Empirical Study on Optimizer Selection for Out-of-Distribution Generalization

Hiroki Naganuma[*] [1,2] **, Kartik Ahuja** [1,2]**, Ioannis Mitliagkas** [1,2,3]
[1] Mila - Quebec AI Institute, [2] Université de Montréal, [3] Canada CIFAR AI Chair
Montréal, Québec, Canada

Shiro Takagi[†]**, Tetsuya Motokawa** [4]**, Rio Yokota** [5]**, Kohta Ishikawa** [6]**, Ikuro Sato** [5,6]
[4]University of Tsukuba, [5]Tokyo Institute of Technology, [6]Denso IT Laboratory Inc.
Tokyo, Japan

Abstract

Modern deep learning systems are fragile and do not generalize well under distribution shifts. While much promising work has been accomplished to address these concerns, a systematic study of the role of optimizers and their out-of-distribution generalization performance has not been undertaken. In this study, we examine the performance of popular first-order optimizers for different classes of distributional shift under empirical risk minimization and invariant risk minimization. We address the problem settings for image and text classification using DomainBed, WILDS, and Backgrounds Challenge as out-of-distribution datasets for the exhaustive study. We search over a wide range of hyperparameters and examine the classification accuracy (in-distribution and out-of-distribution) for over 20,000 models. We arrive at the following findings: i) contrary to conventional wisdom, adaptive optimizers (e.g., Adam) perform worse than non-adaptive optimizers (e.g., SGD, momentum-based SGD), ii) in-distribution performance and out-of-distribution performance exhibit three types of behavior depending on the dataset – linear returns, increasing returns, and diminishing returns. We believe these findings can help practitioners choose the right optimizer and know what behavior to expect. The code is available at https://github.com/Hiroki11x/Optimizer_Comparison_OOD.

1 Introduction

Selecting an appropriate optimizer is crucial for the successful training of deep neural networks. For example, the optimizer influences training speed, stability, and generalization performance. Several studies have compared a variety of optimizers and investigated their influence on trainability and generalizability [Wil+17; SBH19; Cho+19]. Some studies concluded that non-adaptive optimizers provide better generalization [Wil+17; BH18], while others countered that optimizer selection does not affect generalization performance [SBH19; SSH20].

Previous conflicting reports have relied on the lack of hyperparameter search, and exhaustive experiments have shown that adaptive optimization methods can sometimes outperform non-adaptive methods in terms of generalization performance when tuned properly [Cho+19].

Although these studies have made substantial progress to improve our understanding of optimizer characteristics, they are based on a common assumption of supervised learning in which the training and test data are drawn from the same distribution. In real-world applications, however, it is often the case that the test data obey a distribution different from the training data, and this distributional

[*] naganuma.hiroki@mila.quebec
[†] Independent Researcher

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

shift violates the typical independent and identically distributed (i.i.d.) assumption for learning [NAN21]. Comparing the generalization performance under this distributional shift, known as out-of-distribution (OOD) generalization [She+21], among different optimizers is of great interest in theory and in practice.

Because our objective is to investigate the impact of commonly used optimizers in the OOD problem setting, we target five of the most popular and standard optimizers that have been used and studied in recent years [SSH21]. In this work, we evaluate the OOD generalization performance on 10 typical OOD benchmarks using different optimizers, including stochastic gradient descent (SGD), Momentum SGD [Pol64], and Nesterov accelerated gradient (NAG, also known as Nesterov's momentum) [Nes03] in the family of non-adaptive methods as well as RMSProp [TH12] and Adam [KB15] in the family of adaptive optimizers. We use DomainBed (which includes seven image datasets) [GL21], Backgrounds Challenge dataset [Xia+21], Amazon-WILDS, and CivilComments-WILDS [Koh+21] for the benchmarks. We train deep neural networks using either Empirical Risk Minimization (ERM) or Invariant Risk Minimization (IRM) [Arj+19] with various optimizers. We conduct an exhaustive hyperparameter search to obtain hyperparameters that give good in-domain validation accuracy for each optimizer. We then test the models on the above OOD test sets.

An advantage of our approach is that we consider a wide range of hyperparameter configurations and datasets, including image classification and NLP problem settings. Given that the parameter search space was different from that of previous studies [Wil+17; SBH19; Cho+19], exploring hyperparameters as comprehensively as possible is crucial for a reliable investigation. Therefore, we follow Choi et al. [Cho+19], who most exhaustively explored hyperparameters for parameter searches. We further checked the soundness of our experiments. Using the same optimizer as the existing benchmark, we explored more hyperparameter configurations and recorded results that exceeded the best-reported performance.

In summary, our contributions are as follows:

- We design and perform a comparison of the effect of optimizers on OOD generalization on a number of OOD benchmarks. To the best of our knowledge, we are the first to consider such a wide variety and scale of datasets. Also, We conduct an empirical study using a wide range of hyperparameter configurations, examining over 20,000 models, evaluating their performance, and measuring the performance gap when moving from the in-distribution test set to the OOD test set.
- We demonstrate that, for the exhaustive image classification and NLP tasks, differences in the optimizer lead to differences in the OOD generalization performance. The adaptive optimizers provide more in-distribution overfitting and degrade OOD performance more than the non-adaptive optimizers within our exhausive experiments.
- We show that the observed correlation behaviors between in-distribution performance and OOD performance can be categorized into typical patterns: linear return, diminishing return, and increasing return [3]. This result will help practitioners better understand and select optimizers.

Our observation is that the adaptive and non-adaptive optimizers have no significant performance difference in terms of in-distribution generalization performance. In contrast, the adaptive optimizers are inferior in OOD generalization performance on 8 out of 10 datasets. In other words, in all appropriate problem settings of image and text classification, we observed that non-adaptive optimizers show superior OOD performance.

We also observe a similar trend not only for training with ERM but also with IRM. This result is consistent with theoretical results [Zou+22] showing the drawback of adaptive optimizers such as Adam under the i.i.d. assumption, where instead of fast convergence, they tend to fit the noise in the data.

[3] These show how much performance can be expected in the out-of-distribution if we increase the in-distribution performance. These terms are explained in detail in Section 3.3.

Table 1: DomainBed, Backgrounds Challenge, Amazon-WILDS, and CivilComments-WILDS: Comparison of the best OOD accuracy of ERM between five optimizers. Except for a small set of problems, momentum SGD outperforms Adam in all but eight cases. As a soundness check, we confirm that our Adam results outperform all existing benchmark results using Adam. Details are given in Appendix J.2.

Model	OOD Dataset	Non-Adaptive Optimizer			Adaptive Optimizer	
		SGD	Momentum	Netsterov	RMSProp	Adam
4-Layer CNN	ColoredMNIST	34.01%	34.23%	40.56%	**89.30%**	73.92%
	RotatedMNIST	90.00%	95.41%	94.06%	96.27%	**96.40%**
ResNet50	VLCS	99.43%	**99.43%**	99.29%	99.29%	99.29%
	PACS	88.67%	**89.55%**	89.25%	88.81%	89.36%
	OfficeHome	64.64%	**65.01%**	63.82%	62.91%	63.82%
	TerraIncognita	**63.21%**	62.41%	62.85%	62.31%	61.35%
	DomainNet	58.38%	61.91%	**62.24%**	55.74%	58.48%
	BackgroundChallenge	-	**80.09%**	-	-	77.90%
DistilBERT	WILDSAmazon	52.00%	**54.66%**	54.66%	53.33%	51.99%
	WILDSCivilComment	51.66%	57.69%	**60.07%**	45.39%	46.82%

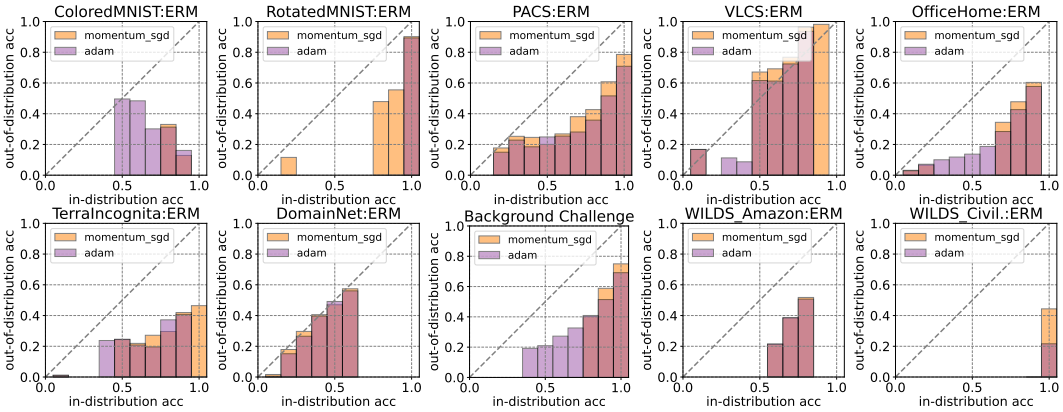


Figure 1. DomainBed, Backgrounds Challenge, Amazon-WILDS, and CivilComments-WILDS: Comparison of the in-distribution accuracy and the OOD accuracy of ERM between Momentum SGD and Adam. We also show the training results in the exhaustive hyperparameter search range as a scatter plot in Appendix H.2. In these ten plots, for each dataset, the in-distribution performance is separated by every ten bins 0.1. The average OOD performance when evaluating the checkpoints in that bin is shown on the vertical axis. We compare which optimizer shows better OOD performance for each model that achieves equivalent in-distribution performance. In most cases, momentum SGD outperforms Adam in OOD performance in the rightmost region of our interest (the region where high in-distribution performance is achieved).

2 Preliminaries

2.1 Out-of-Distribution Generalization Datasets

We use the following datasets to evaluate the optimizer influence on OOD generalization: DomainBed [GL21], Backgrounds Challenge [Xia+21], Amazon-WILDS [Koh+21], and CivilComments-WILDS [Koh+21]. These datasets include both artificially created and real-world data, and the applications include image classification and text classification. We describe the details of these datasets in Appendix C.2.

2.2 Model Selection Method and Evaluation Metrics

We follow the method of the benchmark paper for each dataset in the model selection method [GL21; Xia+21; Koh+21]. The OOD accuracy also follows existing studies for each dataset. A detailed description is provided in Appendix E.

