(y_i|x_i,\theta). \tag{5}$$

Note that the conventional EFI is the case when $h(x)$ is uniform. EFI provides a guideline for understanding the test cross-entropy loss of MLE $\hat{\theta}_{h(x),\mathcal{D}_n}$.

E.2 Understanding dataset bias problem via min-max problem

Debiasing formulation from the perspective of min-max problem. Improving generalization is understandable as reducing *(min)* the loss of test samples that are difficult *(max)* to infer. There was a trial for formulating the dataset bias problem when multiple training datasets from different training distributions are given [5]. The problem is seen differently when we can only control the sampling probability $h(x) \in \mathcal{H}$ on the given one training dataset $\mathcal{D}_n$. The problem, in our view, can be defined from the left-hand side of equation 4 in a practical point of view as follows:

Definition 1. When the training dataset $\mathcal{D}_n \sim p(x)f(y|x,\theta^\star)$ is given, the debiasing objective is

$$\min_{h(x)\in\mathcal{H}} \max_{q(x)\in\mathcal{M}} \mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)} \left[-\log f(y|x,\hat{\theta}_{h(x),\mathcal{D}_n})\right]. \tag{6}$$

The meaning of Definition 1 is that we have to train the model $\hat{\theta}_{h(x),\mathcal{D}_n}$ so that the loss of the worst case test samples ($\max_{q(x)}$) is minimized by controlling the sampling probability $h(x)$ ($\min_{h(x)}$). Note that since we cannot control the given training dataset $\mathcal{D}_n$ and test marginal distribution $q(x)$, the only controllable term is the sampling probability $h(x)$. Therefore, from Theorem 1 and EFI, we design a practical objective function for the dataset bias problem as follows:

$$\min_{h(x)\in\mathcal{H}} \mathrm{Tr}\left[\hat{I}_{h(x)}(\hat{\theta}_{h(x),\mathcal{D}_n})^{-1}\right]. \tag{7}$$

E.3 Meaning of PGD in terms of equation 7.

In this section, we present an analysis of PGD with respect to equation 7. To do so, we try to understand equation 7, which is difficult to directly solve. It is because computing the trace of the inverse matrix is computationally expensive. Therefore, we intuitively understand equation 7 in the one-dimensional toy scenario.

One-dimensional example. We assume that $\mathcal{D}_n$ comprises sets M and m such that elements in each set share the same loss function $\frac{1}{2}(\theta+a)^2$ for all elements in M and $\frac{1}{2}(\theta-a)^2$ for all elements in m with a given constant a. In other words, there are only two distinguishable points, and we construct the training data from only these two points. We also assume that each sample of M and m has the set dependent probability mass $h_M(x)$ and $h_m(x)$, respectively. With these settings, our objective is to determine $h^\star(x) = \arg\min_{h(x)\in\mathcal{H}} \mathrm{Tr}[\hat{I}_{h(x)}(\hat{\theta}_{h(x),\mathcal{D}_n})^{-1}]$. Thanks to the model's simplicity, we can easily find $h^\star(x)$ in a closed form with respect to the gradient at $\hat{\theta}_{U(x),\mathcal{D}_n}$ for each set, *i.e.,* $g_M(\hat{\theta}_{U(x),\mathcal{D}_n})$ and $g_m(\hat{\theta}_{U(x),\mathcal{D}_n})$.

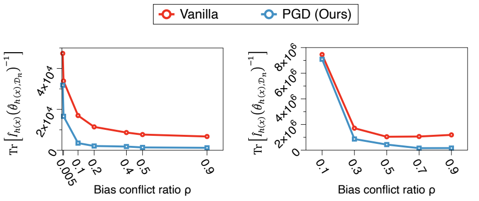
Theorem 2. *Under the above setting, the solution of* $(h_M^\star(x), h_m^\star(x)) = \arg\min_{h(x)\in\mathcal{H}} \mathrm{Tr}[\hat{I}_{h(x)}(\hat{\theta}_{h(x),\mathcal{D}_n})^{-1}]$ *is:*

$$h_M^\star(x) = |g_M(\hat{\theta}_{U(x),\mathcal{D}_n})|/Z, \quad h_m^\star(x) = |g_m(\hat{\theta}_{U(x),\mathcal{D}_n})|/Z,$$

where $Z = |M||g_M(\hat{\theta}_{U(x),\mathcal{D}_n})| + |m||g_m(\hat{\theta}_{U(x),\mathcal{D}_n})|$*, and* $|M|$ *and* $|m|$ *denote the cardinality of* M *and* m*, respectively.*

The proof of Theorem 2 is provided in Appendix E. Note that $h_M^\star(x)$ and $h_m^\star(x)$ are computed using the trained biased model with batches sampled from the uniform distribution $U(x)$. It is the same with the second step of PGD.

PGD tries to minimize equation 7. Theorem 2 implies that equation 7 can be minimized by sampling in proportion to their gradient norm. Because the fundamental of PGD is oversampling based on the gradient norm from the biased model, we can deduce that PGD strives to satisfy equation 7. Furthermore, we empirically show that PGD reduces the trace of the inverse of EFI in the high-dimensional case, as evident in Figure 8.



(a) Colored MNIST (b) Multi-bias MNIST

Figure 8: Target objective $\mathrm{Tr}[\hat{I}_h(\hat{\theta}_{h(x),\mathcal{D}_n})^{-1}]$. PGD : $h(x) = \hat{h}(x)$, and vanilla: $h(x) = U(x)$.

F Backgrounds For Theoretical Analysis

F.1 Notations Summary.

For convenience, we describe notations used in Appendix E, Appendix F, G, and H.

