Just Mix Once: Mixing Samples with Implicit Group Distribution

Giorgio Giannone*
Technical University of Denmark
gigi@dtu.dk

Serhii Havrylov
Amazon
havrys@amazon.co.uk

Jordan Massiah
Amazon
jormas@amazon.co.uk

Emine Yilmaz
Amazon, UCL
eminey@amazon.co.uk

Yunlong Jiao
Amazon
jyunlong@amazon.co.uk

Abstract

Recent work has unveiled how average generalization frequently relies on superficial patterns in data. The consequences are brittle models with poor performance in the presence of domain shift in group distribution at test time. When the subgroups in the training data are known, we can use tools from robust optimization to tackle the problem. However, group annotation and identification are time-consuming tasks, especially on large datasets. A recent line of research [12] is trying to solve this problem with implicit group distribution at training time, leveraging self-supervision and oversampling to improve generalization on minority groups. Following such ideas, we propose a new class-conditional variant of MixUp [21] for worst-group generalization, augmenting the training distribution with a continuous distribution of groups. Our method, called Just Mix Once (JM1), is domain-agnostic, computationally efficient, and performs on par or better than the state-of-the-art on worst-group generalization.

1 Introduction

In supervised learning, the goal is to fit a model on a training set to maximize a relevant global metric at test time. However, the optimization process can exploit spurious correlations between the target y and superficial input patterns in the data. We are interested when undesirable patterns can be categorized in groups g, and study the generalization performance in the presence of group distribution shift at test time. For example, in the case of colored MNIST (Fig. 1), a model can use information about the color (spurious correlation with the target class) instead of the shape to classify a digit (almost all 6s are blue at train time) during training. Then the model uses this bias at test time with a considerable decrease in generalization performance with a significant shift in the group distribution (for example, 6s are green and blue with equal proportion at test time). It is well-known that a powerful DNN model [7] tends to exploit easy superficial correlations, like texture, color, background, to solve a task if not constrained in some way [20, 8, 19, 2, 14, 5]. Similarly, in the presence of minority groups in a dataset, the model will tend to disregard such groups, relying on patterns that are frequent in majority groups. Recent work has shown how to tackle this problem in a supervised [16] and self-supervised [13] way. In this work, we focus on improving worst-group generalization with implicit group distribution. Self-supervision and oversampling have successfully been employed to deal with implicit group distribution in Just Train Twice (JTT) [12].

Contribution. Following this line of work, our contributions are: **I)** We propose a simple mechanism, called Just Mix Once (JM1), based on a class-conditional variant of MixUp [21] to improve

*Work done during an internship at Amazon, Alexa Shopping.

Workshop on Distribution Shifts - Connecting Methods and Applications at NeurIPS 2021

generalization on minority groups in the data. **II)** Our method does not rely on computationally expensive oversampling or the need to tune the oversampling rate. **III)** We perform extensive experiments with different levels of group annotation, demonstrating that JM1 outperforms the state-of-the-art (SOTA) on vision and language datasets.

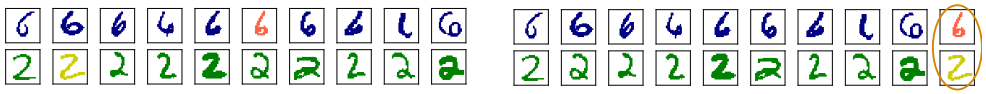

Figure 1: Groups Identification Phase. Left: training set. Right: training set partitioned. In the first phase, the training dynamics are exploited to split the data into two partitions. The assumption is that, among the samples identified as difficult (samples in the orange circle), there are typically samples from minority groups. The misclassification rate in the early stage of training is used as a clustering signal to estimate the group distribution: patterns that are superficially frequent in the data (e.g., color, texture, background) are easy to classify and have a small loss, partitioning a *majority* group; infrequent patterns (e.g., shape) are challenging to model and are misclassified during the early stage of training, partitioning a *minority* group.

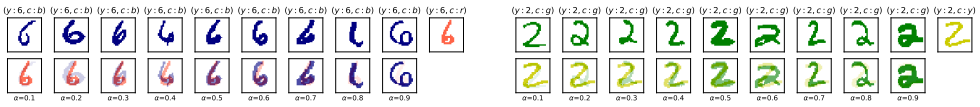

Figure 2: Class-conditional Mixing Phase. After the identification phase, we exploit label information to augment the training data, using a MixUp-inspired strategy to augment samples from different partitions. We use the label information to select two samples from different partitions, i.e., (h_g^i, y_g^i) and $(h_{\bar{g}}^j, y_{\bar{g}}^j)$ where $y_g^i = y_{\bar{g}}^j = y$, $g \neq \bar{g}$. Then we create a mixed-up sample (h_{mix}, y_{mix}) s.t. $h_{mix} = \alpha\, h_g^i + (1-\alpha)\, h_{\bar{g}}^j$, $y_{mix} = y$. Note that h can either be an input [21] or a learned representation [17]. This simple mechanism gives us a principled and domain-agnostic way to augment samples marginalizing the group information in the data: by sampling α, we augment the training data and build a continuous distribution of groups in the data. Therefore, the model cannot rely on frequent patterns because each sample has a "slightly different" group pattern.