3 Experiments

3.1 Experimental Overview

Our study aims to elucidate the influence of optimizer selection under distributional shifts. To that end, we perform image and text classification and evaluate the trained model accuracy on the benchmark datasets introduced in Section 2.1. By comparing the test accuracy for in-distribution samples with

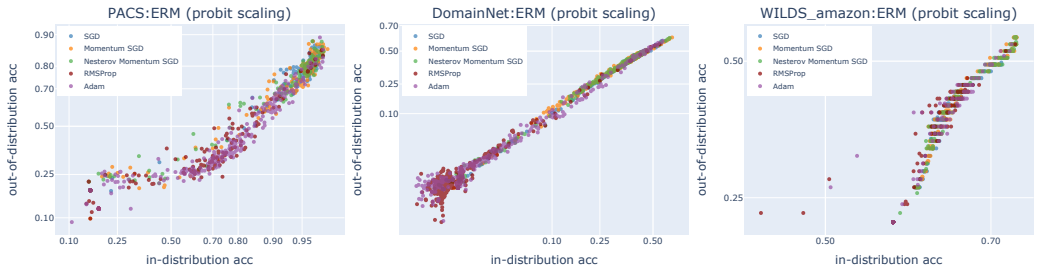


Figure 2. Three-types of correlation behavior: increasing return (PACS), linear return (DomainNet), and diminishing return (Amazon-WILDS). The legend circles on the right side of each figure show, in order, the SGD, Momentum SGD, Nesterov Momentum, RMProp, and Adam. The results of other detailed dataset experiments are described in Appendixes 22, 23 and 24

that for OOD samples, we can observe how the solution found by each optimizer is robust to the distributional shift. We investigate both ERM and IRMv1 [Arj+19], a problem setting to solve IRM, to clarify the relationship between the problem formulation and optimizer selection in the OOD problem. For all datasets except for the Backgrounds Challenge dataset, we compare five optimizers. For the Backgrounds Challenge dataset, we compare only Momentum SGD and Adam due to the computational cost.

We describe the configurations of hyperparameters and protocol for the experiments in further detail in Appendix F and Appendix E respectively. The remaining experimental settings of environment are explained in Appendix D.

3.2 Experimental Results

Figure 1 shows the relationship between the in-distribution accuracy and the OOD accuracy in the ERM setting. The x-axis of the plot is the in-distribution accuracy and the y-axis is the OOD accuracy. To make the trend more clear, the in-distribution accuracy corresponding to the x-axis is divided into 10 bins, and the average performance of the OOD accuracy in each bin is shown on the y-axis. We compared Momentum SGD as the best non-adaptive optimizer, with Adam as the best adaptive optimizer. In our area of interest, where a high in-distribution performance is achieved, such as top 10% trial, Momentum SGD outperforms Adam on 8 of the 10 datasets. This indicates that non adaptive optimizer is more advantageous than adaptive optimizer in OOD, even though the performance is similar in the IID environment. A detailed explanation of the experimental results for each dataset is given in Appendix G.

3.3 Correlation Behaviors

Our results show that three typical types of behavior are observed in terms of the correlation between in-distribution performance and OOD performance for different datasets. Detailed results of the experiments on all datasets are shown in Appendix H.3. We follow Miller et al. [Mil+21] who used a probit transform to show relationship. The three types are increasing return, linear return, and diminishing return. These show how much performance in OOD can be expected if we increase the in-distribution performance.

We show in Appendix H.3 the results of our experiments without a probit transform. These results confirm that diminishing returns, in particular, are not necessarily linear even before probit transformation. This is consistent with the recent point by [Wen+22; Ten+22] that the accuracy of IID and OOD differ depending on the dataset. Furthermore, as recent work, [Bae+22] states, whether or not the linear return actually occurs depends on the problem set. Our results support their claims, and we have comprehensively addressed data sets and problem sets that Miller et al. [Mil+21] did not address and uncovered trends that they did not reveal.

4 Discussion and Conclusion

We conduct an exhaustive empirical comparison of the generalization performance of various optimizers under different practical distributional shifts. The investigation elucidates how optimizer selection affects OOD generalization.

As the main claim, the answer to our research objective is that the non-adaptive optimizer is superior to the adaptive optimizer in terms of OOD generalization.

This is supported by following evidence: i) when comparing in-distribution accuracy is the same, OOD accuracy of non-adaptive optimizers is greater than that of adaptive optimizers (Figure 1); ii) on average and top performance, the OOD accuracy of non-adaptive optimizer is higher (Figures 3, 4, and 5). All these points support our main claim that non-adaptive optimizers are superior in OOD generalization within our exhaustive experiments. We have tuned Adam to the range that it is used in practice and have updated Adam's scores on all OOD datasets which use Adam optimizer as default for benchmarks. This supports the soundness of our experiments.

Overall, we can conclude that optimizer selection influences OOD generalization in the cases we are interested in. Future research should consider the algorithm or loss function and optimizers in the OOD problem. The results of IRM show a trend similar to ERM's, but a more detailed analysis is needed to consider the differences in loss landscapes. All these points support our main claim that non-adaptive optimizers are superior in OOD generalization within our exhaustive experiments.

Finally, we would like to mention the limitations of our work. One limitation is that we did not study recently proposed and less popular optimizers. The choice of optimizers we study is in line with previous work [Cho+19], [SSH21] on optimizer comparison and with most recent OOD work; those studies overwhelmingly focused on Adam rather than e.g. AdamW [LH17]. Similarly other less popular optimizers such as SWA [Izm+18], SWAD [Cha+21], SAM [For+20], and second-order methods, have been omitted to allow for a more extensive study of the chosen methods.

Another limitation is that we employed a total of four models used in the DomainBed (ConvNet and ResNet50), Backgrounds Challenge (ResNet50), and WILDS (DistilBERT [San+19]) benchmarks, but we did not evaluated other deep neural network architectures such as the Vision Transformer [Dos+20].

Acknowledgments

We would like to thank Kilian Fatras, Divyat Mahajan and Mehrnaz Mofakhami for their helpful feedback. This work was supported by the Masason Foundation Fellowship awarded to Hiroki Naganuma. Computational resource of AI Bridging Cloud Infrastructure (ABCI) was awarded by "ABCI Grand Challenge" Program, National Institute of Advanced Industrial Science and Technology (AIST), and resource of the TSUBAME 3.0 was awarded by the TSUBAME Grand Challenge Program, Tokyo Institute of Technology.

References

[Pol64] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *Ussr computational mathematics and mathematical physics* 4.5 (1964), pp. 1–17.

[Ama98] Shunichi Amari. "Natural gradient works efficiently in learning". In: *Neural computation* 10.2 (1998), pp. 251–276.

[Nes03] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2003.

[FFP04] Li Fei-Fei, Rob Fergus, and Pietro Perona. "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories". In: *2004 conference on computer vision and pattern recognition workshop*. IEEE. 2004, pp. 178–178.

[Rus+08] Bryan C Russell et al. "LabelMe: a database and web-based tool for image annotation". In: *International journal of computer vision* 77.1-3 (2008), pp. 157–173.

[PY09] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[Ben+10] Shai Ben-David et al. "A theory of learning from different domains". In: *Machine learning* 79.1 (2010), pp. 151–175.

[Cho+10] Myung Jin Choi et al. "Exploiting hierarchical context on a large database of object categories". In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 129–136.

[Eve+10] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[SK12] Masashi Sugiyama and Motoaki Kawanabe. *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press, 2012.

[TH12] Tijmen Tieleman and Geoffrey Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[FXR13] Chen Fang, Ye Xu, and Daniel N Rockmore. “Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1657–1664.

[Sze+14] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *International Conference on Learning Representations*. 2014.

[Ghi+15] Muhammad Ghifary et al. “Domain generalization for object recognition with multi-task autoencoders”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2551–2559.

[KB15] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *International Conference on Learning Representations*. 2015.

[RBV16] Veselin Raychev, Pavol Bielik, and Martin Vechev. “Probabilistic model for code with decision trees”. In: *ACM SIGPLAN Notices* (2016).

[Li+17] Da Li et al. “Deeper, broader and artier domain generalization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5542–5550.

[LH17] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).

[Ven+17] Hemanth Venkateswara et al. “Deep hashing network for unsupervised domain adaptation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5018–5027.

[Wil+17] Ashia C Wilson et al. “The Marginal Value of Adaptive Gradient Methods in Machine Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. 2017.

[BH18] Lukas Balles and Philipp Hennig. “Dissecting adam: The sign, magnitude and variance of stochastic gradients”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 404–413.

[Ban+18] Peter Bandi 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* (2018).

[BVP18] Sara Beery, Grant Van Horn, and Pietro Perona. “Recognition in terra incognita”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 456–473.

[Cal+18] Sebastian Caldas et al. “Leaf: A benchmark for federated settings”. In: *arXiv preprint arXiv:1812.01097* (2018).

[Chr+18] Gordon Christie et al. “Functional Map of the World”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[Izm+18] Pavel Izmailov et al. “Averaging weights leads to wider optima and better generalization”. In: *arXiv preprint arXiv:1803.05407* (2018).

[RKK18] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. “On the Convergence of Adam and Beyond”. In: *International Conference on Learning Representations*. 2018.

[Arj+19] Martin Arjovsky et al. “Invariant risk minimization”. In: *arXiv preprint arXiv:1907.02893* (2019).

[Bor+19] Daniel Borkan et al. “Nuanced metrics for measuring unintended bias with real data for text classification”. In: *Companion Proceedings of The 2019 World Wide Web Conference*. 2019.

[Cho+19] Dami Choi et al. “On empirical comparisons of optimizers for deep learning”. In: *arXiv preprint arXiv:1910.05446* (2019).

[Kho19] Marc Khoury. “Adaptive versus Standard Descent Methods and Robustness Against Adversarial Examples”. In: *arXiv preprint arXiv:1911.03784* (2019).

[Liu+19] Liyuan Liu et al. "On the Variance of the Adaptive Learning Rate and Beyond". In: *arXiv e-prints* (2019), arXiv–1908.

[NLM19] Jianmo Ni, Jiacheng Li, and Julian McAuley. "Justifying recommendations using distantly-labeled reviews and fine-grained aspects". 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)*. 2019.

[Pen+19] Xingchao Peng et al. "Moment matching for multi-source domain adaptation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1406–1415.

[San+19] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).

[SBH19] Frank Schneider, Lukas Balles, and Philipp Hennig. "DeepOBS: A Deep Learning Optimizer Benchmark Suite". In: *International Conference on Learning Representations*. 2019.

[Tay+19] J. Taylor et al. "RxRx1: An Image Set for Cellular Morphological Variation Across Many Experimental Batches." In: *International Conference on Learning Representations (ICLR). AI for Social Good Workshop*. 2019.

[Wan+19] Yixiang Wang et al. "Assessing optimizer impact on dnn model sensitivity to adversarial examples". In: *IEEE Access* 7 (2019), pp. 152766–152776.

[Zha+19] Guodong Zhang et al. "Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model". In: *arXiv preprint arXiv:1907.04164* (2019).

[ACS20] Vahdat Abdelzad, Krzysztof Czarnecki, and Rick Salay. "The Effect of Optimization Methods on the Robustness of Out-of-Distribution Detection Approaches". In: *arXiv preprint arXiv:2006.14584* (2020).

[BCG20] Sara Beery, Elijah Cole, and Arvi Gjoka. "The iWildCam 2020 Competition Dataset". In: *arXiv preprint arXiv:2004.10340* (2020).

[Dav+20] Etienne David et al. "Global Wheat Head Detection (GWHD) dataset: a large and diverse dataset of high-resolution RGB-labelled images to develop and benchmark wheat head detection methods". In: *Plant Phenomics* 2020 (2020).

[Dos+20] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[For+20] Pierre Foret et al. "Sharpness-aware minimization for efficiently improving generalization". In: *arXiv preprint arXiv:2010.01412* (2020).

[Gei+20] Robert Geirhos et al. "Shortcut learning in deep neural networks". In: *Nature Machine Intelligence* 2.11 (2020), pp. 665–673.