Table 13: Notation Table

Type	Notation	Description	Remark
Variables	(x, y)	(image, label)	$x \in \mathbb{R}^d, y \in C = \{1, ..., c\}$
	$y_{true}(x)$	the true label of image x	labeled by the oracle model $f(y\|x, \theta^\star)$
	θ	model parameter	-
	$\theta^\star$	oracle model parameter	satisfying $f(y_{true}(x)\|x, \theta^\star) = 1$ for any x
	$\mathcal{D}_n$	training dataset	composed of $\{(x_i, y_i)\}_{i=1}^n \sim p(x, y\|\theta^\star)$
	$h(x)$	sampling probability of each sample in $\mathcal{D}_n$	satisfying $h(x) \in \mathcal{H}$
Distributions	$\mathcal{P}(x)$	general distribution of input image x	-
	$p(x)$	distribution of training image	-
	$q(x)$	distribution of test image	-
	$f(y\|x, \theta)$	conditional distribution with model parameter θ	-
	$\mathcal{P}(x, y\|\theta)$	general joint distribution with model parameter θ	$\mathcal{P}(x)f(y\|x, \theta)$
	$p(x, y\|\theta^\star)$	joint distribution of the training dataset	$p(x)f(y\|x, \theta^\star)$
	$q(x, y\|\theta^\star)$	joint distribution of the test dataset	$q(x)f(y\|x, \theta^\star)$
Estimators	$\hat{\theta}_{h(x),\mathcal{D}_n}$	MLE solution on the $\mathcal{D}_n$ with h	$\triangleq \arg\max_\theta \quad \sum_{i=1}^n h(x_i) \log f(y_i\|x_i, \theta)$
	$\hat{\theta}_{U(x),\mathcal{D}_n}$	MLE solution on the $\mathcal{D}_n$ with uniform distribution U	solution of ERM
Fisher information	$I_{\mathcal{P}(x)}(\theta)$	Fisher Information	$\mathbb{E}_{(x,y)\sim\mathcal{P}(x)f(y\|x,\theta)}[\nabla_\theta \log f(y\|x, \theta)\nabla_\theta^\top \log f(y\|x, \theta)]$
	$\hat{I}_{h(x)}(\theta)$	Empirical Fisher Information	$\sum_{i=1}^n h(x_i)\nabla_\theta \log f(y_i\|x_i, \theta)\nabla_\theta^\top \log f(y_i\|x_i, \theta)$
Set	$\mathcal{H}$	set of all possible $h(x)$ on $\mathcal{D}_n$	$\{h(x)\|\sum_{(x_i,y_i)\in\mathcal{D}_n} h(x_i) = 1 \text{ and } h(x_i) \geq 0 \quad \forall(x_i, y_i) \in \mathcal{D}_n\}$
	$\mathcal{M}$	set of all possible marginal $\mathcal{P}(x)$ on input space $\mathcal{X}$	$\{\mathcal{P}(x)\|\int_{x\in\mathcal{X}} \mathcal{P}(x)\, dx = 1\}$
	$\mathcal{W}$	set of all possible $(x, y_{true}(x))$	-
	$supp(\mathcal{P}(x, y\|\theta))$	Support set of $\mathcal{P}(x, y\|\theta)$	$\{(x, y) \in X \times \{1, \cdots, c\} \mid \mathcal{P}(x, y\|\theta) \neq 0\},\ \forall\ \mathcal{P}(x, y\|\theta)$
Order notations in probability	O_{p}	Big O, stochastic boundedness	-
	o_{p}	Small o, convergence in probability	-
Toy example (Appendix H)	M	set of majority (*i.e.,* bias-aligned) samples	-
	m	set of minority (*i.e.,* bias-conflicting) samples	-
	$g_M(\theta)$	gradient of samples in M at θ	-
	$g_m(\theta)$	gradient of samples in m at θ	-
	$h_M^\star(x)$	Optimal sampling probability of samples in M	-
	$h_m^\star(x)$	Optimal sampling probability of samples in m	-

F.2 Main Assumption

Here, we organize the assumptions that are used the proof of Theorems. These are basically used when analyzing models through Fisher information as follows. The assumptions are motivated by [60].

Assumption 1.

(A0). The general joint distribution $\mathcal{P}(x, y|\theta)$ is factorized into the conditional distribution $f(y|x, \theta)$ and the marginal distribution $\mathcal{P}(x)$, not depend on model parameter θ, that is:

$$\mathcal{P}(x, y|\theta) = \mathcal{P}(x)f(y|x, \theta). \tag{8}$$

Thus, the joint distribution is derived from model parameter θ and the marginal distribution $\mathcal{P}(x)$, which is determined from the task that we want to solve. Without loss of generality, we match the joint distribution's name with the marginal distribution.

(A1). *(Identifiability):* The CDF $\boldsymbol{\mathcal{P}}_\theta$ (whose density is given by $\mathcal{P}(x, y|\theta)$) is identifiable for different parameters. Meaning that for every distinct parameter vectors θ_1 and θ_2 in Ω, $\boldsymbol{\mathcal{P}}_{\theta_1}$ and $\boldsymbol{\mathcal{P}}_{\theta_2}$ are also distinct. That is,

$$\forall \theta_1 \neq \theta_2 \in \Omega, \quad \exists A \subseteq X \times \{1, \cdots, c\} \quad \text{s.t.} \quad \boldsymbol{\mathcal{P}}_{\theta_1}(A) \neq \boldsymbol{\mathcal{P}}_{\theta_2}(A),$$

where X, $\{1, \cdots, c\}$ and Ω are input, label and model parameter space, respectively.

(A2). The joint distribution $\boldsymbol{\mathcal{P}}_\theta$ has common support for all $\theta \in \Omega$.

(A3). *(Model Faithfulness)*: For any $x \in X$, we assume an oracle model parameter $\theta^\star$ that generates $y_{\text{true}}(x)$, a true label of input x with the conditional distribution $f(y_{\text{true}}(x)|x, \theta^\star) = 1$.

(A4). *(Training joint):* Let $p(x)$ denote the training marginal with no dependence on the parameter. Then, the set of observations in $\mathcal{D}_n \triangleq \{(x_1, y_1). \cdots (x_n, y_n)\}$ are drawn independently from the training/proposal joint distribution of the form $p(x, y|\theta^\star) \triangleq p(x)f(y|x, \theta^\star)$, because we don't think the existence of mismatched label data situation in the training data.

(A5). *(Test joint):* Let $q(x)$ denote the test marginal with no dependence on the parameter. The unseen test pairs are distributed according to the test/true joint distribution of the form $q(x, y|\theta^\star) \triangleq q(x)f(y|x, \theta^\star)$, because we don't think the existence of mismatched label data situation in the test task.

(A6). *(Differentiability):* The log-conditional $\log f(y|x, \theta)$ is of class $\mathcal{C}^3(\Omega)$ for all $(x, y) \in X \times \{1, 2, \cdots, c\}$, when being viewed as a function of the parameter.[11]

(A7). The parameter space Ω is compact and there exists an open ball around the true parameter of the model $\theta^\star \in \Omega$.

(A8). *(Invertibility):* The arbitrary Fisher information matrix $I_{\mathcal{P}(x)}(\theta)$ is positive definite and therefore invertible for all $\theta \in \Omega$.

(A9). $\{(x, y) \in supp(q(x, y|\theta^\star)) \mid \nabla^2_\theta \log q(x, y|\theta^\star) \text{ is singular}\}$ is a measure zero set.

Compare to [60], we modify (A3) so that the oracle model always outputs hard label *i.e.,* $f(y_{true}(x)|x, \theta^\star) = 1$ and add (A9) which is not numbered but noted in the statement of Theorem 3 and Theorem 11 in [60].

F.3 Preliminaries

We organize the two background knowledge, maximum likelihood estimator (MLE) and Fisher information (FI), needed for future analysis.

F.3.1 Maximum Likelihood Estimator (MLE)

In this section, we derive maximum likelihood estimator in the context of classification problem with sampling probability $h(x)$. Unless otherwise stated, training set $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ is sampled from $p(x, y|\theta^\star)$. For given probability mass function (PMF) $h(x)$ on $\mathcal{D}_n$, in abstract $h(x) \in \mathcal{H}$, we define

[11] We say that a function $f : X \to Y$ is of $\mathcal{C}^p(X)$, for an integer $p > 0$, if its derivatives up to p-th order exist and are continuous at all points of X.