2 Just Train Twice

Our work tackles the problem of worst-group generalization with implicit group distribution during training. This is different from the standard case, where we measure the average error among groups. In this section we discuss a closely related work, Just Train Twice (JTT) [12]. JTT is a SOTA approach that solves such a problem in a simple two-stage approach:

I) Groups Identification. The goal is to partition the data into two clusters: one with majority groups (frequent superficial patterns); and one with minority groups (uncommon patterns of interest). The assumption is that samples from minority patterns are difficult to model and frequently misclassified in early-stage of training. JTT uses the misclassified samples in the early stage of training as a signal to partition the data. Specifically, a supervised learner, parameterized by θ, is trained on the data $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^n$ using ERM loss (up to constant) $J_{\texttt{ERM}}(\theta) := \sum_{(x,y)\in\mathcal{D}} l(x, y; \theta)$ with early stopping. To avoid overfitting, a small validation set with group annotation is employed to select the best identification epoch T in order to create appropriate partitions. Then misclassified samples (samples with with high loss [13]) are saved in a buffer $\mathcal{B} := \{(x_b, y_b) \text{ s.t. } \hat{f}_T(x_b) \neq y_b\}$.

II) Groups Re-weighting. Once the partitions are identified, the same learner is trained, for a second time, on a re-weighted version of the data, where samples in $\mathcal{B}$ are oversampled $\lambda \gg 1$ times. The hope is that if the examples in the error set $\mathcal{B}$ come from challenging groups, such as those where the spurious correlation does not hold, then up-weighting them will lead to better worst-group performance [12]. Specifically, the loss for phase II) can be written as (up to constant): $J_{\text{JTT}}(\theta) := \lambda \sum_{(x_b,y_b)\in\mathcal{B}} l(x_b, y_b; \theta) + \sum_{(x_{\bar{b}},y_{\bar{b}})\notin\mathcal{B}} l(x_{\bar{b}}, y_{\bar{b}}; \theta)$, where λ is the up-sampling rate for samples in $\mathcal{B}$ identified from phase I).

Limitations. For JTT to work, the authors emphasize that the two hyper-parameters are crucial to guarantee the success of JTT: the number of identification epochs T in phase I) and re-weighting λ in phase II). This requires careful hyper-parameter tuning using a small annotated validation set. Another caveat is that naively oversampling a portion of the training data $\lambda \gg 1$ times can make training JTT much slower than ERM.

3 Just Mix Once

We address some of the limitations of JTT by proposing Just Mix Once (JM1). Remind that our goal is to improve the classification accuracy of minority groups, (with or) without explicit annotation on the groups at training time.

I) (Optional) Groups Identification. If the explicit group annotation is unavailable on the training set, similarly to phase I) of JTT, we assume that we have annotated groups on a small validation set and resort to using the same self-supervised signal [13, 1] based on the misclassification rate (and the loss magnitude) in the early stage of training like JTT. Fig. 1 illustrates how relevant samples from the minority groups are identified.

II) Class-conditional Mixing. To generalize on minority groups at test time, a model should not rely on spurious correlations during training (e.g., texture or color) but on the signal of interest (e.g., shape). Unlike the oversampling approach taken by JTT, we resort to a better augmentation (Fig. 2) to improve generalization for low-density groups. Our mixing strategy is inspired by the MixUp strategy [17, 21], and we propose a novel class-conditional variant that achieves robust worst-group generalization. Specifically, for any two samples $(h_g^i, h_{\bar{g}}^j)$ in the input (or representation) space from the two partitions $g, \bar{g}$ (i.e., difficult/misclassified/minority group g vs the other majority group $\bar{g}$) with the same class label $y_g^i = y_{\bar{g}}^j = y$, we mix them up using a convex interpolation [17, 21]:

$$h_{mix} = \alpha h_g^i + (1-\alpha) h_{\bar{g}}^j, \quad y_{mix} = y, \tag{1}$$

where α is the mixing parameter (details of how to choose α are deferred to Sec. 4). Notably, we empirically show that naively implementing the standard MixUp without the proposed two-stage approach fails to generalize in terms of worst-group performance (Fig. 3).

