UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[Re] Reproducibility study of "Focus On The Common Good: Group Distributional Robustness Follows"

Walter Simoncini[1], Ioanna Gogou[1], Marta Freixo Lopes[1], and Ron Kremer[1]

[1]University of Amsterdam, Amsterdam, The Netherlands

Edited by
(Editor)

Received
–

Published
–

DOI
–

Reproducibility Summary

Scope of Reproducibility — This paper attempts to reproduce the main claims of "Focus On The Common Good: Group Distributional Robustness Follows" by Piratla et al., which introduces Common Gradient Descent (CGD), a novel optimization algorithm for handling spurious correlations and sub-population shifts. We have identified three central claims: (I) CGD is more robust than Group-DRO and leads to the largest average loss decrease across all groups (II) CGD generalizes better across all groups in comparison to ERM, and (III) CGD monotonically decreases the group-average loss.

Methodology — The experiments of this paper are based on the open source implementation of CGD released by the authors, which required some modifications to work with the latest version of the WILDS framework.

Results — The results from our experiments were overall in line with the paper. We show that CGD outperforms Group-DRO on synthetic datasets with induced spurious correlations, but the benefits of CGD are not clear in a real-world setting. Beyond the results of the original paper, our attempt to empirically verify the mathematical proof of the authors that CGD monotonically decreases the loss was not conclusive.

What was easy — The implementation from the original paper was available on GitHub with detailed instructions provided in the documentation. It was also relatively easy to introduce additional datasets and algorithms to the WILDS codebase.

What was difficult — The CGD implementation and several experiments could not be run out-of-the-box and required major modifications to run with the latest version of WILDS. The majority of the hyperparameter values for the experiments were not clearly stated. Lastly, the experiments were computationally expensive and required 440 GPU hours.

Communication with original authors — We reached out to the original authors to request additional information about the hyperparameter values and the implementation of some experiments. The authors promptly responded with sources for the hyperparameters, useful information about WILDS and provided some missing parts of the code. Overall, the communications were timely and effective.

Copyright © 2023 W. Simoncini et al., released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to Walter Simoncini (walter.simoncini@student.uva.nl)
The authors have declared that no competing interests exist.
Code is available at https://github.com/WalterSimoncini/CGD-Reproduction. – SWH swh:1:dir:4a89288fc050158c419caee05af572cad7b71a12.
Open peer review is available at https://openreview.net/forum?id=ye8PftiQLQ.

1 Introduction

In recent years, deep neural networks have become state-of-the-art solutions for many tasks. However, for some, they tend to achieve high overall prediction accuracy at the cost of mispredicting samples from minority groups. This is dangerous for automatic decision systems that make critical decisions. For instance, models trained with Empirical Risk Minimization (ERM) are particularly susceptible to this issue. One of the key reasons for this behavior, identified by both Sagawa et al.[1] and Koh et al.[2], is the existence of spurious correlations between the features and labels of the majority group, which may not exist or correlate negatively for minority groups. To overcome this, several new algorithms have been proposed, such as Group Distributionally Robust Optimization (Group-DRO or G-DRO) [1], and Common Gradient Descent (CGD) [3]. The first tackles the problem by training on the group with the largest training loss. Nevertheless, Piratla, Netrapalli, and Sarawagi[3] observed that this might lead to imbalanced training due to increased loss in the other groups. In contrast, CGD trains on the group which minimizes the loss across all groups. The authors claim that such an approach models inter-group interactions and addresses spurious correlations. This report aims to verify the claims of the authors by reproducing their findings and performing additional experiments. Our contribution can be summarized as follows:

- We reproduce both the qualitative and quantitative experiments to identify which claims can be verified. We also quantify the computational and development cost needed.
- We update the CGD code to make it compatible with the latest version of the WILDS framework [2] and document the steps taken to reproduce the paper, making future reproductions easier and more accessible.
- We implemented the code required to run experiments on the MultiNLI [4] dataset.

2 Scope of reproducibility

The paper "Focus On The Common Good: Group Distributional Robustness Follows" [3] attempts to tackle the problems of spurious correlations and sub-population shift with a new optimization algorithm: Common Gradient Descent (CGD). CGD optimizes the model using the group "whose gradients lead to the largest decrease in average training loss over all groups" [3]. In this reproducibility study, we attempt to validate the following claims made by the authors:

- CGD is more robust than Group-DRO in the presence of spurious features: by focusing on the group that leads to the largest loss decrease across all groups, CGD is robust against spurious features.
- CGD generalizes better across all groups in comparison to ERM: a model trained with CGD should perform better on minority groups while achieving a comparable average accuracy.
- CGD monotonically decreases the macro/group-average loss: the authors prove this claim mathematically, showing that CGD finds first-order stationary points. We attempt to validate this claim empirically over three datasets used by the paper.