MLE $\hat{\theta}_{h(x),\mathcal{D}_n}$ as follows:

$$\begin{aligned}
\hat{\theta}_{h(x),\mathcal{D}_n} &\triangleq \arg\max_{\theta} \quad \log \mathbb{P}(\mathcal{D}_n|\theta; h(x)) \\
&= \arg\min_{\theta} \quad -\sum_{i=1}^{n} h(x_i) \log p(x_i, y_i|\theta) \qquad (9) \\
&= \arg\min_{\theta} \quad -\sum_{i=1}^{n} h(x_i) \log f(y_i|x_i, \theta) \qquad (10) \\
&= \arg\min_{\theta} \quad \sum_{i=1}^{n} h(x_i)\, \mathcal{L}_{\text{CE}}(x_i, y_i; \theta). \qquad (11)
\end{aligned}$$

In equation 9 and equation 10, (A4) and (A0) of the Assumption 1 in Appendix F was used, respectively. Note that MLE $\hat{\theta}_{h(x),\mathcal{D}_n}$ is a variable controlled by two factors: (1) a change in the training dataset $\mathcal{D}_n$ and (2) the adjustment of the sampling probability $h(x)$. If $h(x)$ is a uniform distribution $U(x)$, then $\hat{\theta}_{U(x),\mathcal{D}_n}$ is the outcome of empirical risk minimization (ERM).

F.3.2 Fisher information (FI)

General definition of FI. Fisher information (FI), denoted by $I_{\mathcal{P}(x)}(\theta)$, is an information measure of samples from the given distribution $\mathcal{P}(x, y|\theta) \triangleq \mathcal{P}(x)f(y|x, \theta)$. It is defined as the expected value of the outer-product of the score function $\nabla_\theta \log \mathcal{P}(x, y|\theta)$ with itself, evaluated at some $\theta \in \Omega$.

$$I_{\mathcal{P}(x)}(\theta) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{P}(x,y|\theta)}[\nabla_\theta \log \mathcal{P}(x, y|\theta)\nabla_\theta^\intercal \log \mathcal{P}(x, y|\theta)]. \qquad (12)$$

Extended version of FI. Here, we summarize extended version of FI, which can be derived by some assumptions. These variants of FI are utilized in the proof of Theorems.

- *(Hessian version)* Under the differentiability condition (A6) of Assumption 1 in Appendix F, FI can be written in terms of the Hessian matrix of the log-likelihood:

$$I_{\mathcal{P}(x)}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{P}(x,y|\theta)}[\nabla_\theta^2 \log \mathcal{P}(x, y|\theta)]. \qquad (13)$$

- *(Model decomposition version)* Under the factorization condition (A0) of Assumption 1 in Appendix F, equation 12 and equation 13 can be transformed as follows:

$$\begin{aligned}
I_{\mathcal{P}(x)}(\theta) &= \mathbb{E}_{(x,y)\sim\mathcal{P}(x)f(y|x,\theta)}[\nabla_\theta \log f(y|x, \theta)\nabla_\theta^\intercal \log f(y|x, \theta)] \qquad (14) \\
&= -\mathbb{E}_{(x,y)\sim\mathcal{P}(x)f(y|x,\theta)}[\nabla_\theta^2 \log f(y|x, \theta)]. \qquad (15)
\end{aligned}$$

Specifically, equation 14 and equation 15 can be unfolded as follows:

$$\begin{aligned}
I_{\mathcal{P}(x)}(\theta) &= \int_{x\in X} \mathcal{P}(x) \sum_{y=1}^{c} \left[f(y|x, \theta) \cdot \nabla_\theta \log f(y|x, \theta)\nabla_\theta^\intercal \log f(y|x, \theta)\right] dx \qquad (16) \\
&= -\int_{x\in X} \mathcal{P}(x) \sum_{y=1}^{c} \left[f(y|x, \theta) \cdot \nabla_\theta^2 \log f(y|x, \theta)\right] dx
\end{aligned}$$

Hereinafter, we define $I_{p(x)}(\theta)$ and $I_{q(x)}(\theta)$ as the FI derived from the training and test marginal, respectively.

F.3.3 Empirical Fisher information (EFI)

When the training dataset $\mathcal{D}_n$ is given, we denote the sampling probability as $h(x)$ which is defined on the probability space $\mathcal{H}$:

$$\mathcal{H} = \{h(x) \,|\, \textstyle\sum_{(x_i,y_i)\in\mathcal{D}_n} h(x_i) = 1\,,\; h(x_i) \geq 0 \quad \forall (x_i, y_i) \in \mathcal{D}_n\}.^{12} \qquad (17)$$

[12] Note that for simplicity, we abuse the notation $h(x, y)$ used in Section 3 as $h(x)$. This is exactly the same for a given dataset $\mathcal{D}_n$ situation.

Practically, the training dataset $\mathcal{D}_n$ is given as deterministic. Therefore, equation 14 can be refined as empirical Fisher information (EFI). This reformulation is frequently utilized, *e.g.,* in [25, 11], to reduce the computational complexity of gathering gradients for all possible classes (*i.e.,* expectation with respect to $f(y|x,\theta)$ as in equation 14). Refer the $\sum_{y=1}^{c}$ term of equation 16. Different with prior EFI which is defined on the case when $h(x)$ is uniform, $U(x)$, we generalize the definition of EFI in terms of $h(x) \in \mathcal{H}$ as follows:

$$\begin{aligned}\hat{I}_{h(x)}(\theta) &:= \mathbb{E}_{h(x)}\left[\nabla_\theta \log\, f(y|x,\theta)\nabla_\theta^\top \log\, f(y|x,\theta)\right] \\ &\overset{(a)}{:=} \textstyle\sum_{(x_i,y_i)\in\mathcal{D}_n} h(x_i)\nabla_\theta \log\, f(y_i|x_i,\theta)\nabla_\theta^\top \log\, f(y_i|x_i,\theta).\end{aligned} \tag{18}$$

Note that (a) holds owing to equation 17.

F.3.4 Stochastic Order Notations o_p and O_p

For a set of random variables X_n and a corresponding set of constant a_n, the notation $X_n = o_\mathrm{p}(a_n)$ means that the set of values X_n/a_n converges to zero in probability as n approaches an appropriate limit. It is equivalent with $X_n/a_n = o_\mathrm{p}(1)$, where $X_n = o_\mathrm{p}(1)$ is defined as:

$$\lim_{n\to\infty} \mathbb{P}(|X_n| \geq \epsilon) = 0 \quad \forall\, \epsilon \geq 0.$$

The notation $X_n = O_\mathrm{p}(a_n)$ means that the set of values X_n/a_n is stochastically bounded. That is $\forall\, \epsilon > 0, \exists$ finite $M > 0$, $N > 0$ s.t. $\mathbb{P}(|X_n/a_n| > M) < \epsilon$ for any $n > N$.

G Theorem 1

In this section, we deal with some require sub-lemmas which are used for the proof of Lemma 8 which is main ingredient of the proof of Theorem 3.