How JM1 works. We briefly discuss how our mixing strategy enables JM1 to build a "continuous" spectrum of groups in data by sampling α. Assume the training data $D = \{x^i, y^i\}_{i=1}^n$ are partitioned into (non-overlapping) groups $\mathcal{G}_1, \ldots, \mathcal{G}_m$, through sub-populations represented by tuples (y, c) of label y and confounding factor(s) c, we can define the following per-group and group-average loss:

$$J(\theta; \mathcal{G}) := \frac{1}{|\mathcal{G}|} \sum_{(x_g, y_g) \in \mathcal{G}} l(x_g, y_g; \theta), \quad J(\theta) := \frac{1}{m} \sum_{k=1}^{m} J(\theta; \mathcal{G}_k). \tag{2}$$

Notice that in case $m = n$, each group contains a single data point, and the group-average loss collapses to the standard ERM loss $J_{\texttt{ERM}}$. Note that, if we define a uniform per-group loss weight $p_k = 1/m, k = 1, \ldots, m$, we can re-write the group-average loss as:

$$J(\theta) = \sum_k p_k J(\theta; \mathcal{G}_k) = \mathbb{E}_{p(\mathcal{G})} J(\theta; \mathcal{G}). \tag{3}$$

Distributional Robust Optimization (DRO) [16] aims to optimize: $J_{\texttt{DRO}} = \texttt{max}_k J(\theta, \mathcal{G}_k)$, which corresponds to a *pointwise* Dirac distribution $p_k = 1$ if $k = \arg\max_k J(\theta; \mathcal{G}_k)$ else 0 in Eq. 3. Using MixUp with α mixing rate, we generate new samples drawn from a continuous mixture (Eq. 1), which has as limiting case the training distribution. In this view, we can interpret each mixed-up sample as drawn from a "mixed-up group" $\mathcal{G}_\alpha$ parametrized by α. This means, assuming an oracle partitioning in phase I), JM1 mixes a majority and a minority group in the data at each iteration while generating a *continuous* group distribution. We denote by $J(\theta; \mathcal{G}_\alpha)$ our per-group loss for a "mixed-up" group $\mathcal{G}_\alpha$, and write the group-average loss by marginalizing out the group distribution:

$$J_{\texttt{mix}}(\theta) := \mathbb{E}_\alpha \mathbb{E}_{p(\mathcal{G};\alpha)}[J(\theta; \mathcal{G})] = \int_\alpha \int_{\mathcal{G}_\alpha} p(\alpha) p(\mathcal{G}; \alpha) J(\theta; \mathcal{G}) d\alpha d\mathcal{G}. \tag{4}$$

Note that, first sampling α and then applying MixUp with rate α (Eq. 1) is equivalent to drawing samples directly from a mixed-up group $\mathcal{G}_\alpha$ drawn implicitly from $p(\mathcal{G}; \alpha)$. Therefore, we can approximate Eq. 4 by a simple MC sampling process in practice: sample α, apply MixUp [21, 17] with rate α, compute the loss using the mixed-up samples; repeat for each mini-batch in SGD updates. This simple mechanism enables to augment data by marginalizing group information, hence the training cannot rely on spurious group patterns because each sample belongs to a "slightly different" group. For intuitions behind the class-conditional mixing of JM1's phase II) in the presence of minority groups, refer to the visualization in Appendix C. We leave thorough theoretical study of JM1 to future work.

4 Experiments

Benchmark Datasets. Our focus for all the experiments is to improve worst-group performance for classification problems across vision and language domains (with or) without group annotation at train time. We report experiments on the benchmark proposed in [16]. The benchmark consists of two vision and one language dataset (Appendix B). In these datasets, the model can easily exploit superficial, majority group patterns (background in CUB, sex in CelebA, negation in MultiNLI) to solve the classification tasks at train time. However, when tested on a set with a different group distribution (same groups but represented in significantly different proportions), the performance drops, showing a lack of generalization capacity. JTT [12]) is SOTA on this benchmark, and we primarily compare JM1 to JTT.

JM1 Implementation Details. For JM1, we initially found that mixing groups by sampling $\alpha \sim U(0,1)$ uniformly, even though improving worst-group accuracy over ERM, underperforms compared to JTT. Therefore, we use a mixing strategy with a slight emphasis on the region closer to the minority groups and propose a coupled strategy: we sample $\alpha \sim U(0,1)$ for half of the epochs, and $\alpha \sim Beta(2,5)$ for the other half. This simple heuristic ensures that the full mixing domain is spanned, with a focus on the region closer to the misclassified samples in phase I). We use this coupled sampling strategy for CUB and CelebA. For MultiNLI, we sample α only from the uniform prior because the mixing phase runs for a very small number of epochs. More training details of JM1 are described in Appendix E. It is worth noting that: I) like JTT, JM1 is domain-agnostic and can be applied to vision and language datasets with minimal architectural changes; II) unlike JTT's oversampling approach, JM1 is a highly scalable approach, which removes the need for tuning the oversampling rate and only introduces an interpolation operation between samples with little computational overhead. Note that we can apply JM1 to mix samples at different levels of abstraction: either the input [21] or the representation space [17]. As the comparison of different MixUp levels is not the focus of our study, we refer the readers to Appendix D for additional ablations.