In this reproducibility study, due to resource and time constraints, we decided to compare the performance of CGD only against ERM and Group-DRO. This decision is also motivated by the fact that the other robustness algorithms mentioned in the paper were not discussed as thoroughly. We consider this subset of algorithms sufficient to validate

the claims stated above. Moreover, for the real-world datasets from the WILDS benchmark [2], we only trained with CGD, given that the documented results with the other algorithms were highly correlated with the values reported in the official WILDS paper [2] and the WILDS benchmark. Therefore, we felt these two were reliable sources for validating these results.

3 Methodology

To conduct our experiments we utilized the code for CGD which was released by the authors on their GitHub repository [1]. The available implementation was designed to be run using version 1.2.2 of the WILDS framework [2]. In the months following the code release, WILDS underwent a major update [5] to version 2.0. In order to make CGD compatible with the new framework version, we had to implement some minor tweaks, especially for datasets with disjoint groups for the training, validation, and test sets. The authors also provided the initialization code for the datasets used in the qualitative evaluation of the algorithm. The code for the datasets used in the quantitative evaluation was included in the WILDS repository, except for MultiNLI, which we implemented using the code in the Group-DRO repository [1] as indicated by the authors.

3.1 Algorithm descriptions

This section describes the algorithms used to validate the claims made by the authors, namely ERM, Group-DRO, and CGD. We closely follow the notation used in the paper, which defines $\mathcal{X}$, $\mathcal{Y}$ as the feature and label spaces, and $\mathcal{G}$ as a set of non-overlapping k groups, each composed of n_i observations distributed as $P_i(\mathcal{X}, \mathcal{Y})$. For each group i, we define $\ell_i(\theta) = \mathbb{E}_{(x,y)\sim P_i(\mathcal{X},\mathcal{Y})}\mathcal{L}(x, y; f_\theta)$ as group loss, where f_θ is a classifier, θ are the model parameters and $\mathcal{L}$ is an appropriate loss. The baseline algorithm is ERM, which minimizes the expected training loss.

$$\theta^{t+1}_{\text{ERM}} = \underset{\theta}{\text{argmin}} \left\{\mathbb{E}_{(x,y)\sim\hat{P}}[\ell(\theta^t)]\right\} , \tag{1}$$

where $\hat{P}$ is the empirical distribution over the training data and ℓ the loss for the whole training set without considering the groups. The ERM formulation implicitly assigns a higher weight to majority groups. It also encourages the model to exploit spurious correlations that work well for predicting the label of majority groups, achieving high average test accuracies at the expense of minority groups. To overcome this issue, Sagawa et al.[1] proposed Group-DRO, which, at each step, trains on the group with the worst training loss j^*:

$$j^* = \underset{i\in\mathcal{G}}{\text{argmax}}\, \ell_i(\theta) \qquad \Rightarrow \qquad \theta^{t+1}_{\text{G-DRO}} = \theta^t - \eta\nabla\ell_{j^*}(\theta^t) \tag{2}$$

While Group-DRO has a better performance on minority groups when compared to ERM, it may overfit on them, jeopardizing the average loss over all groups. To solve this problem, Piratla, Netrapalli, and Sarawagi[3] proposed Common Gradient Descent (CGD), which, at each step, picks the group which minimizes the overall loss across all groups as follows:

$$j^* = \underset{j}{\text{argmin}}\{\sum_i \ell_i[\theta^t - \eta\nabla_\theta\ell_j(\theta^t)]\} \qquad \Rightarrow \qquad \theta^{t+1}_{\text{CGD}} = \theta^t - \eta\sum_i \alpha_i^{t+1}\nabla_\theta\ell_i(\theta^t) , \tag{3}$$

where α_i is the weight for group i. A more thorough explanation of this formula and the CGD algorithm is provided in the original paper. The algorithms above were evaluated on various model/dataset combinations as shown in Table 9 of Appendix A.

[1] https://github.com/vihari/CGD

3.2 Datasets

The paper uses two groups of datasets: one composed of 2 synthetic toy datasets to compare the qualitative performance of CGD against Group-DRO, and a second one consisting of 2 synthetic datasets and six real-world datasets.

Qualitative Evaluation — The qualitative performance of CGD was evaluated on two toy datasets: a 2-feature dataset sampled from a standard normal distribution, and MNIST. On these datasets we applied three multi-group setups:

- Label Noise Setup - where 20% of the first group labels were flipped for Simple and 50% for MNIST.
- Rotation Setup - where labels (or the images for MNIST) were rotated by 30 degrees per group, such that the distances from the first and third group to the second is the same.
- Spurious Setup - where each sample has a third feature (the digit color for MNIST), which has an 80% correlation with the label.