G.1 Sub-Lemmas

Lemma 1 ([39], Theorem 5.1)**.** *When $\xrightarrow{P}$ denotes convergence in probability, and if (A0) to (A7) of the Assumption 1 in Appendix F hold, then there exists a sequence of MLE solutions $\{\hat{\theta}_{U(x),\mathcal{D}_n}\}_{n\in\mathbb{N}}$ that $\hat{\theta}_{U(x),\mathcal{D}_n} \xrightarrow{P} \theta^\star$ as $n \to \infty$, where $\theta^\star$ is the 'true' unknown parameter of the distribution of the sample.*

Proof. We refer to [39] for a detailed proof. □

Lemma 2 ([39], Theorem 5.1)**.** *Let $\{\hat{\theta}_{U(x),\mathcal{D}_n}\}_{n\in\mathbb{N}}$ be the MLE based on the training data set $\mathcal{D}_n$. If (A0) to (A8) of the Assumption 1 in Appendix F hold, then the MLE $\hat{\theta}_{U(x),\mathcal{D}_n}$ has a zero-mean normal asymptotic distribution with the covariance equal to the inverse Fisher information matrix, and with the convergence rate of 1/2:*

$$\sqrt{n}(\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star) \xrightarrow{D} \mathcal{N}(\vec{0}, I_{p(x)}(\theta^\star)^{-1}),$$

where $\xrightarrow{D}$ represents convergence in distribution.

Proof. We refer to [39] for a detailed proof, which is based on Lemma 1. □

Lemma 3 ([65], Theorem 9.18)**.** *Under the (A0) to (A8) of the Assumption 1 in Appendix F hold, we get*

$$\sqrt{n}I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{1/2}(\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star) \xrightarrow{D} \mathcal{N}(\vec{0}, \mathbb{I}_d),$$

where $\xrightarrow{D}$ represents convergence in distribution.

Proof. We refer to [65] for a detailed proof, which is based on Lemma 2. □

Lemma 4. *([55], Chapter 1) Let $\{\theta_n\}$ be a sequence of random vectors. If there exists a random vector $\tilde{\theta}$ such that $\theta_n \xrightarrow{D} \tilde{\theta}$, then $\|\theta_n - \tilde{\theta}\|_2 = O_p(1)$, where $\|\cdot\|_2$ denote the L_2 norm.*

Proof. We refer to [55] for a detailed proof. □

Lemma 5. *([60], Theorem 27) Let $\{\theta_n\}$ be a sequence of random vectors in a convex and compact set $\Omega \subseteq \mathbb{R}^d$ and $\theta^\star \in \Omega$ be a constant vector such that $\|\theta_n - \theta^\star\|_2 = O_p(a_n)$ where $a_n \to 0$ (as $n \to \infty$). If $g : \Omega \to \mathbb{R}$ is a $\mathcal{C}^3$ function, then*

$$g(\theta_n) = g(\theta^\star) + \nabla_\theta^\mathsf{T} g(\theta^\star)(\theta_n - \theta^\star) + \frac{1}{2}(\theta_n - \theta^\star)^\mathsf{T} \nabla_\theta^2 g(\theta^\star)(\theta_n - \theta^\star) + o_p({a_n}^2).$$

Proof. We refer to [55] for a detailed proof. □

Lemma 6. *If (A0) and (A3) of the Assumption 1 in Appendix F hold, then $\nabla_\theta \log \mathcal{P}(x, y_{true}(x)|\theta^\star) = \vec{0}$ for any joint distribution $\mathcal{P}(x, y|\theta^\star)$.*

Proof.

$$\begin{aligned}\nabla_\theta \log \mathcal{P}(x, y_{true}(x)|\theta^\star) &= \nabla_\theta \log f(y_{true}(x)|x, \theta^\star) \\ &= \nabla_\theta \log 1 \\ &= \vec{0}.\end{aligned}$$

On the first equality, (A0) of the Assumption 1 in Appendix F is used. At the second equality, (A3) of the Assumption 1 in Appendix F is used.

□

Lemma 7. *If (A0) to (A8) of the Assumption 1 in Appendix F hold and the case $\nabla_\theta^2 \log p(x, y_{true}(x)|\theta^\star)$ is non-singular for given data $(x, y_{true}(x))$ satisfies, then the asymptotic distribution of the log-likelihood ratio is a mixture of first-order Chi-square distributions, and the convergence rate is one. More specifically:*

$$n\left(\log \frac{p(x, y_{true}(x)|\theta^\star)}{p(x, y_{true}(x)|\hat{\theta}_{U(x),\mathcal{D}_n})}\right) \xrightarrow{D} \frac{1}{2}\sum_{i=1}^{d} \lambda_i(x, y_{true}(x)) \cdot \chi_1^2, \tag{19}$$

where $\{\lambda_i(x, y_{true}(x))\}_{i=1}^d$ are eigenvalues of $I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\left\{-\nabla_\theta^2 \log p(x, y_{true}(x)|\theta^\star)\right\} I_{p(x)}(\theta^\star)^{-\frac{1}{2}}$.

Proof. The proof is based on taylor expansion theorem. Remind that we deal with the data $(x, y_{true}(x))$ satisfying $\nabla_\theta^2 \log p(x, y_{true}(x)|\theta^\star)$ is non-singular. From the property $\sqrt{n}(\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star) \xrightarrow{D} \mathcal{N}(\vec{0}, I_{p(x)}(\theta^\star)^{-1})$ derived from Lemma 3, one concludes that $\sqrt{n}\|\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star\|_2 = O_\mathrm{p}(1)$ and therefore $\|\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star\|_2 = O_\mathrm{p}(\frac{1}{\sqrt{n}})$ by the Lemma 4.

Thus, by the Lemma 5,

$$\begin{aligned}\log p(x, y_{true}(x)|\hat{\theta}_{U(x),\mathcal{D}_n}) = \log p(x, y_{true}(x)|\theta^\star) + (\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star)^\mathsf{T} \nabla_\theta \log p(x, y_{true}(x)|\theta^\star) \\ + \frac{1}{2}(\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star)^\mathsf{T} \nabla_\theta^2 \log p(x, y_{true}(x)|\theta^\star)(\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star) + o_\mathrm{p}\left(\frac{1}{n}\right)\end{aligned}$$

hold. By the Lemma 3 and the property $\nabla_\theta \log p(x, y_{true}(x)|\theta^\star) = \vec{0}$ derived by Lemma 6, we can obtain

$$\begin{aligned}&n\left[\log p(x, y_{true}(x)|\theta^\star) - \log p(x, y_{true}(x)|\hat{\theta}_{U(x),\mathcal{D}_n})\right] \\ &= -\frac{1}{2}\left[\sqrt{n}(\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star)\right]^\mathsf{T} \nabla_\theta^2 \log p(x, y_{true}(x)|\theta^\star)\left[\sqrt{n}(\hat{\theta}_{U(x),\mathcal{D}_n} - \theta^\star)\right] + o_\mathrm{p}(1) \\ &\xrightarrow{D} \frac{1}{2}\mathcal{N}\left(\vec{0}, I_{p(x)}(\theta^\star)^{-1}\right)^\mathsf{T}\left[-\nabla_\theta^2 \log p(x, y_{true}(x)|\theta^\star)\right]\mathcal{N}\left(\vec{0}, I_{p(x)}(\theta^\star)^{-1}\right) \\ &= \frac{1}{2}\mathcal{N}\left(\vec{0}, \mathbb{I}_d\right)^\mathsf{T}\left[-I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\nabla_\theta^2 \log p(x, y_{true}(x)|\theta^\star) I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\right]\mathcal{N}\left(\vec{0}, \mathbb{I}_d\right).\end{aligned}$$