Worst-Group Generalization. The first set of experiments evaluate JM1 vs JTT in the setting of implicit group distribution on the training set. We assume that the models have access to a small validation set with group annotation to guide group identification in phase I). To ensure a fair comparison between methods, we use the same identified set for both JTT and JM1 in all our experiments. Results in Table 1 show that our approach is competitive with the SOTA on improving worst-group performance on all three datasets with comparable average accuracy. In Table 2 we report the per-group accuracy on CUB for JTT and JM1 under the same setting. We see that, compared to JTT, not only does JM1 improve worst-group accuracy, but it improves accuracy on both of the minority groups with little to no loss in performance on the other two majority groups. Overall, JM1 is competitive or better than SOTA in terms of worst-group performance on this benchmark.

Table 1: Results on CUB, CelebA and MultiNLI datasets for average accuracy and worst-group accuracy. JM1 performs comparably or better than the SOTA in worst-group performance with implicit group distribution, closing gap to upper anchor GroupDRO requiring full group annotations on the training set. JM1 confidence intervals are evaluated on 5 different runs.

	CUB		CelebA		MultiNLI		group
	avg acc	worst acc	avg acc	worst acc	avg acc	worst acc	labels
ERM [12]	97.6	72.6	95.6	47.2	82.4	67.9	✗
CVaR-DRO [10]	96.5	69.5	82.4	64.4	82.0	68.0	✗
LfF [13]	97.3	75.2	86.0	70.6	80.8	70.2	✗
EIIL [5]	96.9	78.6	-	-	-	-	✗
JTT [12]	93.3	86.7	88.0	81.1	78.6	**72.6**	✗
JM1 (ours)	93.0 ± 0.4	$\mathbf{87.5 \pm 0.6}$	86.6 ± 0.6	$\mathbf{83.3 \pm 0.7}$	80.3 ± 0.2	$\mathbf{72.5 \pm 0.6}$	✗
GroupDRO [16]	93.5	91.4	92.9	88.9	81.4	77.7	✓

Oracle Groups. The second set of experiments compare JM1 vs JTT in a different scenario: we assume to have access to the ground-truth minority group assignment on the training set. Note that this does not mean the full group annotation: we only need to know which sample has a group-conflicting confounder against its label (see a comparison of JM1-inspired GroupJM1 and GroupDRO in Appendix D in a setting given full group annotation on training data). Now we can use such oracle majority vs minority groups in place of the identified set from phase I). As noted in Appendix C.2

Table 2: Per-group accuracy on CUB dataset. The goal is to classify land birds y_l vs water birds y_w in the presence of a confounders: land background c_l vs water background c_w. JM1 is competitive with JTT improves accuracy on both minority groups (y_l, c_w) and (y_w, c_l) and competitive on the other two majority groups.

	(y_l, c_l)	(y_l, c_w)	(y_w, c_l)	(y_w, c_w)	avg acc	worst acc
JTT	94.3	86.7	87.5	91.6	93.3	86.7
JM1	94.2	88.5	88.9	90.5	93.1	**88.5**

in [12], the performance of JTT drops if not using identified error sets from phase I), since JTT can excessively oversample the minority group. Given its inner working of mixing groups rather than oversampling, we expect JM1 to outperform JTT in this oracle configuration. Table 3 validates our intuition. In fact, JM1 (oracle) outperforms JTT (oracle) on both datasets.

Class-conditional MixUp. We compare the proposed JM1 with standard (unconditional) MixUp and a class-conditional MixUp baseline, where we mix samples from random groups (i.e., JM1 without group identification phase). Note that we use an oracle configuration for simplicity in these experiments. Fig. 3 shows that, while standard MixUp fails to improve worst-group performance, the proposed mixing strategy performs significantly better than both MixUp baselines by a large margin, despite that all three methods perform very similarly in terms of average performance.

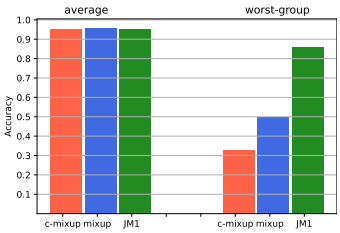

Figure 3: Results on CUB dataset JM1 vs class-conditional (c-MixUp) and standard MixUp baselines. Note that standard MixUp fails to improve worst-group accuracy.

Table 3: Results on CUB and CelebA datasets in an oracle configuration, where we replace identified sets with ground-truth majority and minority groups. JM1 with class-conditional mixing generalizes better than oversampling-based JTT.