Quantitative Evaluation — For the quantitative evaluation of the algorithm, we used eight datasets categorized into synthetic or real-world and based on whether they include spurious correlations (non-WILDS) or sub-population shift (WILDS). The datasets are summarized in Table 1.

Dataset (non-WILDS)	Type	Classes	Spurious Variable
CMNIST [6]	S	Digits	Digit Color
WaterBirds [1]	S	Water/landbird	Background
CelebA [7]	R	Blond/Non-Blond	Gender
MultiNLI [4]	R	NLI[a]	Negation Words
Dataset (WILDS)	**Type**	**Classes**	**Groups**
Camelyon17 [8]	R	Tumor/Non-Tumor	Source Hospital
PovertyMap [9]	R	Wealth-Index	Country & Rural/Urban
FMoW [10]	R	Building/Land use	Year & Region
CivilComments [11]	R	Toxic/Non-Toxic	Mentioned Demographics

[a] Natural Language Inference

Table 1. Summary of the datasets used for the quantitative evaluation. Datasets are categorized into synthetic (S) or real (R) and based on whether they include spurious correlations (non-WILDS) or sub-population shift (WILDS). The rightmost column shows the spurious variable for the non-WILDS dataset and the groups in which samples are partitioned for the WILDS datasets. MultiNLI and CivilComments are text datasets, while the others are image datasets.

3.3 Hyperparameters

Qualitative Evaluation — the models for the qualitative evaluation are trained using SGD for 400 epochs, with a learning rate of 0.1 as indicated in the paper. The batch size and the weight decay are not specified, and we used respectively 128 and 0.01, the latter of which was selected from the set $\{1, 0.1, 0.01, 0.001, 0\}$ by manually inspecting the CGD training weights (α) plots and choosing the value which produces a plot resembling Figure 1 of the original paper.

Dataset	lr	Weight Decay	Epochs	Batch Size	C	Step Size
CMNIST	1e-3	1e-1	10	32	0	0.05
WaterBirds	1e-3	1e-1	300	128	2	0.05
CelebA	1e-3	1e-4	10	8	2	0.05
MultiNLI	1e-3	1e-4	3	8	2	0.05
CivilComments	1e-5	1e-2	5	16	0	0.05
PovertyMap	1e-3	0	10	64	0	0.05
FMoW	1e-4	0	5	32	0	0.2
Camelyon17	1e-3	1e-2	5	32	0	0.05

Table 2. Hyperparameters used for each dataset and all algorithms in our experiments. C and step size are only relevant for CGD. If a hyperparameter is not included then the default value in WILDS should be assumed.

Quantitative Evaluation — following Piratla, Netrapalli, and Sarawagi[3] the models for the qualitative evaluation are trained using SGD with a momentum of 0.9, except for PovertyMap, which uses Adam. For the WILDS datasets, we followed the setup in Koh et al.[2], with some exceptions for the number of epochs and the batch size due to computational limitations. For CGD, the group-adjustment parameter C is 0 following Piratla, Netrapalli, and Sarawagi[3] and the step size η is selected from the WILDS leaderboard[2]. As for the non-WILDS datasets, the authors ran a grid search over the learning rate, weight decay, batch size, and C (set to 0 for CMNIST), but only provided the hyperparameter ranges and not the selected values. Due to computational constraints, we used the values provided in the `Group-DRO` paper [1], but if the value was out of the range defined by the authors, we selected the closest one in that range. For CGD, C was set to 2 according to Sagawa et al.[1], and η according to the WILDS leaderboard. The hyperparameter values are summarized in Table 2. The sensitivity of CGD with regards to hyperparameters C and η is explored in Appendix C.

3.4 Experimental setup and code

To make our study reproducible, we provide a guide on setting up a conda environment with all the required dependencies in our GitHub repository. The instructions are for a Google Compute Engine (GCE) virtual machine with preinstalled NVIDIA drivers. However, they are highly flexible and should work on any machine with minor tweaks (namely the CUDA version). We also provide a modified version of the WILDS repository which can run the CGD experiments out of the box. The complete instructions on reproducing our experiments are available in the README.md file of the repository.

3.5 Computational requirements

The experiments were executed on three machines, whose hardware configurations are listed in Table 3. The estimated runtime of each model/dataset pair on a Google Compute Engine Virtual Machine (GCE) is listed in Table 4, and each pair was run with three different seeds. Therefore the total computation time was approximately 440 hours.