Define $\Gamma(x, y_{true}(x))$ as $-I_{p(x)}(\theta^\star)^{-\frac{1}{2}} \nabla^2_\theta \log\, p(x, y_{true}(x)|\theta^\star) I_{p(x)}(\theta^\star)^{-\frac{1}{2}}$ and rewrite the right-hand-side element-wise[13] as

$$\begin{aligned}
&\frac{1}{2}\mathcal{N}\left(\vec{0}, \mathbb{I}_d\right)^\intercal \Gamma(x, y_{true}(x)) \mathcal{N}\left(\vec{0}, \mathbb{I}_d\right) \\
&\qquad = \frac{1}{2}\sum_{i=1}^{d} \lambda_i(x, y_{true}(x)) \cdot \mathcal{N}(0,1)^2 = \frac{1}{2}\sum_{i=1}^{d} \lambda_i(x, y_{true}(x)) \cdot {\chi_1}^2,
\end{aligned}$$

where $\{\lambda_i(x, y_{true}(x))\}_{i=1}^d$ are eigenvalues of $\Gamma(x, y_{true}(x))$. Thus, desired property is obtained. □

G.2 Main Lemma

In this section, we derive main Lemma which represents the test cross entropy loss can be understood as Fisher information ratio (FIR) [60].

G.2.1 Main Lemma statement and proof

Lemma 8 (FIR in expected test cross entropy loss with MLE)**.** *If the Assumption 1 in Appendix F holds, then*

$$\begin{aligned}
\lim_{n\to\infty} n\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)} &\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[-\log f(y|x, \hat{\theta}_{U(x),\mathcal{D}_n})\right]\right] \\
&= \frac{1}{2}\operatorname{Tr}\left[I_{p(x)}(\theta^\star)^{-1} I_{q(x)}(\theta^\star)\right].
\end{aligned} \tag{20}$$

Proof. We prove Lemma 8 via two steps. First we show that the expected cross entropy loss term can be rewritten in terms of log-likelihood ratio. Then, we prove that the expected log-likelihood ratio can be asymptotically understood as FIR.

Step 1: Log-likelihood ratio We show that the expected log-likelihood ratio can be formulated as expected test cross-entropy loss as follows:

This property holds because,

$$\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[\log\frac{p(x,y|\theta^\star)}{p(x,y|\hat{\theta}_{U(x),\mathcal{D}_n})}\right]\right]$$

$$= \mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[\log\frac{f(y|x,\theta^\star)}{f(y|x,\hat{\theta}_{U(x),\mathcal{D}_n})}\right]\right] \tag{21}$$

$$= \mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[\log\frac{f(y|x,\theta^\star)}{f(y|x,\hat{\theta}_{U(x),\mathcal{D}_n})}\right]\mathbb{1}_{Supp(q(x,y|\theta^\star))}\right] \tag{22}$$

$$= \mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[\log\frac{f(y|x,\theta^\star)}{f(y|x,\hat{\theta}_{U(x),\mathcal{D}_n})}\ \mathbb{1}_{Supp(q(x,y|\theta^\star))}\right]\right]$$

$$= \mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[-\log f(y|x,\hat{\theta}_{U(x),\mathcal{D}_n})\ \mathbb{1}_{Supp(q(x,y|\theta^\star))}\right]\right] \tag{23}$$

$$= \mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[-\log f(y|x,\hat{\theta}_{U(x),\mathcal{D}_n})\right]\right]$$

Since (A0) of Assumption 1 in Appendix F, equation 21 and equation 22 hold. At equation 23, the properties, (i) $Supp(q(x,y|\theta^\star)) \subseteq \mathcal{W}$ and (ii) $f(y|x,\theta^\star) = 1\ \forall\,(x,y) \in \mathcal{W}$, was used which is derived by (A3) of the Assumption 1 in Appendix F.

Step 2: FIR Here, we show that the expected test loss of MLE can be understood as FIR.

[13] Suppose $\Gamma = U\Sigma U^\intercal$ and $\Sigma = diag(\lambda_1, \cdots \lambda_d)$. Then, $\mathcal{N}\left(\vec{0}, \mathbb{I}_d\right)^\intercal U \sim \mathcal{N}\left(\vec{0}, UU^\intercal\right) = \mathcal{N}\left(\vec{0}, \mathbb{I}_d\right)$. Thus, $\mathcal{N}\left(\vec{0}, \mathbb{I}_d\right)^\intercal \Gamma \mathcal{N}\left(\vec{0}, \mathbb{I}_d\right) = \mathcal{N}\left(\vec{0}, \mathbb{I}_d\right)^\intercal \Sigma \mathcal{N}\left(\vec{0}, \mathbb{I}_d\right) = \sum_{i=1}^d \lambda_i \mathcal{N}(0,1)^2$.

By (A0) in Assumption 1 in Appendix F, trivially

$$\begin{aligned}\{(x,y) \in & Supp(q(x,y|\theta^\star)) \mid \nabla_\theta^2 \log q(x,y|\theta^\star) \text{ is singular}\} \\ &= \{(x,y) \in Supp(q(x,y|\theta^\star)) \mid \nabla_\theta^2 \log p(x,y|\theta^\star) \text{ is singular}\}.\end{aligned} \tag{24}$$

holds. Since equation 24, and (A9) in Assumption 1 in Appendix F, $Supp(q(x,y|\theta^\star))$ can be replaced by,

$$S \triangleq Supp(q(x,y|\theta^\star)) \setminus \{(x,y) \in \mathcal{W} \mid \nabla_\theta^2 \log p(x,y|\theta^\star) \text{ is singular}\} \tag{25}$$

when calculate expectation.