[Hu+20] Weihua Hu et al. "Open Graph Benchmark: Datasets for machine learning on graphs". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[Liu+20] Liyuan Liu et al. "On the Variance of the Adaptive Learning Rate and Beyond". In: *International Conference on Learning Representations*. 2020.

[Met+20] Luke Metz et al. "Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves". In: *arXiv preprint arXiv:2009.11243* (2020).

[SSH20] Robin M Schmidt, Frank Schneider, and Philipp Hennig. "Descending through a Crowded Valley–Benchmarking Deep Learning Optimizers". In: *arXiv preprint arXiv:2007.01547* (2020).

[Wad+20] Neha S Wadia et al. "Whitening and second order optimization both destroy information about the dataset, and can make generalization impossible". In: *arXiv preprint arXiv:2008.07545* (2020).

[Yeh+20] Christopher Yeh et al. "Using publicly available satellite imagery and deep learning to understand economic well-being in Africa". In: *Nature Communications* (2020).

[Cha+21] Junbum Cha et al. "Swad: Domain generalization by seeking flat minima". In: *Advances in Neural Information Processing Systems* 34 (2021).

[Dav+21] Etienne David et al. *Global Wheat Head Dataset 2021: an update to improve the benchmarking wheat head localization with more diversity*. 2021. arXiv: 2105.07660 [cs.CV].

[GL21] Ishaan Gulrajani and David Lopez-Paz. "In Search of Lost Domain Generalization". In: *International Conference on Learning Representations*. 2021.

[Koh+21] Pang Wei Koh et al. "Wilds: A benchmark of in-the-wild distribution shifts". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5637–5664.

[Lu+21] Shuai Lu et al. "CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation". In: *arXiv preprint arXiv:2102.04664* (2021).

[Mil+21] John P Miller et al. "Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 7721–7735.

[NAN21] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. "Understanding the failure modes of out-of-distribution generalization". In: *International Conference on Learning Representations*. 2021.

[SSH21] Robin M Schmidt, Frank Schneider, and Philipp Hennig. "Descending through a crowded valley-benchmarking deep learning optimizers". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9367–9376.

[She+21] Zheyan Shen et al. "Towards out-of-distribution generalization: A survey". In: *arXiv preprint arXiv:2108.13624* (2021).

[Xia+21] Kai Yuanqing Xiao et al. "Noise or Signal: The Role of Image Backgrounds in Object Recognition". In: *International Conference on Learning Representations*. 2021.

[Bae+22] Christina Baek et al. "Agreement-on-the-Line: Predicting the Performance of Neural Networks under Distribution Shift". In: *arXiv preprint arXiv:2206.13089* (2022).

[Ten+22] Damien Teney et al. *ID and OOD Performance Are Sometimes Inversely Correlated on Real-world Datasets*. 2022. DOI: 10.48550/ARXIV.2209.00613. URL: https://arxiv.org/abs/2209.00613.

[Wen+22] Florian Wenzel et al. "Assaying Out-Of-Distribution Generalization in Transfer Learning". In: *arXiv preprint arXiv:2207.09239* (2022).

[Zou+22] Difan Zou et al. *Understanding the Generalization of Adam in Learning Neural Networks with Proper Regularization*. 2022. URL: https://openreview.net/forum?id=G7PfyLimZBp.

Contents (Appendix)

A Appendix

B Related Works

B.1 Optimizer Selection

Understanding the characteristics of the many optimizers proposed for deep neural network training [SSH21] is of great importance to the machine learning research community. In terms of convergence, preconditioned optimizers, including Adam, are known to be superior to non-adaptive optimizers [KB15; Liu+19; Ama98].

While preconditioned optimization methods seem to be better than non-pre-conditioned ones in terms of convergence, Zhang et al. argued that there is a trade-off between generalization performance and convergence rate [Zha+19]. Wadia et al. also reported that the preconditioned methods, especially second-order methods, do not provide high generalization performance either empirically or theoretically [Wad+20]. With a simple theoretical and exhaustive empirical analysis, Wilson et al. showed that adaptive optimizers are worse at generalization than simple SGD [Wil+17]. Balles and Hennig also reported that Adam generalizes worse than Momentum SGD [BH18].

Contrary to these studies, Schneider et al. found that no single optimizer is the “best" in general [SBH19]. Similarly, Schmidt et al. claimed that the optimizer performance varies from task to task [SSH20]. Choi et al. have come to a different conclusion from all the studies cited above [Cho+19]. Choi et al. tuned not only the learning rate but also other hyperparameters of adaptive methods. As a result, they found that well-tuned adaptive optimizers never underperform simple gradient methods. The study of Choi et al. is a solid report in that it is the most exhaustive hyperparameter tuning of any study to date.

These studies provided insights on how optimizer selection influences generalization. However, the focus of these studies was on the classical supervised learning setting, in which the test distribution is assumed to be the same as the training one. Our research differs from them in that we investigate the influence of optimizer selection on the OOD generalization.

B.2 Out-of-Distribution Generalization Problem

Taming the distributional shift is a big challenge in machine learning research [SK12; Ben+10; PY09; Sze+14; Arj+19]. Geirhos et al. argued that many modern deep neural network models sometimes learn shortcut features instead of intended features and overfit to a specific dataset [Gei+20]. These models show high in-distribution performance, making predictions using shortcut features but performing poorly in OOD environments. Here, the choice of hyperparameters and optimizers is an inductive bias in learning, leading to convergence to a different local minimum.

Some studies have focused on generalization with adversarial noise to evaluate the impact of optimizer selection on OOD generalization. For instance, the theoretical and empirical analysis by Khoury showed that SGD is more robust against adversarial noise than the adaptive optimizers [Kho19]. Wang et al. argued that adversarial examples to some methods are not necessarily adversarial to others [Wan+19]. Other related works also pertain to OOD problems. For example, Abdelzad et al. found that the best optimizers for OOD detection vary by experimental setting [ACS20]. Metz et al. reported that a learned optimizer somehow unexpectedly outperformed a human-designed optimizer in terms of the OOD generalization [Met+20].

These studies provide valuable insights on optimizer selection for the OOD problem. However, we emphasize that the previous works discussed above explored hyperparameters in a limited range. Khoury conducted the most exhaustive hyperparameter search but searched only the learning rate for adaptive optimizers [Kho19]. As we briefly explain in Section 1, the exhaustiveness of the hyperparameter search is crucial for empirical investigation of an optimizer’s effect. Thus, we basically follow [Cho+19], which most exhaustively searched hyperparameters for optimizer comparison and explored more hyperparameters than did previous studies.

Here we emphasize that only shifts such as adversarial noise have been studied in previous studies of optimization selection for OOD generalization. Thus, we use a much more diverse set of real OOD datasets, including image classification and NLP tasks where the distributional shift is significant, covariate shift, correlation shift, subpopulation shift, and background shift, not only

domain generalization. The set of datasets we explored is the most exhaustive for evaluating the optimizer's role in OOD generalization, as far as we know.

C Preliminaries

C.1 Optimizers Subjected to Our Analysis

Similar to previous studies [Wil+17; SBH19; Cho+19], we compare two types of optimizers. The first one is non-adaptive optimizers. The update equation at iteration t of model parameter $\boldsymbol{\theta}_t$ is as follows:

$$\boldsymbol{v}_t \leftarrow \gamma \boldsymbol{v}_{t-1} + \eta_t \tilde{\nabla}_{\boldsymbol{\theta}_{t-1}} \ell(\boldsymbol{\theta}_{t-1}), \quad \boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \boldsymbol{v}_t \tag{1}$$

where η_t is learning rate, $\ell(\boldsymbol{\theta})$ is the loss, and $\tilde{\nabla}_{\boldsymbol{\theta}_{t-1}}$ is the stochastic gradient, in the particular case of stochastic gradient descent, $\gamma = 0$. Optimizers with momentum terms such as Momentum SGD [Pol64], and Nesterov momentum [Nes03] are also classified as non-adaptive optimizers, and γ is the parameter for controlling the momentum term. For Nesterov momentum, $\ell(\boldsymbol{\theta}_{t-1})$ should be replaced $\ell(\boldsymbol{\theta}_{t-1} - \gamma \boldsymbol{v}_{t-1})$.

The second type of optimizer is adaptive methods.

Adam and RMSprop are adaptive optimizers and they can be written in the form of the generic adaptive optimization method. The generic adaptive optimization method can be written as in Algorithm 1. This is based on what [Liu+20] and [RKK18] propose.

Algorithm 1 Generic adaptive optimization method setup.

Require: $\{\eta_t\}_{t=1}^T$: step size, $\{\phi_t, \psi_t\}_{t=1}^T$ function to calculate momentum and adaptive rate, $\boldsymbol{\theta}_0$: initial parameter, $\ell(\boldsymbol{\theta})$: objective function

1: **for** $t = 1$ to T **do**
2: $\quad \boldsymbol{g}_t \leftarrow \tilde{\nabla}_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta}_{t-1})$ (Calculate stochastic gradients w.r.t. objective at timestep t)
3: $\quad \boldsymbol{w}_t \leftarrow \phi_t(\boldsymbol{g}_1, ..., \boldsymbol{g}_t)$ (Calculate momentum)
4: $\quad \boldsymbol{l}_t \leftarrow \psi_t(\boldsymbol{g}_1, ..., \boldsymbol{g}_t)$ (Calculate adaptive learning rate)
5: $\quad \boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \boldsymbol{w}_t \boldsymbol{l}_t$ (Update parameters)
6: **end for**

The choice of optimizers we study is in line with previous work on optimizer comparison [Cho+19] and with most recent OOD work; those studies overwhelmingly focused on Adam rather than e.g. AdamW. It is this overwhelming popularity that informed our study design: studying Adam yields more relevant results for current practice.

A concrete formulation of these five optimizer update formulas is as follows.