	CUB		CelebA	
	avg	worst	avg	worst
JTT (oracle) [12]	92.8	75.9	93.4	57.2
JM1 (oracle)	95.0	**86.1**	92.3	**85.6**

Robustness. While JTT and JM1 do not use full group annotation on the training set, they both rely on group annotation on a small validation set to guide group identification (i.e., choosing early stopping epoch). A natural question to investigate is if we can remove the need for group annotation at all, and JM1 can still perform reasonably well. To that end, we test the robustness of JM1 to different identification epochs/sets in phase I) using the CUB dataset. Specifically, we checkpoint three different identification scenarios (20, 40, 60 epochs) in phase I) for both JM1 and JTT. Then, we select for JM1 the best phase II) model in terms of average accuracy, while we tune the best-performing JTT using group annotation on a validation set in terms of worst-group accuracy. Note that JM1 is now completely free from group annotations. Table 4 shows that JM1 without group annotation at all loses only around 5% compared to the full-suite JTT with access to group annotation on a validation set. This is a preliminary result that indicates the potential of training group-robust models in a fully self-supervised manner. We can hope to achieve worst-group robust generalization with no more data requirement than what we need to train typical ML models.

Table 4: Results on CUB dataset for average accuracy and worst-group accuracy without validation set. We checkpoint three identification epochs in the phase I) (20, 40, 60 epochs for identification). We select JM1 using average accuracy, and tune JTT using worst-group accuracy. JM1 handles reasonably well misspecification of the identified error set. (S: number of error samples / possibly spurious correlations detected. NS: number of not spurious correlations detected. P: precision. R: recall.)

	JM1		JTT-best		$\Delta\%$					
	avg	worst	avg	worst	avg	worst	S	NS	P	R
20id	92.3	72.9	78.9	67.5	+ 17.0	+ 8.0	107	290	0.269	0.445
40id	93.6	83.0	90.0	86.9	+ 4.0	- 4.5	115	153	0.429	0.479
60id	93.2	82.2	90.3	86.7	+ 3.2	- 5.2	110	128	0.462	0.458

References

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019.

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

[3] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

[4] Luigi Carratino, Moustapha Cissé, Rodolphe Jenatton, and Jean-Philippe Vert. On mixup regularization. *arXiv preprint arXiv:2006.06049*, 2020.

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

[6] Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*, 2015.

[7] 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. *arXiv preprint arXiv:1811.12231*, 2018.

[8] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29:3315–3323, 2016.

[9] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed H Chi. Fairness without demographics through adversarially reweighted learning. *arXiv preprint arXiv:2006.13114*, 2020.

[10] Daniel Levy, Yair Carmon, John C Duchi, and Aaron Sidford. Large-scale methods for distributionally robust optimization. *arXiv preprint arXiv:2010.05893*, 2020.

[11] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020.

[12] 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.

[13] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: Training debiased classifier from biased classifier. *arXiv preprint arXiv:2007.02561*, 2020.

[14] Luca Oneto and Silvia Chiappa. Fairness in machine learning. *Recent Trends in Learning From Data*, pages 155–196, 2020.

[15] Judea Pearl. *Causality*. Cambridge university press, 2009.

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

[17] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019.

[18] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P Gummadi. Fairness constraints: A flexible approach for fair classification. *The Journal of Machine Learning Research*, 20(1):2737–2778, 2019.

[19] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics*, pages 962–970. PMLR, 2017.

[20] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International conference on machine learning*, pages 325–333. PMLR, 2013.

[21] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[22] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819*, 2020.

A Related Work

Robust Classification and Implicit Group Distribution. Optimizing average accuracy with overparametrized neural networks tends to get rid of minority group information. This fact is related to how the model is trained and the way the model exploits superficial correlations in the data [7]. Recently approaches to preserve group information have been proposed [16] when full group annotation is available. However assuming sub-population annotation in large diverse datasets is unrealistic both in terms of time effort and most important because the process is prone to errors. Learning from Failure [13] proposes an annotation-free and domain agnostic procedure to improve worst-group performance. The authors propose a self-supervised approach to identify bias-aligned (majority) vs bias-contrastive (minority) groups in the data. The self-supervised signal consists of the per-sample loss magnitude in the early stage of training; Then the dataset is re-weighted based on the loss magnitude and the procedure applied multiple times during training. Inspired by this approach Just Train Twice [12] proposes a similar solution, where the identification phase is performed one time, and then a reweighting phase is trained. Just Train Twice is effective but relies on an annotated and noise-free validation set for the groups. Invariant Learning [2, 5] is another approach that can be used to improve worst-group performance. The goal is to obtain a learner that performs well in different environments, where in each environment we have the same classes but different confounders.