We can get a result of Lemma 8 as follows:

$$\begin{aligned}
&\lim_{n\to\infty} n\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[-\log f(y|x,\hat{\theta}_{U(x),\mathcal{D}_n})\right]\right] \\
&= \lim_{n\to\infty} n\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[\log\frac{p(x,y|\theta^\star)}{p(x,y|\hat{\theta}_{U(x),\mathcal{D}_n})}\right]\mathbf{1}_{Supp(q(x,y|\theta^\star))}\right] &(26)\\
&= \lim_{n\to\infty} n\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[\log\frac{p(x,y|\theta^\star)}{p(x,y|\hat{\theta}_{U(x),\mathcal{D}_n})}\right]\mathbf{1}_S\right] &(27)\\
&= \mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\lim_{n\to\infty}\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[n\log\frac{p(x,y|\theta^\star)}{p(x,y|\hat{\theta}_{U(x),\mathcal{D}_n})}\right]\mathbf{1}_S\right]
\end{aligned}$$

$$\begin{aligned}
&= \mathbb{E}_{(x,y_{true}(x))\sim q(x)f(y|x,\theta^\star)}\left[\lim_{n\to\infty}\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[n\log\frac{p(x,y_{true}(x)|\theta^\star)}{p(x,y_{true}(x)|\hat{\theta}_{U(x),\mathcal{D}_n})}\right]\mathbf{1}_S\right] &(28)\\
&= \mathbb{E}_{(x,y_{true}(x))\sim q(x)f(y|x,\theta^\star)}\left[\left(\frac{1}{2}\sum_{i=1}^d \lambda_i(x,y_{true}(x))\mathbb{E}\left[\chi_1^2\right]\right)\mathbf{1}_S\right] &(29)\\
&= \mathbb{E}_{(x,y_{true}(x))\sim q(x)f(y|x,\theta^\star)}\left[\left(\frac{1}{2}\sum_{i=1}^d \lambda_i(x,y_{true}(x))\right)\mathbf{1}_S\right] &(30)\\
&= \mathbb{E}_{(x,y_{true}(x))\sim q(x)f(y|x,\theta^\star)}\left[\frac{1}{2}\operatorname{Tr}\left[I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\left\{-\nabla_\theta^2\log p(x,y_{true}(x)|\theta^\star)\right\}I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\right]\mathbf{1}_S\right] &(31)\\
&= \mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\frac{1}{2}\operatorname{Tr}\left[I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\left\{-\nabla_\theta^2\log p(x,y|\theta^\star)\right\}I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\right]\mathbf{1}_S\right] &(32)\\
&= \frac{1}{2}\operatorname{Tr}\left[I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[-\nabla_\theta^2\log p(x,y|\theta^\star)\ \mathbf{1}_S\right]I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\right]\\
&= \frac{1}{2}\operatorname{Tr}\left[I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[-\nabla_\theta^2\log q(x,y|\theta^\star)\ \mathbf{1}_S\right]I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\right] &(33)\\
&= \frac{1}{2}\operatorname{Tr}\left[I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[-\nabla_\theta^2\log q(x,y|\theta^\star)\right]I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\right]\\
&= \frac{1}{2}\operatorname{Tr}\left[I_{p(x)}(\theta^\star)^{-\frac{1}{2}}I_{q(x)}(\theta^\star)I_{p(x)}(\theta^\star)^{-\frac{1}{2}}\right]\\
&= \frac{1}{2}\operatorname{Tr}\left[I_{p(x)}(\theta^\star)^{-1}I_{q(x)}(\theta^\star)\right]. &(34)
\end{aligned}$$

equation 26 holds from Step 1. From equation 25, equation 27 satisfied. Since (x,y) is sampled from $q(x)f(y|x.\theta^\star)$, equation 28 and equation 32 holds. From equation 29 to equation 31, the result of Lemma 7 is used. equation 33 can be obtained thanks to (A0). Lastly, equation 34 holds because trace satisfies the commutative law about matrix multiplication. The last term, $\operatorname{Tr}\left[I_{p(x)}(\theta^\star)^{-1}I_{q(x)}(\theta^\star)\right]$ is named as Fisher information ratio (FIR) since it is understanble as a ratio when the scalar case.

□

G.3 Theorem 1

In this section, we finally prove Theorem 3. To do so, we additionally follow assumptions in [60].

G.3.1 Additional Assumption

Assumption 2. We assume to exist four positive constants $L_1, L_2, L_3, L_4 \geq 0$ such that following properties hold $\forall x \in X, y \in \{1, \cdots, c\}$ and $\theta \in \Omega$:

- $I(\theta, x) = -\nabla_\theta^2 \log f(y|x, \theta)$ is independent of the class labels y.
- $\nabla_\theta \log f(y|x, \theta^\star)^\mathsf{T} I_{q(x)}(\theta^\star)^{-1} \nabla_\theta \log f(y|x, \theta^\star) \leq L_1$
- $\|I_{q(x)}(\theta^\star)^{-1/2} I(\theta^\star, x) I_{q(x)}(\theta^\star)^{-1/2}\| \leq L_2$
- $\|I_{q(x)}(\theta^\star)^{-1/2}(I(\theta^{'}, x) - I(\theta^{''}, x)) I_{q(x)}(\theta^\star)^{-1/2}\| \leq L_3(\theta^{'} - \theta^{''})^\mathsf{T} I_{q(x)}(\theta^\star)(\theta^{'} - \theta^{''})$
- $-L_4\|\theta - \theta^\star\| I(\theta^\star, x) \preceq I(\theta, x) - I(\theta^\star, x) \preceq L_4\|\theta - \theta^\star\| I(\theta^\star, x)$

G.3.2 Replacing $\theta^\star$ by $\hat{\theta}_{U(x),\mathcal{D}_n}$

Lemma 9. *Suppose Assumption 1 in Appendix F and Assumption 2 in Appendix G hold, then with high probability:*

$$\mathrm{Tr}\left[I_{p(x)}(\theta^\star)^{-1}\right] = \lim_{n\to\infty} \mathrm{Tr}\left[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}\right]. \tag{35}$$

Proof. It is shown in the proof of Lemma 2 in [12] that under assumptions mentioned in the Assumption 2, the following inequalities hold with probability $1 - \delta(n)$:

$$\frac{\beta(n) - 1}{\beta(n)} I(\theta^\star, x) \preceq I(\hat{\theta}_{U(x),\mathcal{D}_n}, x) \preceq \frac{\beta(n) + 1}{\beta(n)} I(\theta^\star, x), \tag{36}$$

where $\beta(n)$ and $1 - \delta(n)$ are proportional to n, which is size of training set $\mathcal{D}_n$.

Note that for any marginal distribution $\mathcal{P}(x)$, $I_{\mathcal{P}(x)}(\theta) = \mathbb{E}_{x\sim\mathcal{P}(x)}\left[I(\theta, x)\right]$ holds because of the independence for the class labels y of $I(\theta, x)$.[14]

Taking the expectation to the equation 36 with respect to the marginal $p(x)$ and $q(x)$, then:

$$\frac{\beta(n) - 1}{\beta(n)} I_{p(x)}(\theta^\star) \preceq I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n}) \preceq \frac{\beta(n) + 1}{\beta(n)} I_{p(x)}(\theta^\star). \tag{37}$$

$$\frac{\beta(n) - 1}{\beta(n)} I_{q(x)}(\theta^\star) \preceq I_{q(x)}(\hat{\theta}_{U(x),\mathcal{D}_n}) \preceq \frac{\beta(n) + 1}{\beta(n)} I_{q(x)}(\theta^\star).$$

Since $I_{p(x)}(\theta^\star)$ and $I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})$ are assumed to be positive definite, we can write equation 37 in terms of inverted matrices[15]:

$$\frac{\beta(n)}{\beta(n) + 1} I_{p(x)}(\theta^\star)^{-1} \preceq I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1} \preceq \frac{\beta(n)}{\beta(n) - 1} I_{p(x)}(\theta^\star)^{-1}. \tag{38}$$

equation 38 is equivalent to

$$\frac{\beta(n) - 1}{\beta(n)} I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1} \preceq I_{p(x)}(\theta^\star)^{-1} \preceq \frac{\beta(n) + 1}{\beta(n)} I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}. \tag{39}$$