C.1.1 SGD

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \tilde{\nabla}_{\boldsymbol{\theta}_{t-1}} \ell\left(\boldsymbol{\theta}_{t-1}\right) \tag{2}$$

C.1.2 Momentum-SGD

$$\boldsymbol{v}_t \leftarrow \gamma \boldsymbol{v}_{t-1} + \eta_t \tilde{\nabla}_{\boldsymbol{\theta}_{t-1}} \ell\left(\boldsymbol{\theta}_{t-1}\right) \tag{3}$$

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \boldsymbol{v}_t \tag{4}$$

C.1.3 Nesterov Momentum

$$\boldsymbol{v}_t \leftarrow \gamma \boldsymbol{v}_{t-1} + \eta_t \tilde{\nabla}_{\boldsymbol{\theta}_{t-1}} \ell\left(\boldsymbol{\theta}_{t-1} - \gamma \boldsymbol{v}_{t-1}\right) \tag{5}$$

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \boldsymbol{v}_t \tag{6}$$

C.1.4 RMSprop

$$\boldsymbol{v}_t \leftarrow \alpha \boldsymbol{v}_{t-1} + (1-\alpha)\tilde{\nabla}_{\boldsymbol{\theta}_{t-1}} \ell\left(\boldsymbol{\theta}_t\right)^2 \tag{7}$$

$$\boldsymbol{m}_t \leftarrow \gamma \boldsymbol{m}_{t-1} + \frac{\eta_t}{\sqrt{\boldsymbol{v}_t + \epsilon}} \tilde{\nabla}_{\boldsymbol{\theta}_{t-1}} \ell\left(\boldsymbol{\theta}_t\right) \tag{8}$$

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \boldsymbol{m}_t \tag{9}$$

C.1.5 Adam

$$\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + \left(1-\beta_1\right) \tilde{\nabla}_{\boldsymbol{\theta}_{t-1}} \ell\left(\boldsymbol{\theta}_t\right) \tag{10}$$

$$\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + \left(1-\beta_2\right) \tilde{\nabla}_{\boldsymbol{\theta}_{t-1}} \ell\left(\boldsymbol{\theta}_t\right)^2 \tag{11}$$

$$b_t \leftarrow \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \tag{12}$$

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \frac{\boldsymbol{m}_t}{\sqrt{\boldsymbol{v}_t} + \epsilon} b_t. \tag{13}$$

C.2 Out-of-Distribution Generalization Datasets

Image Classification Datasets: DomainBed consists of a set of benchmark datasets for domain generalization, which includes Colored MNIST [Arj+19], Rotated MNIST [Ghi+15], PACS [FXR13], VLCS [Li+17], Office-Home [Ven+17], Terra Incognita [BVP18], and DomainNet [Pen+19]. These datasets contain a variety of distributional shifts. VLCS is a set of different image datasets, for example, images of *birds* from several datasets. Terra Incognita is a dataset consisting of images taken by cameras at different locations. The difference in the location of the camera corresponds to the difference in the domain. PACS, Office-Home, and DomainNet are image datasets whose style varies by domain. For example, an image of PACS in one domain is photography, while that in another domain is a sketch. Rotated MNIST is an artificially generated dataset of domains that have been given different rotation angles.

Colored MNIST is an *anomaly* in the DomainBed dataset. $P(Y|X)$ remains the same for all datasets (covariate shift) except in Colored MNIST. This dataset is designed to make models fail by exploiting spurious correlations in training environments. In particular, the dataset is such that the model can only exploit the source of spurious correlation (color) and achieve a very high training accuracy without relying on the true invariant source of correlation (shape). In contrast, none of the other datasets have such strong spuriousness. The strength of the spurious correlation flips in the test domain and thus it induces a negative correlation between the validation and the test accuracy.

The Backgrounds Challenge dataset measures a model's robustness against background shift [Xia+21]. A model is trained on an image and evaluated on the same image with a different background. If the model exploits the background features during training, it will be fooled during evaluation. Therefore, if the model strongly depends on the background, this dataset is a difficult OOD dataset, while if the model does not, this dataset is easy to model.

Natural Language Processing (NLP) Datasets: The CivilComments-WILDS dataset is cast as a subpopulation shift problem. The shift problem tackles a binary classification problem that predicts the toxicity of comments on articles, and domain vectors are assigned according to whether the comment refers to one of eight demographic identities. In the subpopulation shift, the test distribution is a subpopulation of the training distribution.

The Amazon-WILDS dataset has the characteristics of a hybrid shift of a subpopulation shift and domain shift. This dataset is cast as the problem of estimating a rating of 1–5 from the rating comments of each user. In this kind of dataset, each user corresponds to a domain, and the goal is to produce a high performance for all user comments.

C.2.1 DomainBed

DomainBed consists of sub-datasets shown in Table 2, where we exclude Terra Incognita and Rotated MNIST as stated in Section 2.1. We summarize the dataset information in the Table by referring to [GL21].

Table 2: DomainBed: Dataset Information

	domain	examples	class
Colored MNIST	[0.1, 0.2, 0.9]	70000	2
Rotated MNIST	[0, 15, 30, 45, 60, 75]	70000	10
PACS	[art, cartoons, photos, sketches]	9991	7
VLCS	[Caltech101, LabelMe,415SUN09, VOC2007]	10729	5
Office-Home	[art, clipart, product, real]	15588	65
TerraIncognita	[L100, L38, L43, L46]	24788	10
DomainNet	[clipart, infograph, painting, quickdraw,420real, sketch]	586575	345

Colored MNIST is a dataset for binary classification of MNIST dataset [Arj+19]. The digits from 0 to 4 are labeled 0, and those greater than 5 are labeled 1, and the task is to classify these classes successfully. However, each digit is also colored by either red or green. This is for having models to confuse the important feature for classification. The domain d indicates the correlation of the label with color. For example, if the domain is 0.1, the correlation between, say red, with the number smaller than 5 is 0.1. Furthermore, the label is flipped at a constant rate: in this paper, 15 % label is flipped. Therefore, the correlation between color and digit is d, while between label and digits is 0.85. That is, what models should learn is the correlation between label and noise, resulting in classification accuracy of 0.85. However, if the model exploits spurious correlation of the domain, it will learn the correlation between label and color, resulting in training accuracy being 0.9 but test accuracy being 0.1 in this case.

PACS and Office-Home are image datasets whose domain determines the style of the image. These are benchmark datasets for domain generalization.

VLCS is a set of different photographic datasets, PASCAL VOC [Eve+10], LabelMe [Rus+08], Caltech101 [FFP04], and SUN09 [Cho+10]. PASCAL VOC, LabelMe, and SUN09 are benchmark datasets for object detection. Caltech101 is 101 classes image datasets, where each class has 40 - 80 samples.

DomainNet is a large dataset proposed for the study of domain generalization. The number of classes, domains, and dataset size is the largest in the DomainBed dataset.

TerraIncognita is a dataset consisting of images taken by cameras at different locations. The difference in the camera's location corresponds to the difference in the domain.

RotatedMNIST is a dataset that artificially rotates MNIST and divides the domain according to the rotation angle. The number in the domain corresponds to the rotation angle.

We leave one domain for the final evaluation and use the remaining domains for training. To evaluate the performance during training, we split the data of each domain into training data and validation data. The split ratio is 80 % for training and 20 % for validation. We take an average of test accuracies and validation accuracies across domains, respectively, and use them to evaluate the OOD generalization.

C.2.2 Backgrounds Challenge Dataset

In Section E.2, we explained that we use the subset of ImageNet (ORIGINAL). ORIGINAL consists of nine classes displayed in table 3. These classes are synthetically created from ImageNet classes based on WordNet ID. This table is a copy of a table in the original paper [Xia+21].

This dataset is filtered to balance samples across classes. We follow the same filtering procedure as the original paper. For further details, please refer to the original paper [Xia+21].

C.2.3 Amazon-WILDS and CivilComments-WILDS Dataset

WILDS is a set ob benchmark datasets with distributional shift and their variants: iWildCam [BCG20], Camelyon17 [Ban+18], RxRx1 [Tay+19], OGB-MolPCBA [Hu+20], GlobalWheat [Dav+20; Dav+21], CivilComments [Bor+19], FMoW [Chr+18], PovertyMap [Yeh+20], Amazon [NLM19], and Py150 [Lu+21; RBV16]. We use CivilComments and Amazon for our experiment.

Table 3: Backgrounds Challenge: Dataset information originally created in [Xia+21]

classes	WordNet ID	num sub-classes
Dog	n02084071	116
Bird	n01503061	52
Vehicle	n04576211	42
Reptile	n01661091	36
Carnivore	n02075296	35
Insect	n02159955	27
Instrument	n03800933	26
Primate	n02469914	20
Fish	n02512053	16

D Implementation and Environment for Experiments

We perform our experiment with ABCI (AI Bridging Cloud Infrastructure), a supercomputer owned by the National Institute of Advanced Industrial Science and Technology, and TSUBAME, a supercomputer owned by the Tokyo Institute of Technology. The total amount of computation is 90,436 GPU hours.

All codes for experiments are modifications of the codes provided by the authors who introduced the datasets [GL21; Koh+21; Xia+21]. Licenses of the codes are MIT license for DomainBed [GL21] and WILDS [Koh+21]. The code of Backgrounds Challenge does not indicate the license. Our code can be found at the link below.
https://github.com/Hiroki11x/Optimizer_Comparison_OOD

E Experimental Protocol

Hyperparameter Tuning:

The hyperparameters are tuned using Bayes optimization functionality of Weights&Biases[4] by evaluating in-distribution validation accuracy. Bayesian optimization sequentially explored the potential hyperparameter candidate points, and we evaluated all the trained models in the search process for comparison. As a confirmation of the soundness of our hyperparameter search, Appendix J.1 shows the histogram that the explored hyperparameters are drawn from reasonably wide range. The data shown in the box plot (Figure 3, 4, and 5) are from the evaluation of several trained models. The number of epochs (the steps budget) used is in line with previous studies [GL21; Koh+21; Cho+19] to ensure the soundness of our experimental design. Since we use the fixed epoch, it might seem to be unfair than when tuning the epoch as well for the optimizer that converges faster. However, we studied the effect of early stopping, which corresponds to the tuned epoch in Appendix I.4 and the result confirms that employing a fixed epoch does not impair the fairness of our comparison experiments. We discuss the details in Appendix I.4.

Boxplot:

We believe that sharing the whole distribution of tuning outcomes is important because: it gives an idea of how sensitive methods are to tuning and how much tuning effort is required. We also share the raw data as scatter plots (Figure 22, 23, and 24) in Appendix H.3.

Model Selection Method and Evaluation Metrics:

In the training phase of DomainBed datasets, we do not access the data in the test domain but split data from the training domains into a training dataset and validation dataset. We choose the model with the highest average performance (accuracy) on the validation data in the training domain. As a metric for evaluation, we evaluated the generalization performance in the test domain as the OOD accuracy.

The Backgrounds Challenge uses Imagenet-1k as the training data set and selects models based on their accuracy on the validation data in the training domain. After that, we measure the in-distribution performance with IN9L [Xia+21], which aggregates the test data into nine classes. As for the OOD performance, we measure the classification performance on the data where the background image of IN9L is replaced with the background image of other images.

In CivilComments-WILDS, we divide the data into training, validation, and test datasets and maximize worst-group accuracy in the validation data (and by association, maximize the average accuracy over all domains). Then, we perform model selection and evaluate the OOD accuracy on the test data. For Amazon-WILDS datasets, we adopt the same hyperparameter selection strategy as that for CivilComments-WILDS. As a metric, we do not evaluate the worst-group performance but rather the 10th-percentile accuracy for the performance of each domain by following the standard federated learning literature [Cal+18].

In line with previous studies ([GL21], [Xia+21], and [Koh+21]), different benchmarks use different evaluation metrics, each of which is outlined in the following sections.

E.1 DomainBed

We follow the setting that is employed in the original paper [GL21]. We train models with training domains and evaluate their performance on the test domain, which is the domain not used for training. We use ResNet-50 for PACS, VLCS, Office-Home, DomainNet and TerraIncognita and MNIST ConvNet [GL21] for RotatedMNIST and Colored MNIST.

In our experiments, one domain is used as the test domain (out-of-distribution) and the other domains as the training domain (in-distribution). More precisely, the test domain is *"Art"* for PACS, *"Caltech101"* for VLSC, *"Art"* for Office-Home, *"Clipart"* for DomainNet, *"L100"* for TerraIncognita, *"*30°*"* for RotatedMNIST, and *"-90%"* for ColoredMNIST. We explain the experimental configurations in Section F.1.