4 Results

4.1 Results reproducing original paper

Qualitative Evaluation — We replicated the qualitative comparison in Section 4 of the paper for both the Simple and MNIST datasets. For the Simple dataset, CGD outperforms

[2] https://wilds.stanford.edu/leaderboard/

Machine	CPU	RAM (GB)	Graphics Card	VRAM (GB)
GCE VM	Intel Xeon 8173M	16	Tesla T4	16
Laptop A	AMD Ryzen 7 5800h	16	RTX 3050Ti Mobile	4
Laptop B	Intel i5-9300H	16	GTX 1660Ti Mobile	6

Table 3. Hardware configuration of the machines used to run experiments.

Algorithm	CMNIST	WaterBirds	CelebA	MultiNLI
ERM, Group-DRO	1h	7h	8h	4h
CGD	2h	15h	31h	4.5h
Algorithm	**CivilComments**	**PovertyMap**	**FMoW**	**Camelyon17**
ERM, Group-DRO	16h	0.3h	2h	3h
CGD	17h	8h	20h	10h

Table 4. Estimated runtime in hours of each model and dataset combination on a Google Compute Engine Virtual Machine with a Tesla T4 GPU. The runtimes of ERM and Group-DRO are similar for all datasets.

Group-DRO, albeit with a smaller gap than the results of the paper, as can be seen in Table 5. An inspection of the group weights α over epochs as shown in Figure 1 reveals the CGD effectively behaves as described in the paper:

- Label Noise setup - CGD avoids training only on the noisy majority.
- Rotation Setup - CGD, on average, focuses on the center group, which has the optimal classifier. However, this selection varies from seed to seed (Appendix B).
- Spurious Setup - CGD correctly identifies the clean majority and assigns it a much stronger weight than in the paper.

For the MNIST dataset, we were not able to replicate the paper results. As shown in Table 6, we achieved similar average and worst-group accuracies between CGD and Group-DRO, except for the Rotation setup, where Group-DRO failed to achieve reasonable accuracy. We suspect there might be an issue with the experimental setup for Group-DRO because the original paper achieved significantly better results.

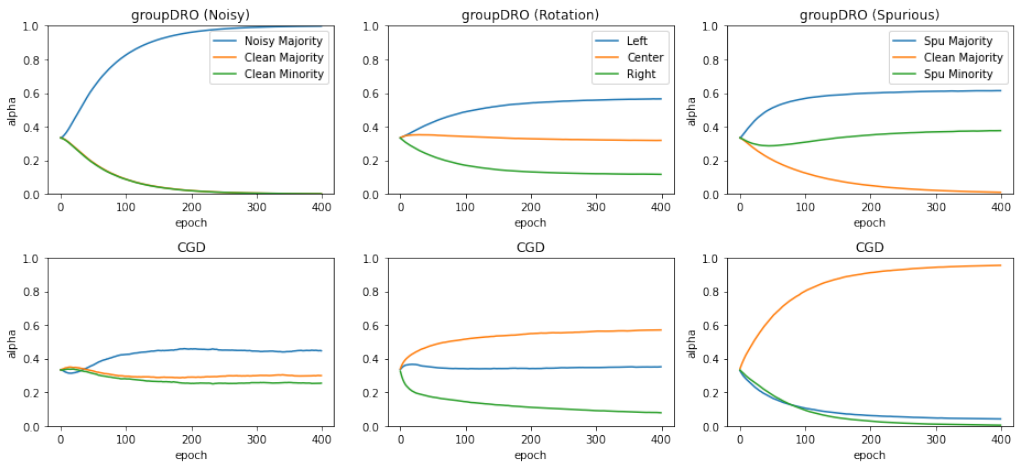

Figure 1. The comparison of group weights α for Group-DRO (top) and CGD (bottom) for the Simple dataset setups: label noise, rotation and spurious.

Algorithm	Noisy Simple	Rotation Simple	Spurious Simple
G-DRO	0.26 (0.02)	0.47 (0.04)	0.42 (0.03)
CGD	**0.22 (0.01)**	**0.46 (0.06)**	**0.32 (0.01)**

Table 5. Worst group losses on the test split of the simple dataset, averaged over six seeds. The standard deviation is shown in parentheses.

Algorithm	Metric	Noisy MNIST	Rotation MNIST	Spurious MNIST
G-DRO	Avg. Acc.	**77.36 (10.02)**	30.58 (3.82)	**92.47 (2.63)**
	W.g. Acc.	**77.25 (9.95)**	28.02 (4.42)	**91.75 (2.68)**
CGD	Avg. Acc.	76.35 (7.8)	**92.51 (1.78)**	92.51 (1.78)
	W.g. Acc.	76.24 (7.81)	**91.7 (2.35)**	91.7 (2.35)

Table 6. Average and worst group accuracies on the test split of MNIST, averaged over three seeds. The standard deviation is shown in parentheses.