From equation 39,

$$\frac{\beta(n) - 1}{\beta(n)} \mathrm{Tr}\left[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}\right] \leq \mathrm{Tr}\left[I_{p(x)}(\theta^\star)^{-1}\right] \leq \frac{\beta(n) + 1}{\beta(n)} \mathrm{Tr}\left[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}\right] \tag{40}$$

[14] $I_{\mathcal{P}(x)}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{P}(x,y|\theta)}\left[-\nabla_\theta^2 \log f(y|x,\theta)\right] = \mathbb{E}_{x\sim\mathcal{P}(x)}\left[\mathbb{E}_{y\sim f(y|x,\theta)}\left[-\nabla_\theta^2 \log f(y|x,\theta)\right]\right] = \mathbb{E}_{x\sim\mathcal{P}(x)}\left[\mathbb{E}_{y\sim f(y|x,\theta)}\left[I(\theta, x)\right]\right] = \mathbb{E}_{x\sim\mathcal{P}(x)}\left[I(\theta, x)\right]$.

[15] For $\forall$ two positive definite matrices A and B, we have that $A \succeq B \Rightarrow A^{-1} \preceq B^{-1}$

satisfies.[16]

Thus,

$$\lim_{n\to\infty} \mathrm{Tr}\left[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}\right] = \mathrm{Tr}\left[I_{p(x)}(\theta^\star)^{-1}\right] \tag{41}$$

holds when taking $n \to \infty$ to the equation 40. Note that $\beta(n)$ is proportional to n.

□

G.4 Statement and proof of Theorem 3

Theorem 3. *Suppose Assumption 1 in Appendix F and Assumption 2 in Appendix G hold, then for sufficiently large $n = |\mathcal{D}_n|$, the following holds with high probability:*

$$\begin{aligned}\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[-\log f(y|x,\hat{\theta}_{U(x),\mathcal{D}_n})\right]\right]\\ \leq \frac{1}{2n}\mathrm{Tr}\left[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}\right]\mathrm{Tr}\left[I_{q(x)}(\theta^\star)\right].\end{aligned} \tag{42}$$

Proof. Because of the (A8) of Assumption 1 in Appendix F, $I_{p(x)}(\theta^\star)^{-1}$ and $I_{q(x)}(\theta^\star)$ are positive definite matrix. Thus, $\mathrm{Tr}\left[I_{p(x)}(\theta^\star)^{-1}I_{q(x)}(\theta^\star)\right] \leq \mathrm{Tr}\left[I_{p(x)}(\theta^\star)^{-1}\right]\mathrm{Tr}\left[I_{q(x)}(\theta^\star)\right]$ holds.[17]

From the result of Lemma 8 and 9 in Appendix G,

$$\begin{aligned}\lim_{n\to\infty} n\mathbb{E}_{(x,y)\sim q(x)f(y|x,\theta^\star)}&\left[\mathbb{E}_{\mathcal{D}_n\sim p(x)f(y|x,\theta^\star)}\left[-\log f(y|x,\hat{\theta}_{U(x),\mathcal{D}_n})\right]\right]\\ &= \frac{1}{2}\mathrm{Tr}\left[I_{p(x)}(\theta^\star)^{-1}I_{q(x)}(\theta^\star)\right]\\ &\leq \frac{1}{2}\mathrm{Tr}\left[I_{p(x)}(\theta^\star)^{-1}\right]\mathrm{Tr}\left[I_{q(x)}(\theta^\star)\right]\\ &= \frac{1}{2}\lim_{n\to\infty}\mathrm{Tr}\left[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}\right]\mathrm{Tr}\left[I_{q(x)}(\theta^\star)\right]\end{aligned}$$

holds with high probability.
Thus, desired result is obtained. □

Note that Theorem 3 means that the upper bound of the test loss of MLE $\hat{\theta}_{U(x),\mathcal{D}_n}$ can be minimized by reducing $\mathrm{Tr}\left[I_{p(x)}(\hat{\theta}_{U(x),\mathcal{D}_n})^{-1}\right]$ when training marginal $p(x)$ is the only tractable and controllable variable.

H Theorem 2

In this section, we introduce the motivation of gradient norm based importance sampling. To get the motivation, we introduce the debiasing object problem for a given $\mathcal{D}_n$ under sampling probability $h(x)$ and provide evidence of solution of this problem in the toy example setting because of the problem difficulty.

H.1 Practical objective function for the dataset bias problem

Remark that right-hand side term of equation 42 are controlled by the training and test marginals $p(x)$, and $q(x)$. Since we can only control the training dataset $\mathcal{D}_n$ not $p(x)$ and $q(x)$, we can design a practical objective function for the dataset bias problem by using EFI and Theorem 3 as follows:

$$\min_{h(x)\in\mathcal{H}} \mathrm{Tr}\left[\hat{I}_{h(x)}(\hat{\theta}_{h(x),\mathcal{D}_n})^{-1}\right], \tag{43}$$

[16] If $A \preceq B$, $\mathrm{Tr}\left[A\right] \leq \mathrm{Tr}\left[B\right]$ holds.
$(\because)$ $A \preceq B \Rightarrow B - A \succeq \mathcal{O}$ and $B - A := U\Sigma U^T$, where $U = [u_1|\cdots,|u_d]$.
Then, $\mathrm{Tr}(B-A) = \sum_{i=1}^{d} u_i(B-A)u_i^T \leq 0$ because of the positive definite property of $B - A$.
$\mathrm{Tr}(B-A) \leq 0 \Rightarrow \mathrm{Tr}(B) \leq \mathrm{Tr}(A)$.

[17] For $\forall$ two positive definite matrices A and B, $\mathrm{Tr}\left[AB\right] \leq \mathrm{Tr}\left[A\right]\mathrm{Tr}\left[B\right]$ satisfies.

where $\hat{I}_{h(x)}(\theta)$ is an empirical Fisher information matrix. Remark that EFI is defined as:

$$\hat{I}_{h(x)}(\theta) = \sum_{i=1}^{n} h(x_i) \nabla_\theta \log\, f(y_i|x_i, \theta) \nabla_\theta^\top \log\, f(y_i|x_i, \theta). \tag{44}$$

Here, $h(x)$ describes the sampling probability on $\mathcal{D}_n$, which is the only controllable term. We deal with equation 43 in the toy example because of the problem difficulty.