Equality of Opportunities in Machine Learning. Mixing information from different groups generates a mixture of distributions with a continuous amount of groups. Having an infinite collection of groups means that the model cannot rely on group information during training and samples in different groups should be considered similarly. In this setting, we assume to have full annotation for the groups, and robust optimization methods are employed. The supervised scenario is closely related to the fairness one [14]: wherewith distribution shift we want to improve the worst-group generalization, in fairness we want to learn features and distributions that are invariant to different groups, using explicit constraints [19, 18, 2], auxiliary tasks [6, 9], and causal structure [15, 14]. In the causality literature [15] the group information is interpreted as a confounder. If we have access to the confounder (group annotation) we can control for it using intervention. The mixing procedure is augmenting the training data with plausible scenarios not present in the data. This approach generates counterfactual scenarios interpolating groups in the data and exploits these imaginative scenarios to account for minority groups. In the case of implicit groups, confounder effects are unknown. We can still use our approach exploiting loss magnitude and pattern formation during early training.

Self-Supervised Learning. Recently a large corpus of works dealt with self-supervised learning. With the amount and complexity of the data increasing exponentially, the availability of annotation is decreasing fast. Self-supervised learning tries to exploit structure in the data to cluster data and improve generalization. Loss magnitude [13, 7, 12, 1] can be used to cluster data. MixUp [21] is frequently employed to improve generalization and robustness to noise. The idea is to increase the variety in the data interpolating the training distribution in sample or representation space [17]. MixUp based methods have been extensively used for semi-supervised learning [11, 3, 1, 10] and improve generalization [22, 4]. Notice that in this work the self-supervision is wrt the group annotation and not label annotation as usual.

B Benchmark

We report experiments on the benchmark proposed in [16]. The benchmark consists of two vision and one language dataset. In all three datasets, there are factors or confounders (background, sex, hair color) that strongly correlate with the label information. In particular, for the image datasets: in Waterbirds (CUB) the task is to classify land birds vs water birds. The class information is spuriously correlated with the background (land vs water) generating 4 relevant groups in the training set; in CelebA the task is to classify blond vs not-blond hairs. The class information is spuriously correlated with the sex information (male vs female) generating 4 relevant groups. For the language dataset, MultiNLI, the task is to classify a reasoning process (two sentences) as entailment, contradiction, or neutral. The class information is spuriously correlated with the presence of negation in the sentences. In all these datasets the model can easily exploit superficial, majority patterns (background, sex, negation) to solve the tasks at train time. However, when tested on a set with a different group distribution (same groups but represented in different proportions), the performance drops, showing a lack of generalization capacity.

Figure 4: Figure adapted from [12]. Examples from the benchmarks. Label information y is spuriously correlated with confounding factors a.

Waterbirds (CUB). The task is to classify land birds y_l and water birds y_w. The label information is correlated with a confounding factor: land background c_l and water background c_w. The majority groups are (land bird, land background) - (y_l, c_l) and (water bird, water background) - (y_w, c_w). The minority groups are (land bird, water background) - (y_l, c_w) and (water bird, land background) - (y_w, c_l). During training, the minority groups are present in 5% of the data. At test time the groups are evenly distributed in the data.

CelebA. The task is to classify blond person y_b and not blond person y_{nb}. The label information is correlated with a confounding factor: male c_m and female c_f. The majority groups are (not blond, male) - (y_{nb}, c_m) and (blond, female) - (y_b, c_f). The minority groups are (not blond, female) - (y_{nb}, c_f) and (blond, male) - (y_b, c_m).

MultiNLI. The task is to classify an inference procedure as entailment y_e, contradiction y_c and neutral y_n. The label information is correlated with a confounding factor: the presence of a negation c_{neg} and absence of a negation c_{nneg} in the reasoning.

C Modelling the Augmented Group Distribution

In this section, we present a toy visualization and a simple idea to better understand the mixing procedure versus an oversampling strategy. Given two classes (blue and brown points) some of the groups are underrepresented on the train set (Fig. 5). These underrepresented groups are the minority groups. On the test set, all the groups are equally represented (Fig. 6). JTT oversamples the minority groups (Fig. 7) and trains on the augmented distribution. JM1 mixed minority and majority groups (Fig. 8) and trains on the augmented distribution. We see that both methods generate a training distribution closer to the target one in Fig. 6.

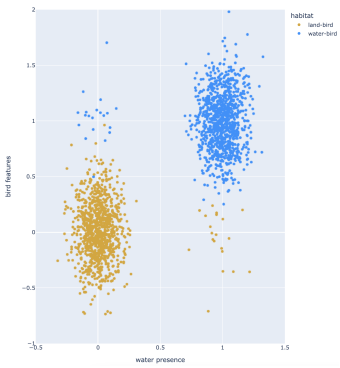

Figure 5: Source domain.