Quantitative Evaluation — We reproduced the experiments on the non-WILDS and WILDS datasets and compared the performance of CGD against ERM and Group-DRO. Table 7 summarizes the results on the four non-WILDS datasets. CGD outperforms the other algorithms on the synthetic datasets with spurious correlations (CMNIST and WaterBirds), but fails to improve in the real-world datasets with spurious correlations (CelebA and MultiNLI) over ERM and Group-DRO. As for the WILDS datasets whose results are shown in Table 8, CGD is the best algorithm only on Camelyon17 (albeit with a larger standard deviation than ERM) and on the in-domain evaluation of PovertyMap, while ERM has a significant advantage on the out-of-domain evaluation against CGD, showing a larger gap than what claimed by the paper. Overall, the results are in line with the paper, which shows that CGD is better in some setups and achieves comparable performances in others, but its superiority is not clear.

	CMNIST		WaterBirds	
Algorithm	Avg. Acc.	W.g. Acc.	Avg. Acc.	W.g. Acc.
ERM	55.3 (2.23)	10.5 (4.47)	97.1 (0.03)	52.2 (1.18)
G-DRO	**97.6** (0.49)	**96.8** (0.69)	**97.3** (0.06)	71.7 (0.55)
CGD	**98.0** (0.34)	**97.0** (0.4)	**97.3** (0.13)	**73.2** (0.39)

	CelebA		MultiNLI	
Algorithm	Avg. Acc.	W.g. Acc.	Avg. Acc.	W.g. Acc.
ERM	**96.0** (0.12)	36.3 (6.04)	**62.2** (11.27)	16.1 (18.59)
G-DRO	94.9 (0.11)	59.1 (1.72)	49.9 (0.76)	**27.5** (2.62)
CGD	95.0 (0.13)	**59.8** (8.72)	50.2 (1.01)	27.1 (1.36)

Table 7. Average and worst-group accuracies on the test splits of the non-WILDS datasets. In parentheses are the standard deviations.

	Camelyon17 Avg. Acc.	PovertyMap W.r. Pearson R		FMoW W.r. Acc.	CivilComments W.g. Acc.
Algorithm	OOD	ID	OOD	OOD	ID
ERM	70.3 (6.4)	0.57 (0.07)	**0.45** (0.06)	**32.3** (1.2)	56.0 (3.6)
G-DRO	68.4 (7.3)	0.54 (0.11)	0.39 (0.06)	30.8 (0.8)	**70.0** (2.0)
CGD	**70.4** (7.56)	**0.63** (0.03)	0.38 (0.07)	29.8 (1.46)	69.7 (1.09)

Table 8. Results for different metrics on the test splits of the WILDS datasets. In parentheses are the standard deviations. w.r. and w.g stand for worst region and worst group accuracy. The values were taken from the original paper with the exception of CGD which we trained ourselves.

4.2 Results beyond original paper

The paper does not discuss the runtime of the algorithms, which we documented in Table 4. We find that CGD is often 2 to 26 times slower than the other algorithms depending on the dataset. As seen in Table 4, the runtime increase varies across datasets: for WaterBirds, we have an increase of 50%, while for PovertyMap, the increase is over 2000%. The computation of the gradients for each group at each training step (Equation 3) might be one possible reason. This hypothesis is supported by the fact that datasets with many groups, such as PovertyMap, FMoW, and Camelyon, with 13, 16, and 5 groups respectively, had the largest increase in training time (the other datasets do not have more than four groups). CelebA is an exception, but the runtime increase may be due to the small batch size (8). In view of the above, we concluded that the small gains in accuracy may not justify the increased training time. Moreover, the authors mathematically proved that CGD is a sound optimization algorithm as it decreases the macro/group-average loss monotonically. We test this empirically by plotting the loss curves for the non-WILDS datasets (except for MultiNLI, since it only has 3 epochs) in Figure 2 for `Group-DRO` and CGD. We observe that the validation loss curve is not monotonic. Instead, it fluctuates and seems to increase for all datasets. This is particularly evident in WaterBirds. One reason for this behavior may be the use of batches to to approximate the gradients, whereas the proof assumes that the whole dataset is used at each training step. Due to these limitations, we cannot confirm or disprove the claim, so we compare the relative monotonicity and stability between CGD and `Group-DRO`. For CMNIST, both show similar degrees of monotonicity. On WaterBirds, CGD has a more stable training than `Group-DRO`, whose validation loss has large fluctuations between epochs. This may be a side effect of focusing on the group with the largest training loss as identified by [3]. With regards to CelebA, the validation loss of CGD increases whereas the loss of `Group-DRO` appears to be decreasing. In conclusion, we cannot clearly show that the loss of CGD decreases monotonically, but our findings suggest that it is more stable than `Group-DRO`. Future research may further investigate this claim by running experiments that come closer to the assumptions of the authors, namely a bigger batch size and more epochs.