H.2 One-dimensional Toy Example Setting

For simplicity, we assume that $\mathcal{D}_n$ comprises sets M and m and the samples in each set share their same loss function and same probability mass. The detail is like below:

- For the given $a \in \mathbb{R}$, at the model parameter $\theta \in \mathbb{R}$, $\frac{1}{2}(\theta + a)^2$ and $\frac{1}{2}(\theta - a)^2$ loss function arise for all data in M and m, respectively.
- $\hat{\theta}_{h,\mathcal{D}_n}$ denote the trained model from the arbitrary PMF $h(x) \in \mathcal{H}$ which has a constraint having degree of freedom 2, $(h_M(x), h_m(x))$.
- Concretely, each samples of M and m has a probability mass $h_M(x)$ and $h_m(x)$, respectively. *i.e.,* $|M| \cdot h_M(x) + |m| \cdot h_m(x) = 1$, where $|M|$ and $|m|$ denote the cardinality of M and m, respectively.
- Let $g_M(\theta)$ and $g_m(\theta)$ denote the gradient of each sample in M and m at $\theta \in \mathbb{R}$, respectively.
- Then, $|M| \cdot h_M(x) \cdot g_M(\hat{\theta}_{h(x),\mathcal{D}_n}) + |m| \cdot h_m(x) \cdot g_m(\hat{\theta}_{h(x),\mathcal{D}_n}) = 0$ hold by the definition of $\hat{\theta}_{h(x),\mathcal{D}_n}$.
- In this settings, our objective can be written as finding $h^\star(x) = \arg\min_{h(x)\in\mathcal{H}} \operatorname{Tr}\left[\hat{I}_h(x)(\hat{\theta}_{h(x),\mathcal{D}_n})^{-1}\right]$ and this is equivalent to find $(h_M^\star(x), h_m^\star(x))$.

H.3 Statement and proof of Theorem 4

In this section, we introduce the motivation of the gradient norm based importance sampling in the toy example setting.

Theorem 4. *Under the above setting, the solution of* $(h_M^\star(x), h_m^\star(x)) = \arg\min_{h(x)\in\mathcal{H}} \operatorname{Tr}\left[\hat{I}_{h(x)}(\hat{\theta}_{h(x),\mathcal{D}_n})^{-1}\right]$ *is:*

$$h_M^\star(x) = \frac{|g_M(\hat{\theta}_{U(x),\mathcal{D}_n})|}{Z}, \quad h_m^\star(x) = \frac{|g_m(\hat{\theta}_{U(x),\mathcal{D}_n})|}{Z},$$

where $Z = |M||g_M(\hat{\theta}_{U(x),\mathcal{D}_n})| + |m||g_m(\hat{\theta}_{U(x),\mathcal{D}_n})|$, *and* $|M|$ *and* $|m|$ *denote the cardinality of* M *and* m*, respectively.*

Proof. The trained model $\hat{\theta}_{h(x),\mathcal{D}_n} \in [-a, a]$ trivially holds for any $h(x) \in \mathcal{H}$. By the loss function definition in the toy setting, $g_M(\theta) = \theta + a$ and $g_m(\theta) = \theta - a, \forall\, \theta$. Thus, $|g_M(\theta)| + |g_m(\theta)| = 2a$ satisfies for $\forall\, \theta \in [-a, a]$. Since the gradient is scalar in the toy setting, $\hat{I}_{h(x)}(\hat{\theta}_{h(x),\mathcal{D}_n})$ is also scalar and same as unique eigenvalue, that is,

$$\hat{I}_{h(x)}(\hat{\theta}_{h(x),\mathcal{D}_n}) = |M| \cdot h_M(x) \cdot \{g_M(\hat{\theta}_{h(x),\mathcal{D}_n})\}^2 + |m| \cdot h_m(x) \cdot \{g_m(\hat{\theta}_{h(x),\mathcal{D}_n})\}^2 \quad \forall h(x) \in \mathcal{H}.$$

Thus, our problem is deciding $h_M(x)$ and $h_m(x)$ that maximize $|M| \cdot h_M(x) \cdot \{g_M(\hat{\theta}_{h(x),\mathcal{D}_n})\}^2 + |m| \cdot h_m(x) \cdot \{g_m(\hat{\theta}_{h(x),\mathcal{D}_n})\}^2$.

Below 3 constrains are hold for arbitrary $\theta \in [-a, a]$ and $h(x) \in \mathcal{H}$, because of the toy setting.

1. $|M| \cdot h_M(x) + |m| \cdot h_m(x) = 1$. (probability definition)

2. $|M| \cdot h_M(x) \cdot g_M(\hat{\theta}_{h(x),\mathcal{D}_n}) + |m| \cdot h_m(x) \cdot g_m(\hat{\theta}_{h(x),\mathcal{D}_n}) = 0.$
   Note that convex linear sum of the sample's gradient w.r.t. $h(x) = (h_M(x), h_m(x))$ is zero at the trained model $\hat{\theta}_{h(x),\mathcal{D}_n}$.

3. $|g_M(\theta)| + |g_m(\theta)| = 2a \Leftrightarrow g_M(\theta) - g_m(\theta) = 2a \Leftrightarrow g_M(\theta) = 2a + g_m(\theta).$
   Note that this is derived by the property of predefined loss function at $\theta \in [-a, a]$.

2nd constraint is equivalent to $|M| \cdot h_M(x) \cdot (2a + g_m(\hat{\theta}_{h(x),\mathcal{D}_n})) + |m| \cdot h_m(x) \cdot g_m(\hat{\theta}_{h(x),\mathcal{D}_n}) = 0. \Leftrightarrow g_m(\hat{\theta}_{h(x),\mathcal{D}_n}) = -2a|M| \cdot h_M(x)$. Because of the 3rd constraint, $g_M(\hat{\theta}_{h(x),\mathcal{D}_n}) = 2a(1 - |M| \cdot h_M(x))$. Then the objective is, maximizing

$$\begin{aligned} |M| \cdot h_M(x) \cdot \{2a(1 - |M| \cdot h_M(x))\}^2 + (1 - |M| \cdot h_M(x))\{2a|M| \cdot h_M(x)\}^2 \\ = 4a^2|M| \cdot h_M(x)(1 - |M| \cdot h_M(x)) \end{aligned} \tag{45}$$

equation 45 is maximized when $|M| \cdot h_M(x) = \frac{1}{2}$, and it means $|m| \cdot h_m(x) = \frac{1}{2}$. Thus, $h^\star_M(x) = \frac{|m|}{2|M| \cdot |m|}, h^\star_m(x) = \frac{|M|}{2|M| \cdot |m|}$. This result is related with the trained model $\hat{\theta}_{U(x),\mathcal{D}_n}$, where $U_M(x) = U_m(x) = \frac{1}{|M|+|m|}$. At $\hat{\theta}_{U(x),\mathcal{D}_n}$, $|M| g_M(\hat{\theta}_{U(x),\mathcal{D}_n}) + |m| g_m(\hat{\theta}_{U(x),\mathcal{D}_n}) = 0$ satisfies and this is equivalent to $|M| : |m| = |g_m(\hat{\theta}_{U(x),\mathcal{D}_n})| : |g_M(\hat{\theta}_{U(x),\mathcal{D}_n})|$. □

Thus, it is coincide with our intuition that setting the sampling probability h for set M and m with proportion to $|g_M(\hat{\theta}_{U(x),\mathcal{D}_n})|$ and $|g_m(\hat{\theta}_{U(x),\mathcal{D}_n})|$ helps to minimize the trace of the inverse empirical Fisher information.