[4] https://wandb.ai/site

E.2 Backgrounds Challenge Dataset

We follow [Xia+21] for using Backgrounds Challenge dataset for evaluation. Thus, we use the following data-generating procedure proposed by [Xia+21]. We train ResNet-50 on ImageNet-1k with two popular optimizers, Momentum SGD, and Adam. The test datasets to evaluate the trained model are derivations of ImageNet dataset. First, we construct a subset of the ImageNet which has nine coarse-grained classes, e.g. insect (*ORIGINAL*). Especially, we refer to ORIGINAL with all images from ImageNet as *IN9L*. Then, we create a dataset by changing the background of the images of IN9L. In particular, we change the background of each image into a random background cropped from another image in ORIGINAL. We call this dataset *Mixed-Random*, following [Xia+21]. By comparing the accuracy of Mixed-Same with that of ORIGINAL, we can measure the dependence of the model on the spurious correlation of background information. Thus, we investigate the relations between these two accuracies. The search range for hyperparameters is shown in Section F.2.

E.3 WILDS

We also follow the setting that is employed in the original paper [Koh+21]. First, we divide the dataset into train, validation, and test and train a model using the train data. For model selection, we use the performance evaluation of validation data. In the test dataset, considering the subpopulation shift, we measure the performance of the OOD in the worst group for CivilComments-WILDS and in the domain of 10-percentile for Amazon-WILDS. In both Amazon-WILDS and CivilComments-WILDS, DistilBERT [San+19] is used as the deep neural network model architecture. We explain the further details of the experimental configurations in Section F.3.

F Hyperparameters and Detailed Configurations

We report the hyperparameter's search space. For vanilla SGD, we search learning rate η, learning rate decay rate ρ and the timing to decay learning rate δ, and regularization coefficient of weight decay λ. When $\delta = 0.7$, it means that the learning rate decays when training passes 70 % of the total iterations. We do not search ρ and δ for DomainBed because we do not employ a learning rate schedule.

For non-adaptive momentum methods, a parameter to control momentum γ is added to the hyperparameters. For RMSprop, we further add parameters α and ϵ, which control the second-order momentum and numerical stability, respectively. Although ϵ is originally introduced for numerical stability, this parameter is found to play a crucial role in generalization performance [Cho+19]. Thus, we follow Choi et al. and vary this parameter as well.

For Adam, we add ϵ, β_1, β_2 to vanilla SGD's configuration. The parameter ϵ is the same as that for RMSprop and β_1 and β_2 control first and second-order momentum terms, respectively.

F.1 DomainBed

We conduct Bayesian optimization for hyperparameter search of DomainBed. First, we sampled hyperparameters from uniform distribution whose minimum and maximum values are shown in the table 5 and 6 Then, we conducted Bayesian optimization on these sampled candidate hyperparameters and selected some of the hyperparameters among them. Note that the values for batch size B in the tables do not indicate minimum and maximum for Bayesian optimization but those for grid search. Unlike other datasets, we implement IRMv1 in addition to ERM. IRMv1 is a heuristic optimization problem to solve IRM, introduced by Arjovsky et al [Arj+19]. Thus, we search the hyperparameters of IRMv1 as well.

IRMv1 is the following constrained optimization problem [Arj+19]:

$$\min_{\Phi:\mathcal{X}\to\mathcal{Y}} \sum_{e\in\mathcal{E}_{\text{tr}}} R^e(\Phi) + \lambda_{\text{IRM}} \left\|\nabla_{w|w=1.0} R^e(w\circ\Phi)\right\|^2, \tag{14}$$

where Φ is representation function and w is the weight on top of the function of a model $f = w \circ \Phi$. $\mathcal{X}$ is the input domain and $\mathcal{Y}$ is the output domain. The character e indicates an environment in training environment set $\mathcal{E}_{\text{tr}}$ and R^e is the risk of the environment. Because this is the optimization with regularization term, we search coefficient λ_{IRM}. In addition, we implement annealing of this coefficient and so we try various penalty annealing iterations N_{IRM} too. The basic workload is summarized in table 4.

We made two changes to the experimental setup in the original paper, as well as to the optimizer. We added regularization to the training of ColoredMNIST and RotatedMNIST to stabilize the learning. In addition, we increased the steps budget for RotatedMNIST because we observed that the training loss of RotatedMNIST was not sufficiently reduced.

Table 4: DomainBed: Workloads

Model	Dataset	Batch size	Step Budget
MNIST ConvNet [GL21]	Colored MNIST	[128, 512, 2048]	5000
MNIST ConvNet [GL21]	Rotated MNIST	[128, 512, 2048]	100K
ResNet-50	VLCS	[64, 128]	5000
ResNet-50	PACS	[64, 128]	5000
ResNet-50	Office Home	[64, 128]	5000
ResNet-50	DomainNet	[64, 128]	5000
ResNet-50	TerraIncognita	[64, 128]	5000

Table 5: DomainBed: ResNet-50

	B	η	λ	γ	α	ϵ	λ_{IRM}	N_{IRM}
SGD	[64, 128]	[1e-5, 1e-2]	[1e-6, 1e-2]	-	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
Momentum	[64, 128]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
Nesterov	[64, 128]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
RMSprop	[64, 128]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	[0, 0.999]	[1e-8, 1e-3]	[1e-1, 1e+5]	[1e+0, 1e+4]

	B	η	λ	β_1	β_2	ϵ	λ_{IRM}	N_{IRM}
Adam	[64, 128]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.999]	[0, 0.999]	[1e-8, 1e-3]	[1e-1, 1e+5]	[1e+0, 1e+4]

Table 6: DomainBed: MNIST ConvNet [GL21]

	B	η	λ	γ	α	ϵ	λ_{IRM}	N_{IRM}
SGD	[128, 521, 2048]	[1e-5, 1e-2]	[1e-6, 1e-2]	-	-	-		
Momentum	[128, 521, 2048]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
Nesterov	[128, 521, 2048]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
RMSprop	[128, 521, 2048]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	[0, 0.999]	[1e-8, 1e-3]	[1e-1, 1e+5]	[1e+0, 1e+4]

	B	η	λ	β_1	β_2	ϵ	λ_{IRM}	N_{IRM}
Adam	[128, 521, 2048]	[1e-5, 1e-2]	-	[0, 0.999]	[0, 0.999]	[1e-8, 1e-3]	[1e-1, 1e+5]	[1e+0, 1e+4]

F.2 Backgrounds Challenge Dataset

We conduct Bayesian optimization for the Backgrounds Challenge dataset as well. We use 4096 as the batch size. For all configurations other than batch size, as we follow Choi et al [Cho+19].

We conduct the hyperparameter search and restart training from scratch. Of the trained models, we evaluate trained model performance in the OOD dataset.

Table 7: Backgrounds Challenge: ResNet-50

	η	λ	γ
Momentum	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]

	η	λ	β_1	β_2	ϵ
Adam	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.999]	[0, 0.999]	[1e-8, 1e-3]

F.3 WILDS

We conduct Bayesian optimization for the hyperparameter search of WILDS. The hyperparameters are sampled by uniform distribution whose minimum and maximum values are shown in the table 8 and 9. Note that the values for batch size B is fixed in these experiments due to computational efficiency. We implement IRMv1 as well as DomainBed experiments.

For training the Amazon-WILDS and CivilComments-WILDS datasets that we used as NLP datasets, we followed the deep neural network model and training configuration proposed in the original paper. DistilBERT, a distillation of BERT Base, is used as the deep neural network model.

Table 8: WILDS-Amazon: DistilBERT

	B	η	λ	γ	α	ϵ	λ_{IRM}	N_{IRM}
SGD	[8]	[1e-5, 1e-2]	[1e-6, 1e-2]	-	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
Momentum	[8]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
Nesterov	[8]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
RMSprop	[8]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	[0, 0.999]	[1e-8, 1e-3]	[1e-1, 1e+5]	[1e+0, 1e+4]

	B	η	λ	β_1	β_2	ϵ	λ_{IRM}	N_{IRM}
Adam	[8]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.999]	[0, 0.999]	[1e-8, 1e-3]	[1e-1, 1e+5]	[1e+0, 1e+4]

Table 9: WILDS-CivilComments: DistilBERT

	B	η	λ	γ	α	ϵ	λ_{IRM}	N_{IRM}
SGD	[16]	[1e-5, 1e-2]	[1e-6, 1e-2]	-	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
Momentum	[16]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
Nesterov	[16]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	-	-	[1e-1, 1e+5]	[1e+0, 1e+4]
RMSprop	[16]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.99999]	[0, 0.999]	[1e-8, 1e-3]	[1e-1, 1e+5]	[1e+0, 1e+4]

	B	η	λ	β_1	β_2	ϵ	λ_{IRM}	N_{IRM}
Adam	[16]	[1e-5, 1e-2]	[1e-6, 1e-2]	[0, 0.999]	[0, 0.999]	[1e-8, 1e-3]	[1e-1, 1e+5]	[1e+0, 1e+4]

G Main Results

G.1 Experimental Results on DomainBed

Figure 3 shows a box plot of the difference between the average in-distribution accuracy and average OOD accuracy for ERM. In the following discussion, we call this difference a *gap* for convenience. Due to the limitation of the paper length, only the plots of PACS and Office-Home are shown here. All results, including IRM results, are shown in Appendixes H.1.1 and H.1.2.

We found two distinct optimizer effect patterns, depending on the dataset: i) PACS, Office-Home, VLCS, Terra Incognita, Rotated MNIST, and DomainNet and ii) Colored MNIST. Because these patterns appear in both the results of ERM and IRM, we focus on ERM for the explanation.

For PACS, Office Home, VLCS, TerraIncognita, RotatedMNIST, and DomainNet, the OOD accuracy is greater for non-adaptive optimizers. The gap between the mean OOD accuracy and the mean in-distribution accuracy was smaller for the non-adaptive optimizer except for TerraIncognita. This means that the models trained with the non-adaptive optimizer are more robust on average. In TerraIncognita, the non-adaptive optimizer significantly outperforms the adaptive optimizer on the average in-distribution performance and OOD performance. We note, however, that except in this problem, the adaptive optimizer achieves a smaller gap between the two. As shown in Figure 1, when comparing models with the same in-distribution performance, the non-adaptive optimizer showed higher OOD accuracy than the adaptive optimizer. The results in Figures 1 and 3 confirm that the non-adaptive method achieves higher OOD accuracy than the adaptive method, both on average and for models with the same in-distribution performance.

Colored MNIST shows the opposite pattern of these results, where the adaptive optimizer is better than the non-adaptive optimizers. As explained in Section 2.1, Colored MNIST is distinct from the other datasets. To understand why adaptive optimizers are better at OOD generalization on this dataset, we plot the time development of the average training accuracy, validation accuracy, and average test accuracy across training, as shown in Figure 35. We explain this result in Appendix I.3.