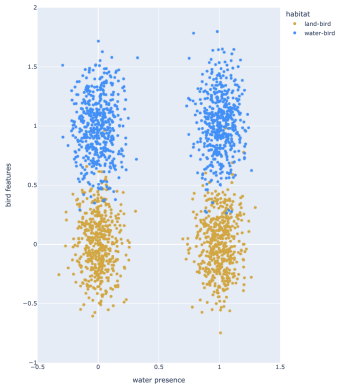

Figure 6: Target domain.

Mixing Groups. Consider that group is encoded as a 2d variable g that is represented jointly by two 1d variables confounder $c \in \{0, 1\}$ and label $y \in \{0, 1\}$. Suppose that a 2d sample representation

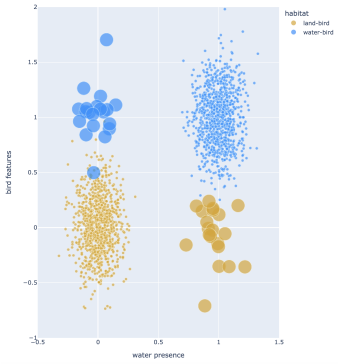

Figure 7: Oversampling (JTT).

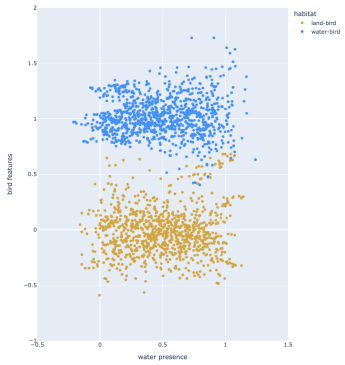

Figure 8: Mixing (JM1).

is sampled from the conditional distribution $h|c, y \sim N((c, y), 1)$ where we denote by $g := (c, y)$. Since minority groups $(c = 0, y = 1)$ and $(c = 1, y = 0)$ are only present in the source domain (Fig. 5) but not in the target domain (Fig. 6), the decision boundary if naively trained on the source domain will perform very poorly in the target domain. For a pair of samples from a minority group g (under-represented) and a majority group $\bar{g}$ (over-represented) respectively, now that $h \sim N(g, 1)$ and $\bar{h} \sim N(\bar{g}, 1)$, we have that the mixed-up sample with rate α follow an augmented distribution:

$$h_{mix} = \alpha\, h + (1 - \alpha)\, \bar{h} \sim N(\underbrace{\alpha\, g + (1 - \alpha)\, \bar{g}}_{:= g_{mix}}, \underbrace{\alpha^2 + (1 - \alpha)^2}_{\leq 1}).$$

We can see that, given a continuous distribution for α, $g_{\texttt{mix}}$ is a sample from a continuous distribution that augments the training groups by convex combination. It is easy to see that a group sampled from the mixing distribution conditional on the same class y (Fig. 8) will be closer to the target than the source domain, achieving a similar effect of the oversampling approach taken by JTT (Fig. 7). It is also clear that, without the class-conditional constraint, the mixing will result in an augmented source distribution whose decision boundary will be inconsistent with the target distribution.

D Additional Experiments

In this section, we report additional ablation experiments.

In Section 4 we have considered only the minority groups in the oracle setting, but potentially we can have access to full group annotation on the train set. Consequently, methods like GroupDRO [16] can be employed. We compare a variant of JM1, called GroupJM1, and GroupDRO. In Table 5 we consider the configuration with full group annotation on the train set and we evaluate a variant of JM1, GroupJM1, with GroupDRO [16]. GroupJM1 is competitive in this scenario, corroborating the idea that the mixing procedure is a general approach to improve worst-group accuracy. In Figure 9 we investigate the performance of JM1 with and without class-conditional constraint. We see that empirically the conditional JM1 performs better than the unconditional one. The worst-group accuracy decreases increasing the dataset complexity (in terms of the number of samples, classes, and groups) for both models. In Table 6 we consider the performance of JM1 mixing samples at different levels of abstraction, where the layer indicates mixing in input space, inside the encoder, in output space, or choosing a layer at random. We also test different sampling distributions. In Figure 10 and Table 10 we study the behaviour of the models varying the number of identification epochs in phase I).

Table 5: Experiments using GroupDRO and JM1 variants. We assume fine-grained group annotation on the train set. We find the worst performing group in each mini batch and use such information to improve JM1. We samples $\alpha \sim U(0,1)$. GroupDRO results from [16].