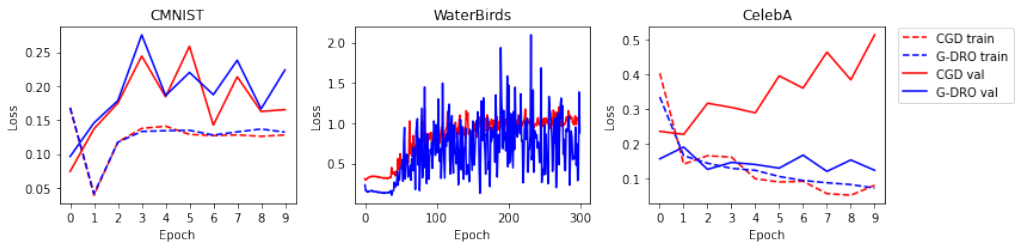

Figure 2. Loss curves for CGD and `Group-DRO` on three non-WILDS datasets.

5 Discussion

Overall, the majority of the claims in the paper were reproducible. CGD indeed performed comparably or better than ERM and Group-DRO depending on the dataset. However, the increased runtime of CGD might outweigh the minor accuracy gain. The claim of the authors about the monotonicity of CGD could not be reproduced empirically in a reliable way, and further research is needed. Lastly, CGD appears to have a more stable training in comparison to Group-DRO.

5.1 What was easy

The methods used in the paper and the results were described clearly and intuitively. Moreover, the code for CGD was published by the authors alongside clear instructions on integrating it into the WILDS framework. Finally, the framework chosen by the authors is modular, and additional datasets and algorithms could be easily integrated.

5.2 What was difficult

- **Resources**: Model training required a massive amount of GPU time due to the dataset size and the sheer number of experiments.
- **Code**: The C parameter for the WILDS implementation of Group-DRO could not be located in the code, so we could not select a value for it. We suspect that there might be an inconsistency between the theory and the code.Even though the code for CGD and WILDS was available, we could not run experiments out of the box: the CDG code had to be updated to work with the latest version of WILDS and required some modifications. Moreover, the dataset code for MultiNLI was missing, so we implemented it following [1] and the advice of the authors.
- **Hyperparameters**: Collecting the correct hyperparameter values was challenging because the paper only provided a range, and there were multiple conflicting sources: the paper, the repository, and the WILDS leaderboard (the latter suggested by the authors in our correspondence). Moreover, some values did not lead to the expected accuracy according to the paper, so we had to experiment with additional values, e.g., the weight decay for CMNIST and Waterbirds. Finally, The best values for the CGD step size η in the WILDS leaderboard (0.05 and 0.2) were not in the range described in the paper.

5.3 Communication with original authors

We reached out to the original authors to obtain more information about the chosen hyperparameter values. They promptly replied, specifying that for the WILDS datasets, the hyperparameters are as configured by default in WILDS 1.2.2. For their algorithm, CGD, they informed us that its hyperparameters could be found in the WILDS leaderboard. In addition, they gave us helpful information about some parts of their code that were missing, such as the MultiNLI dataset.

References

1. S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang. "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization." In: **Proceedings of the International Conference on Learning Representations** (2020).
2. P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, et al. "Wilds: A benchmark of in-the-wild distribution shifts." In: **Proceedings of the International Conference on Machine Learning** (2021).
3. V. Piratla, P. Netrapalli, and S. Sarawagi. "Focus on the Common Good: Group Distributional Robustness Follows." In: **Proceedings of the International Conference on Learning Representations** (2022).
4. A. Williams, N. Nangia, and S. R. Bowman. "A broad-coverage challenge corpus for sentence understanding through inference." In: **Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies** (2018).
5. S. Sagawa, P. W. Koh, T. Lee, I. Gao, S. M. Xie, K. Shen, A. Kumar, W. Hu, M. Yasunaga, H. Marklund, et al. "Extending the wilds benchmark for unsupervised adaptation." In: **arXiv preprint arXiv:2112.05090** (2021).
6. I. Gulrajani and D. Lopez-Paz. "In search of lost domain generalization." In: **Proceedings of the International Conference on Learning Representations** (2021).
7. Z. Liu, P. Luo, X. Wang, and X. Tang. "Deep learning face attributes in the wild." In: **Proceedings of the IEEE international conference on computer vision** (2015).
8. P. Bandi, O. Geessink, Q. Manson, M. Van Dijk, M. Balkenhol, M. Hermsen, B. E. Bejnordi, B. Lee, K. Paeng, A. Zhong, et al. "From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge." In: **IEEE Transactions on Medical Imaging** (2019).
9. C. Yeh, A. Perez, A. Driscoll, G. Azzari, Z. Tang, D. Lobell, S. Ermon, and M. Burke. "Using publicly available satellite imagery and deep learning to understand economic well-being in Africa." In: **Nature communications** 11.1 (2020), p. 2583.
10. G. Christie, N. Fendley, J. Wilson, and R. Mukherjee. "Functional map of the world." In: (2018).
11. D. Borkan, L. Dixon, J. Sorensen, N. Thain, and L. Vasserman. "Nuanced metrics for measuring unintended bias with real data for text classification." In: **Companion proceedings of the 2019 world wide web conference**. 2019, pp. 491–500.