We note our considerations regarding the exceptional behavior of ColoredMNIST. ColoredMNIST is a binary classification task dataset with random labels, where spurious features are positively correlated with invariant features in in-distribution and negatively correlated with OOD. Non-adaptive optimizers learn the spurious features, achieve the oracle performance for in-distribution and perform worse than a random guess for OOD due to inverted correlations. Adaptive optimizers, in contrast, seem to overfit the training data, achieving 100% accuracy in the training set (more than oracle [5]) in this dataset, and failing to learn the spurious features. Due the aforementioned (synthetic) inverted correlations in the dataset, this overfitting behaviour in-distribution happens to favor adaptive optimizers. This behavior happens exceptionally on ColoredMNIST due to these synthetic flips in correlation. This exploitation unexpectedly enables Adam to avert being trapped in the training domain and produces better OOD generalization. We would like to study this possibility further in future research.

[5] Since the 15% of the labels were flipped as noise, even the best model that correctly learns the data rules will only perform 85%.

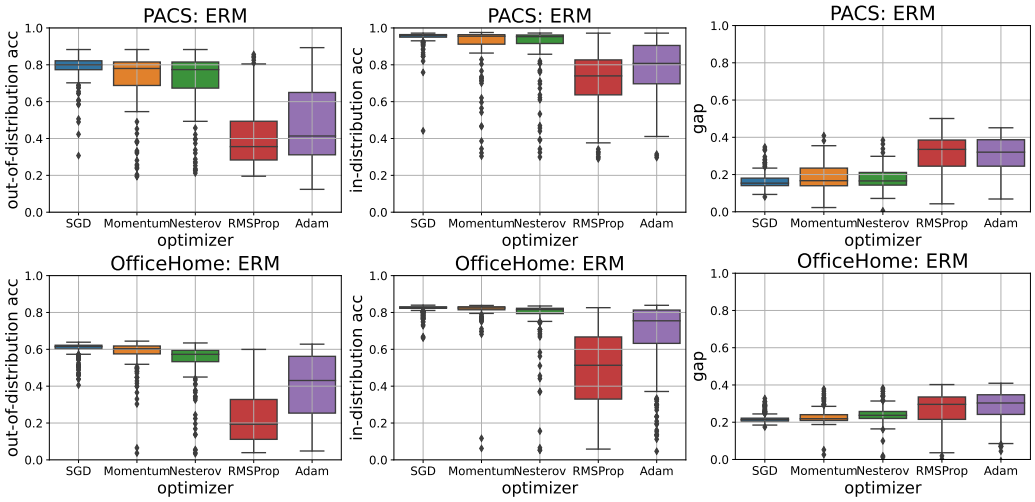


Figure 3. PACS and OficeHome in DomainBed: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers. Non-adaptive optimizers outperform the adaptive optimizers in terms of OOD generalization, and the gap between in-distribution performance and OOD performance is small. The results of other detailed dataset experiments are described in Appendix H.1.

G.2 Experimental Results on Backgrounds Challenge

Backgrounds Challenge requires training of ImageNet-1k on ResNet50 from scratch, and it takes 256 GPU hours to obtain a single trained model, so we only compared Momentum SGD with Adam. Because Momentum SGD achieved competitive performance among non-adaptive optimization methods in the DomainBed and WILDS experiments, we adopted Momentum SGD to represent non-adaptive optimization methods. In the same way, Adam outperformed RMSProp in all the experiments, so we adopted Adam as representative of the adaptive optimizers.

Figure 4 compares the accuracy for ORIGINAL (in-distribution) and Mixed-Rand (out-of-distribution). The best in-distribution performance is the same for each optimizer, but concerning the best OOD performance, the non-adaptive optimizer outperformed the adaptive optimizer. As can be seen from Figure 1 (second row, middle column), particularly in the region of high in-distribution performance on the right, the OOD performance of Momentum SGD exceeds that of Adam.

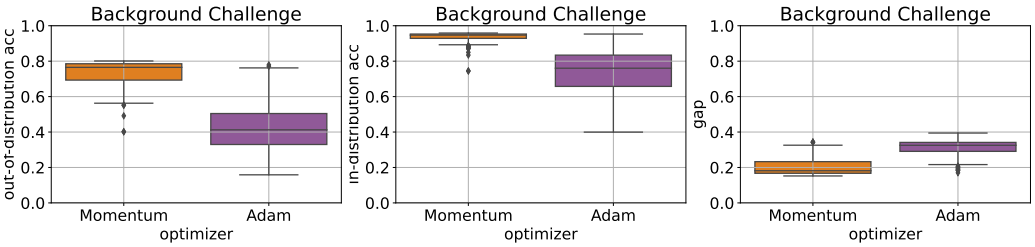


Figure 4. Backgrounds Challenge: Comparison of the validation accuracy and the test accuracy of ERM across two optimizers. In terms of OOD generalization, Momentum SGD outperforms Adam in both average and best performance. The highest value of in-distribution accuracy remains the same, but Momentum SGD shows higher performance on average.

G.3 Experimental Results on WILDS

A comparison of in-distribution and OOD averages for WILDS is shown in Figure 5. It can be clearly seen that the adaptive optimizer is fit too well to the in-distribution in the WILDS problem setting. For Amazon-WILDS and CivilComment-WILDS, as in DomainBed and Backgrounds Challenge, the non-adaptive optimizer outperforms the adaptive optimizer in terms of OOD generalization. The gap between in-distribution accuracy and OOD accuracy is also tiny for non-adaptive optimizers. In particular, the CivilComments-WILDS experimental result is remarkable, as both non-adaptive and adaptive optimizers show similar high in-distribution performance, but in the OOD environment, adaptive methods significantly fail to make inferences.

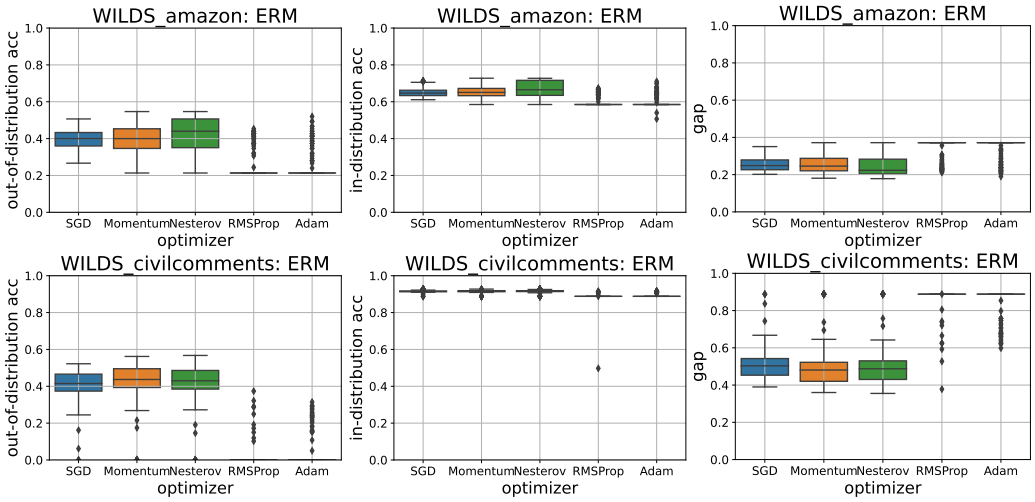


Figure 5. Amazon-WILDS and CivilComments-WILDS: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers. Both non-adaptive and adaptive optimizers have similar in-distribution accuracy, but the adaptive method significantly degrades the performance of the OOD generalization.

G.4 Experimental Results: Correlation Behaviors

The increasing return is an example, as shown in the leftmost part of Figure 2. The increasing returns in large regions of the in-distribution generalization significantly affect the OOD generalization, suggesting that the last tuning is significant for the OOD generalization, as seen in all domain generalization problems except for DomainNet.

The linear return is as shown in the middle of Figure 2. The OOD accuracy increases linearly with the in-distribution accuracy. This is generally the same result for in-distribution validation and test.

The diminishing return is an example like that shown in the rightmost part of Figure 2, where the increase in a large region of the in-distribution is saturating the OOD generalization with a small percentage of its effect. This behavior suggests that the effort of tuning at the end is often not worth it to improve OOD generalization. In addition to Amazon-WILDS, we have seen a similar trend in problem settings with subpopulation shifts, such as CivilComments-WILDS.

H Full Results of Experiments

H.1 Full Results of Boxplot

Since we could not include all the results in the main paper, we report all the OOD acc, In-distribution acc and their difference (we denote as gap) results including ERM and IRM results in a box plot.

H.1.1 Full Results of Filtered Boxplot

What we want to find out is the OOD generalization performance for models that perform better than random guess in the In-distribution test set, i.e., models that have been trained. As Filtered Results, we define random guess $= 1/\text{num of classes}$ in each problem setting, and show the results of eliminating those models whose in-distribution accuracy does not exceed this threshold.

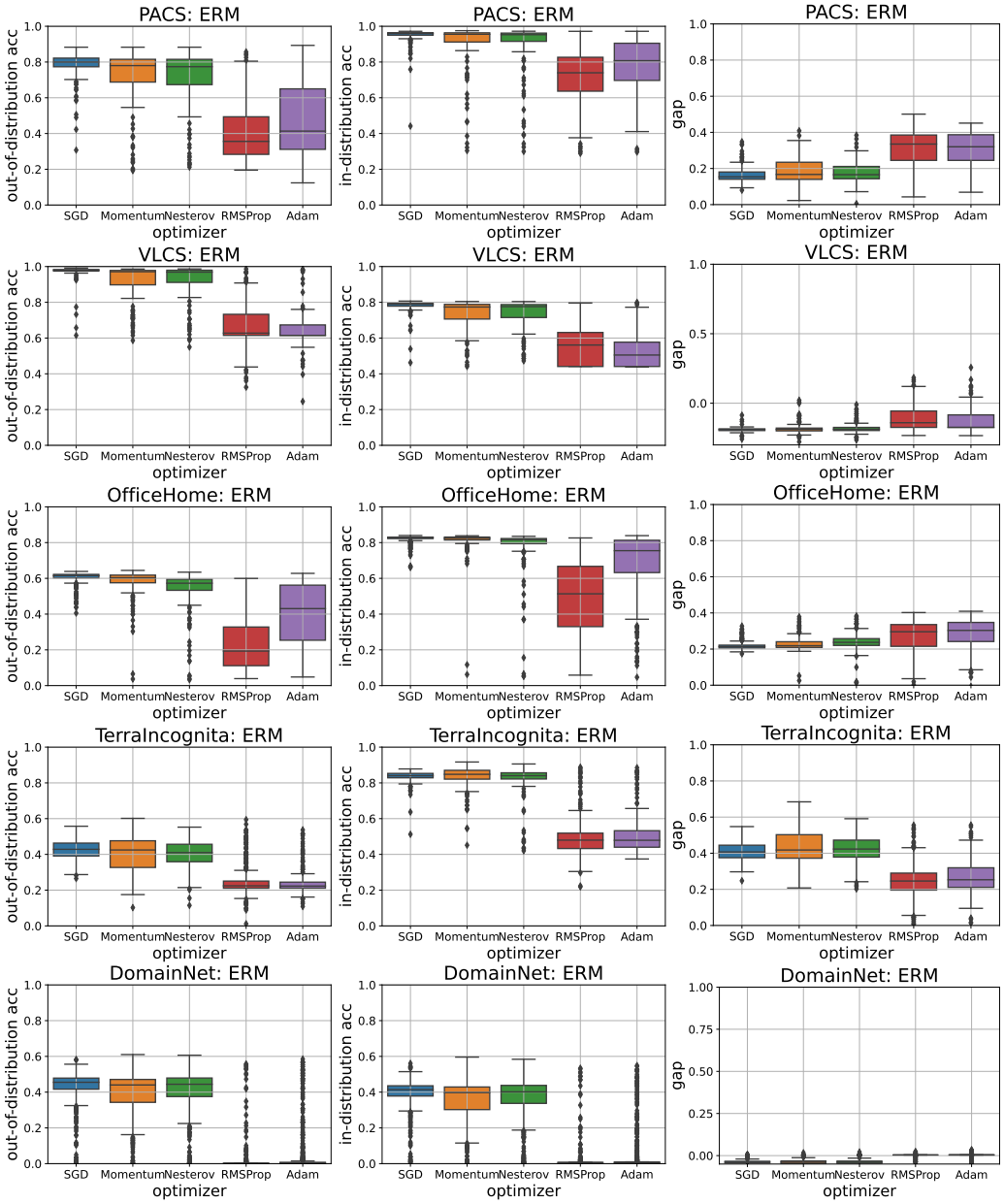


Figure 6. Filtered Results of PACS, VLCS, OfficeHome, TerraIncognita, and DomainNet in DomainBed: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers.