	CUB		CelebA		MultiNLI		$\mathcal{L}$
	avg	worst	avg	worst	avg	worst	
GroupDRO	93.5	**91.4**	92.9	88.9	81.4	77.7	DRO
GroupJM1	90.9	90.5	92.0	86.1	82.1	**81.5**	ERM
GroupJM1	93.3	**91.3**	93.0	**90.0**	81.9	79.3	DRO

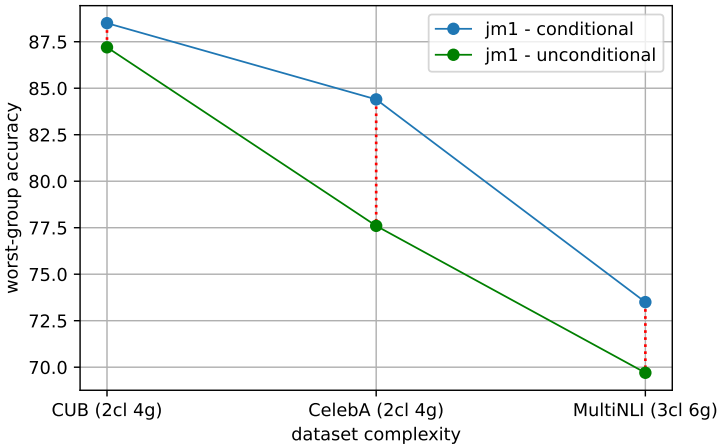

Figure 9: Unconditional vs conditional JM1 on the benchmark. The worst-group performance decreases increasing the dataset complexity (dataset size, number of groups, variety of the data).

Table 6: Ablation on CUB dataset split by group. The goal is to classify land birds y_l vs water birds y_w in the presence of a confounders: land background c_l vs water background c_w. JM1+ is a variant of JM1 with a fixed amount of oversampling.

	layer	α	(y_l, c_l)	(y_l, c_w)	(y_w, c_l)	(y_w, c_w)	worst acc
JTT	-	-	94.3	86.7	87.5	91.6	86.7
JM1	input	U	98.4	84.2	77.7	93.0	77.7
JM1	input	$U(0.5)$	96.7	84.8	86.8	93.5	84.8
JM1	early	U	97.5	84.7	80.1	92.2	80.1
JM1	output	U	97.3	85.9	84.7	92.7	84.7
JM1	random	U	93.8	87.8	89.7	90.8	87.8
JM1	random	U/B	94.2	88.5	88.9	90.5	**88.5**
JM1+	input	U	91.9	89.4	90.5	84.9	84.9
JM1+	input	U/B	96.3	87.6	88.8	93.3	87.6
JM1+	early	U	92.5	87.9	87.5	89.9	87.5
JM1+	random	U/B	94.0	89.6	90.7	90.2	**89.6**

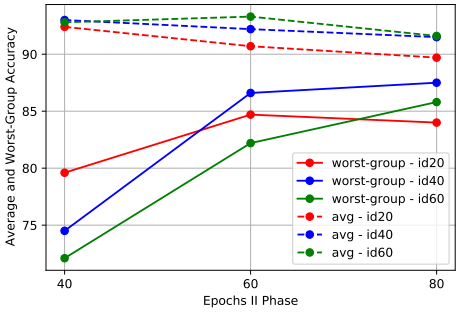

Figure 10: Worst-group and average accuracy for different setups. On the x-axis we have different epochs we evaluate the model performance in the II phase. Each line represents a different epoch to identify samples in the I phase.

Figure 11: Results on CUB for different identification epochs. For JM1-avg we choose the model based on average accuracy.

	JM1-avg		JM1-robust	
	avg	worst	avg	worst
10id	86.6	39.7	80.6	64.2
20id	92.2	72.7	89.8	84.4
30id	93.6	81.8	90.8	86.4
40id	93.7	81.8	90.3	87.5
50id	94.1	82.6	90.8	87.5
60id	93.3	82.7	91.1	87.0
70id	93.3	79.1	90.2	85.8
80id	92.8	79.3	90.2	85.0
90id	93.0	78.0	89.9	84.7
100id	91.2	69.0	86.3	81.3

E Experimental Details

Table 7: Training details for JM1. CE: cross-entropy. U: uniform distribution $U(0, 1)$. B: beta distribution $B(2, 5)$. For all the other hyper-parameters we use the one proposed in [12]

	CUB	CelebA	MultiNLI	CUB (oracle)	CelebA (oracle)
Batch Size	32	64	32	32	64
Identification Epoch	40	1	2	-	-
Early stopping	✓	✓	✓	✓	✓
Epochs (I Phase)	200	40	4	-	-
Epochs (II Phase)	200	40	4	200	80
α	U/B	U/B	U	U/B	U/B
Learning Rate	1e-5	1e-5	1e-5	1e-5	1e-5
$\mathcal{L}$	CE	CE	CE	CE	CE
Number Groups	4	4	6	4	4
Number Classes	2	2	3	2	2
Mixing Layer	random	early layer	output	random	early layer
Pretrained Encoder	ResNet50	ResNet50	BERT	ResNet50	ResNet50
Weight Decay	1.0	0.1	0.1	1.0	0.1