A Model Specifications

The optimization algorithms evaluated in this study were applied on several models/-dataset combinations, as shown in Table 9. A different model was used for each dataset, depending on the task.

Dataset	Model	Pretrained	Parameters
Simple	Linear Binary Classifier	False	6
MNIST	ResNet18	False	11M
CMNIST	ResNet18	True	11M
WaterBirds	ResNet50	True	25M
CelebA	ResNet50	True	25M
MultiNLI	DistilBERT-uncased	True	66M
CivilComments	DistilBERT-uncased	True	66M
PovertyMap	$\text{Resnet18}_{\text{MS}}$	True	11M
FMoW	DenseNet121	True	76M
Camelyon17	DenseNet121	True	76M

Table 9. The datasets, alongside the model selection, if they use or not pretrained weights and the number parameters. $\text{Resnet18}_{\text{MS}}$ refers to Resnet18 Multi-Spectral.

B Training Group Selection

Even though CGD is generally consistent with the group choice, in the Simple-Rotation setup, the group the algorithm focuses on varies, as shown in Figure 3 plots. While for seed 3 CGD correctly identifies the center group for seeds 13 and 42, it focuses on the left group. In comparison, Group-DRO shows a more consistent group choice.

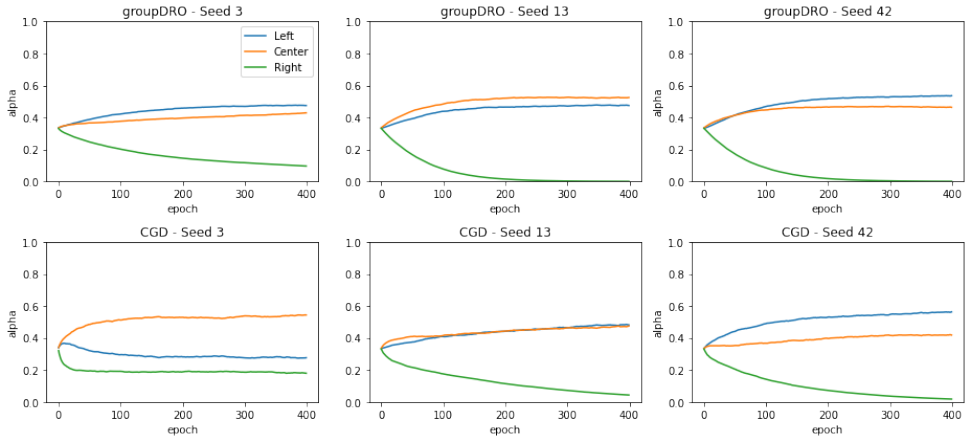

Figure 3. `Group-DRO` and CGD group weights α for seeds 3, 13, and 42 over epochs for the Simple dataset and the Rotation setup. While `Group-DRO` shows a consistent behavior CGD either focuses on the center group as expected (seed 3) or on a mix of the center and left groups

C Hyperparameter Sensitivity

We evaluated the hyperparameter sensitivity for CGD with respect to the group adjustment parameter C and the step size η, using the values in Table 2 for the other hyperparameters. While a more throughout evaluation on real-world datasets is recommended, the evaluation was conducted using CMNIST, which allowed us to test multiple values and average the results over three seeds with limited compute.