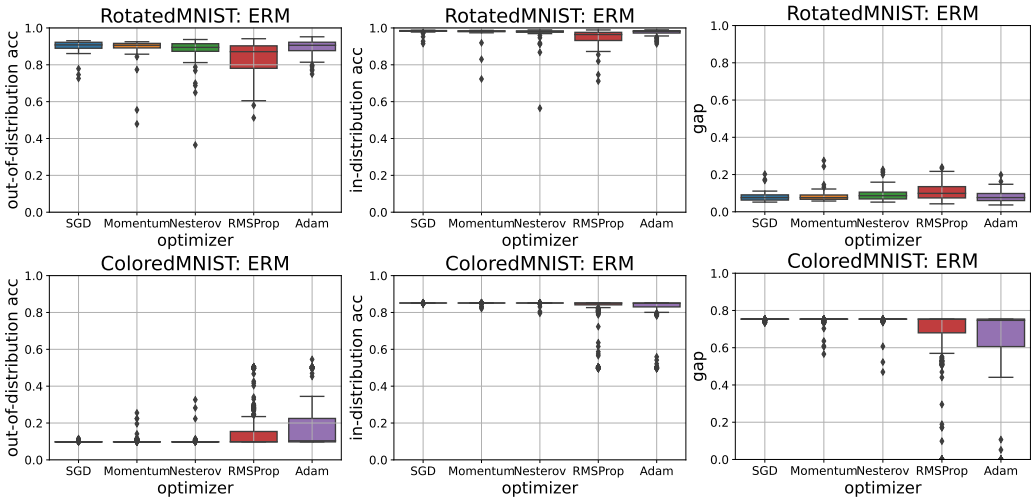


Figure 7. Filtered Results of RotatedMNIST and ColoredMNIST in DomainBed: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers.

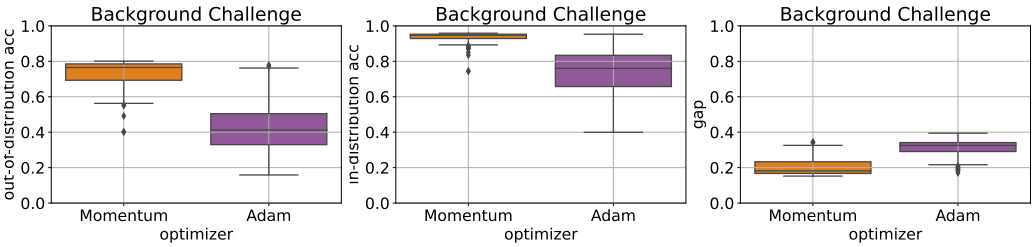


Figure 8. Filtered Results of Backgrounds Challenge: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers.

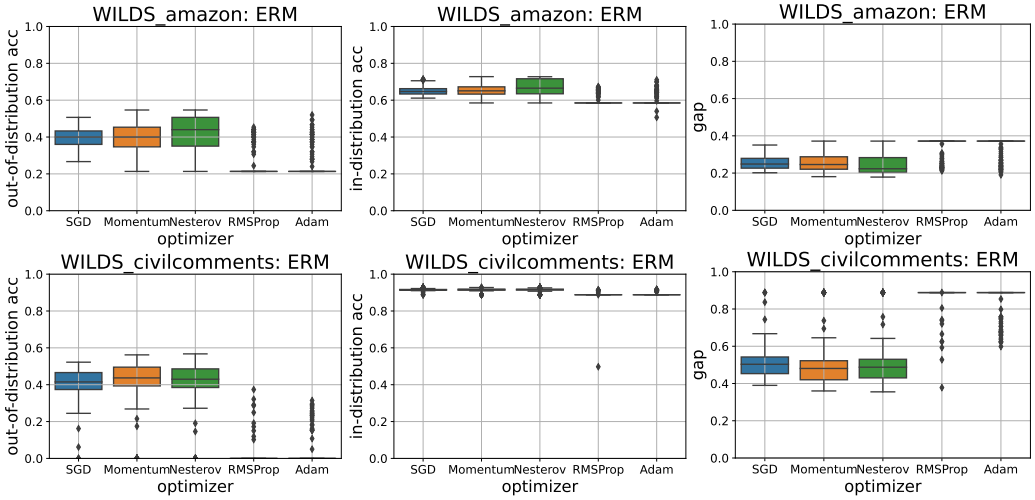


Figure 9. Filtered Results of Amazon-WILDS and CivilComments-WILDS: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers.

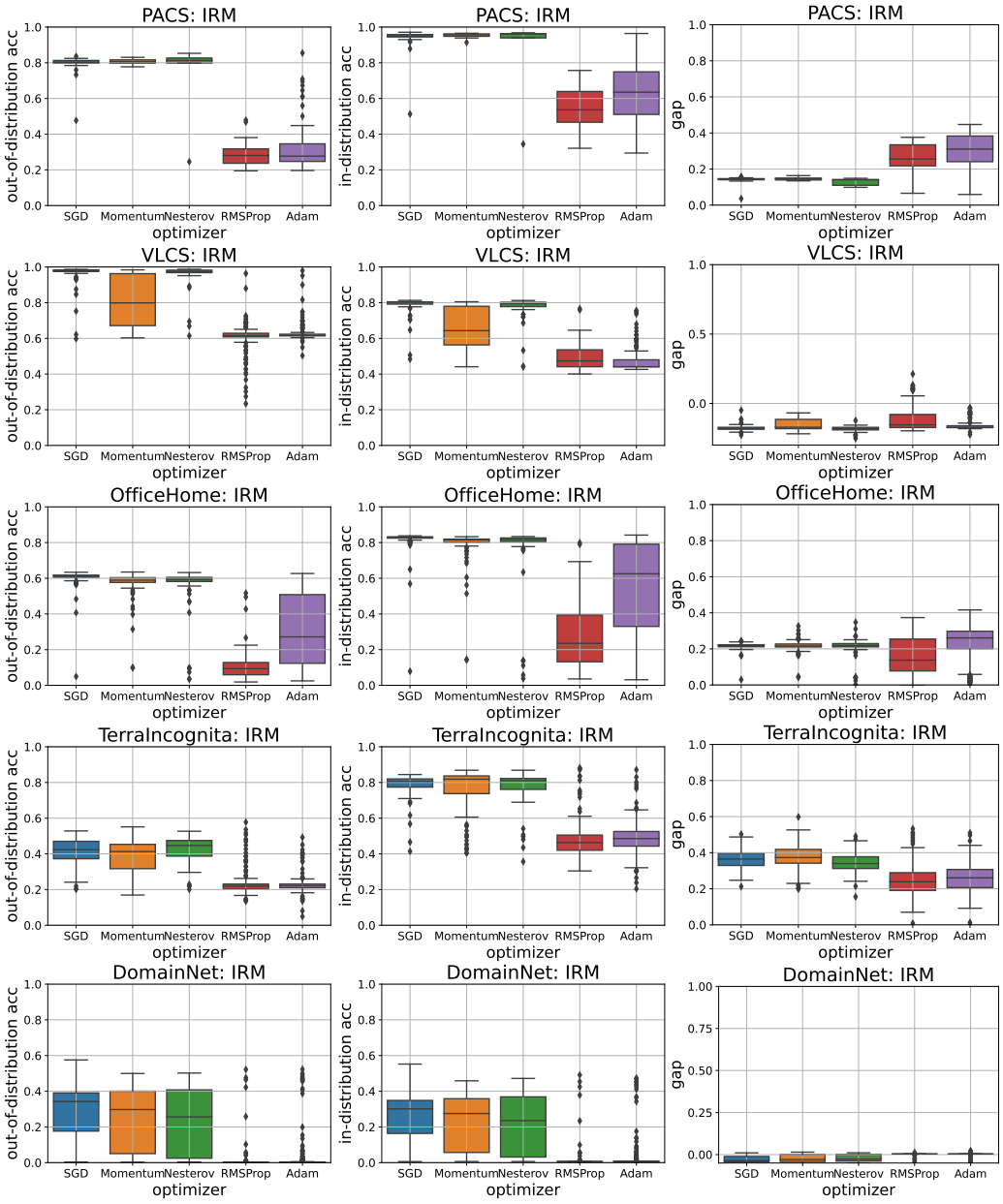


Figure 10. Filtered Results of PACS, VLCS, OfficeHome, TerraIncognita and DomainNet in DomainBed: Comparison of the validation accuracy and the test accuracy of IRM across five optimizers.

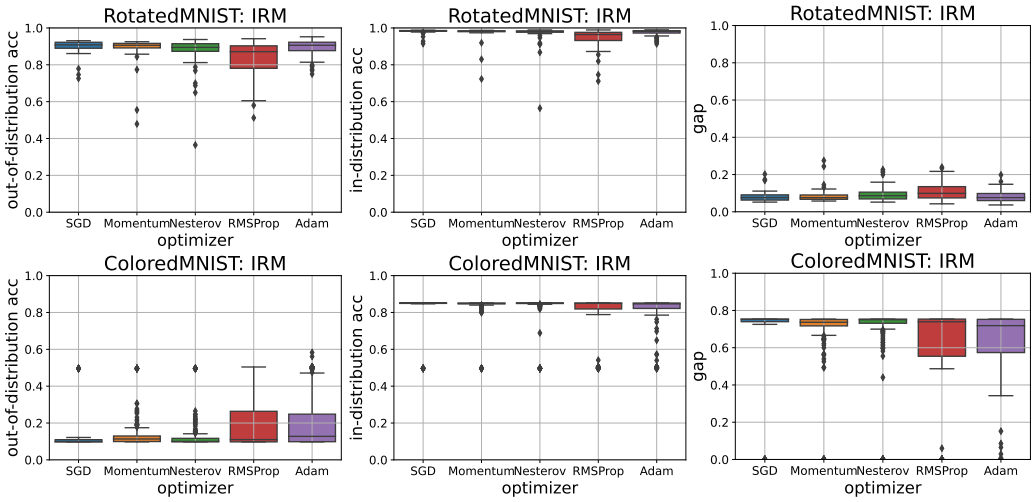


Figure 11. Filtered Results of RotatedMNIST and ColoredMNIST in DomainBed: Comparison of the validation accuracy and the test accuracy of IRM across five optimizers.