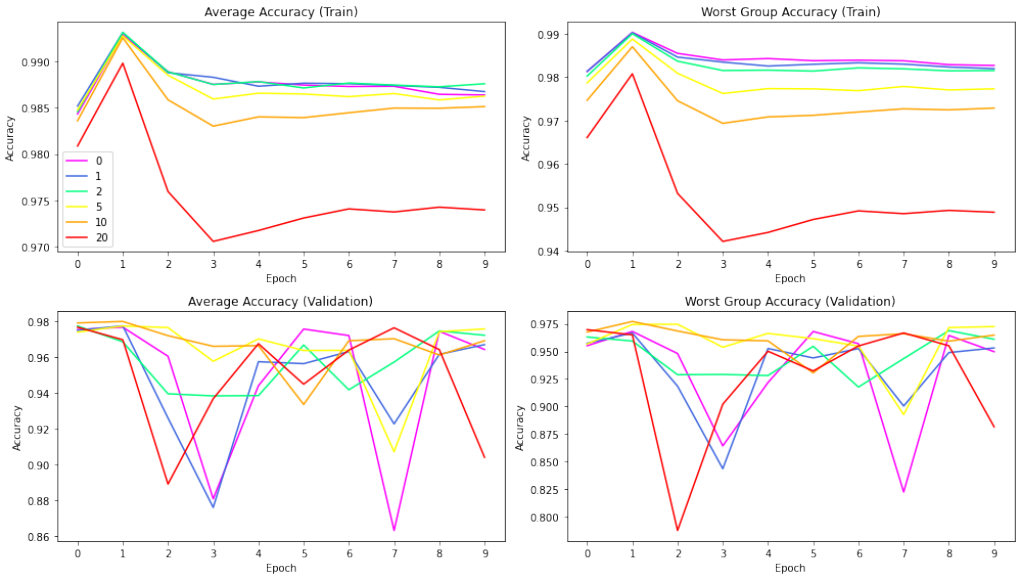

Figure 4. The average and worst group accuracies obtained by CGD on CMNIST with respect to the group adjustment parameter C. The results are averaged over three seeds.

To evaluate the effect of C we fixed the step size η to 0.05 and trained the model using $C \in \{0, 1, 2, 5, 10, 20\}$. By observing the plots in Figure 4 we can notice that large increases of C lead to a degradation of training performance for the average and worst group accuracy but, interestingly, this effect is not replicated in the validation set, which instead reveals that both small and large values of C cause instabilities in the validation metrics. This is confirmed by the test set results in Table 10, for which $C \in \{5, 10\}$ perform best with regards to the average and worst group accuracies.

Step Size η	Avg. Acc.	W.g. Acc.
0.001	0.973	0.968
0.01	0.978	0.973
0.05	0.976	0.963
0.1	0.980	0.972
1	0.978	0.971

C	Avg. Acc.	W.g. Acc.
0	0.974	0.964
1	0.974	0.962
2	0.976	0.963
5	0.975	0.972
10	0.976	0.972
20	0.974	0.966

Table 10. The average and worst group accuracies for the test set of CMNIST obtained by CGD with respect to the group adjustment parameter C and the step size η

As for the step size hyperparameter η we fixed $C = 2$ and evaluted the performance of CGD over the set $\{1, 0.1, 0.05, 0.01, 0.001\}$. With regards to the training performance the different values performed similarly, but as can be seen in figure 5 the validation set accuracy shows that smaller values result in a more unstable training, and, as confirmed by the test set metrics in Table 10, $\eta \in \{0.1, 1\}$ work best for this dataset.

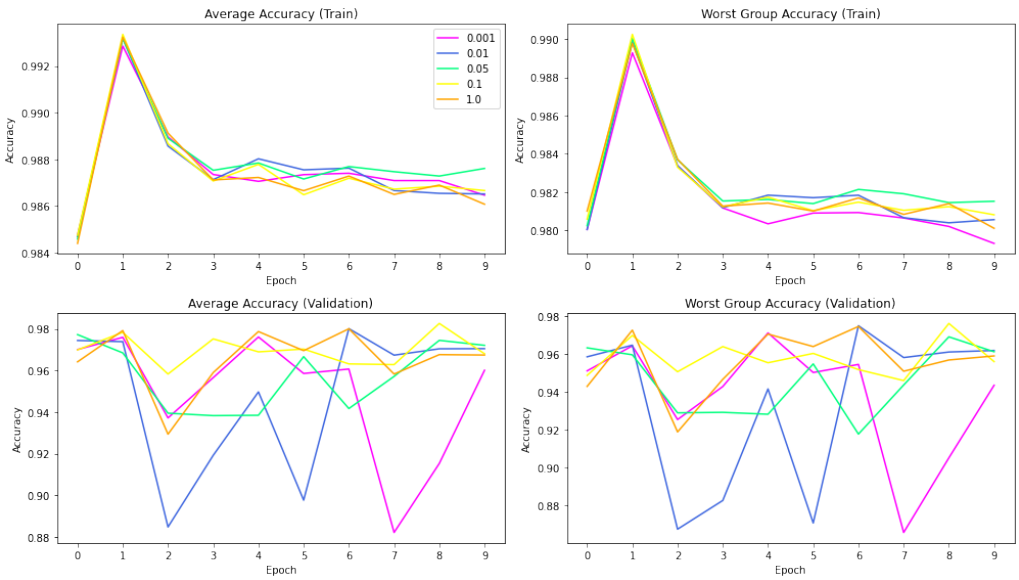

Figure 5. The average and worst group accuracies obtained by CGD on CMNIST with respect to the step size parameter η. The results are averaged over three seeds.