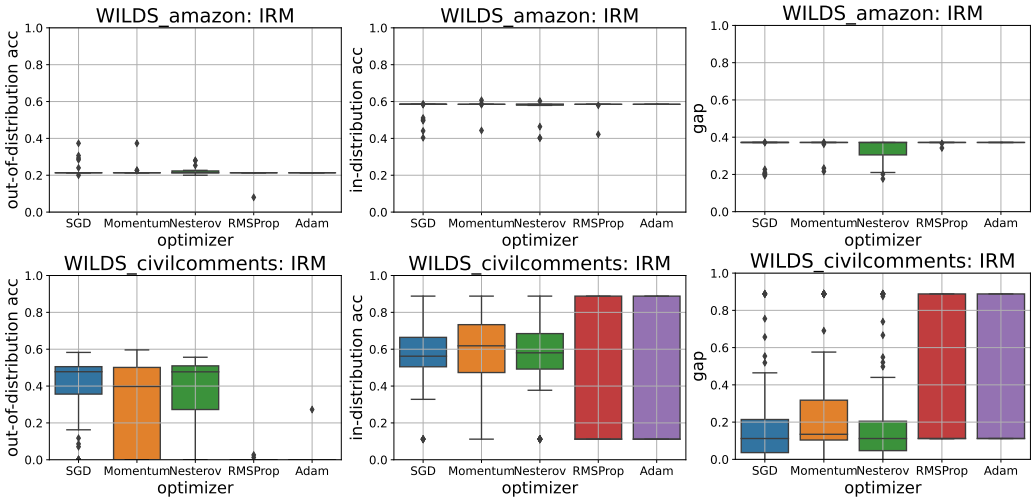


Figure 12. Filtered Results of Amazon-WILDS and CivilComments-WILDS: Comparison of the validation accuracy and the test accuracy of IRM across five optimizers.

H.1.2 Full Results of Non-Filtered Boxplot

In the filtered results, we excluded the evaluation of models that failed to learn, but if we want to discuss the possibility of learning failure, it is important to look at the unfiltered data. Therefore, we also created a box plot for models that performed below a random guess, without excluding them.

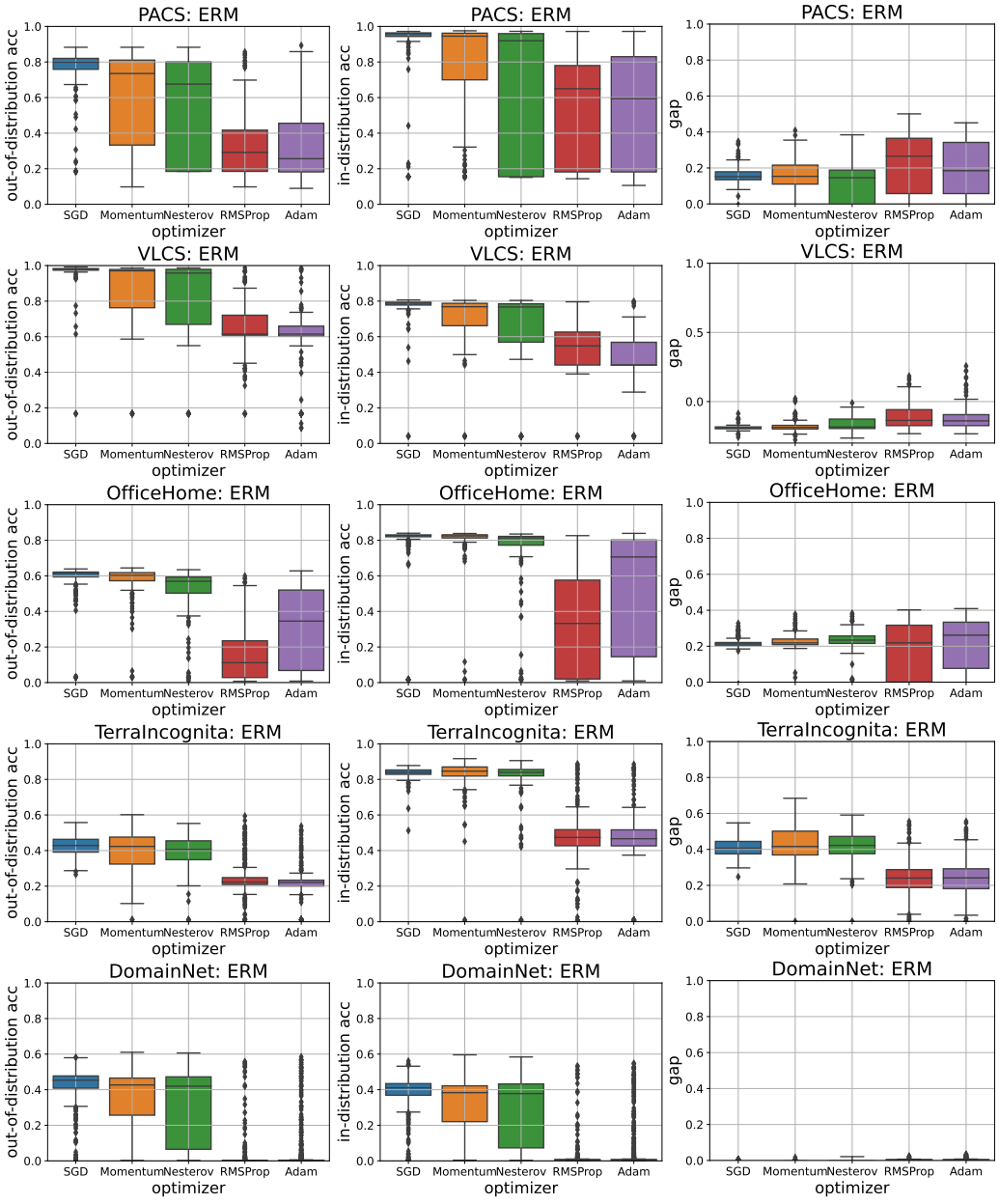


Figure 13. Non-Filtered Results of PACS, VLCS, OfficeHome, TerraIncognita, and DomainNet in DomainBed: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers.

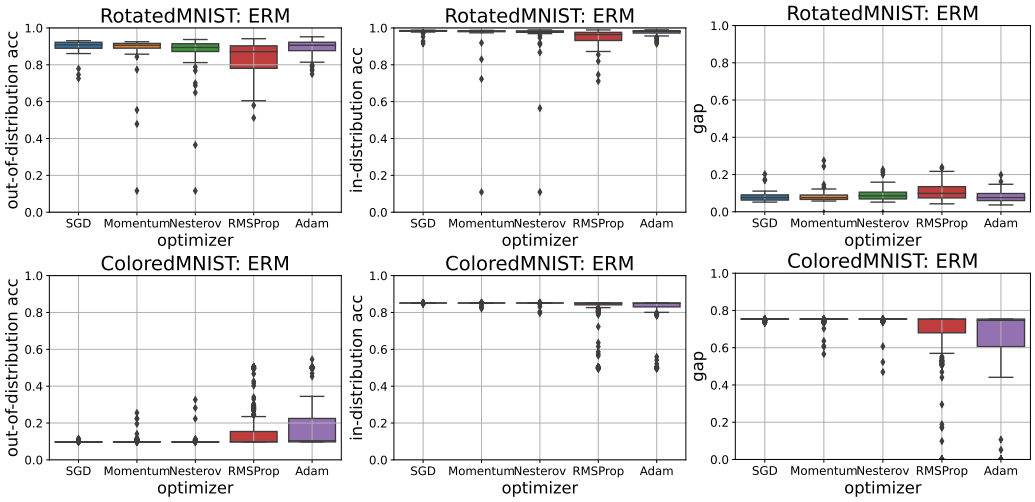


Figure 14. Non-Filtered Results of RotatedMNIST and ColoredMNIST in DomainBed: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers.

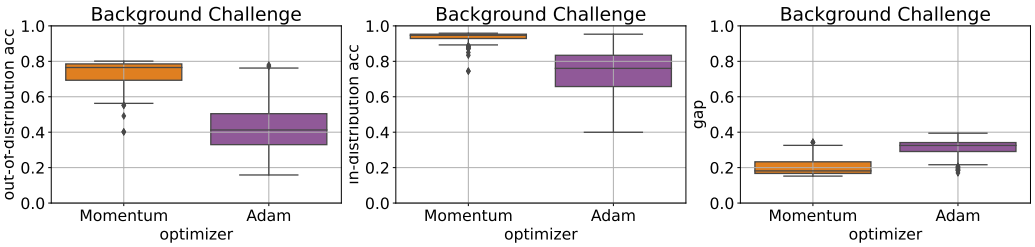


Figure 15. Non-Filtered Results of Backgrounds Challenge: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers.

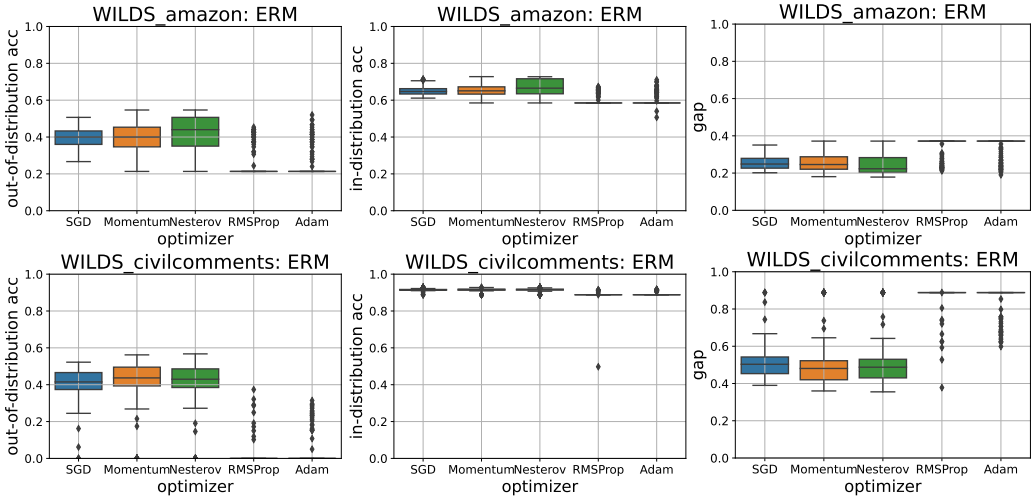


Figure 16. Non-Filtered Results of Amazon-WILDS and CivilComments-WILDS: Comparison of the validation accuracy and the test accuracy of ERM across five optimizers.

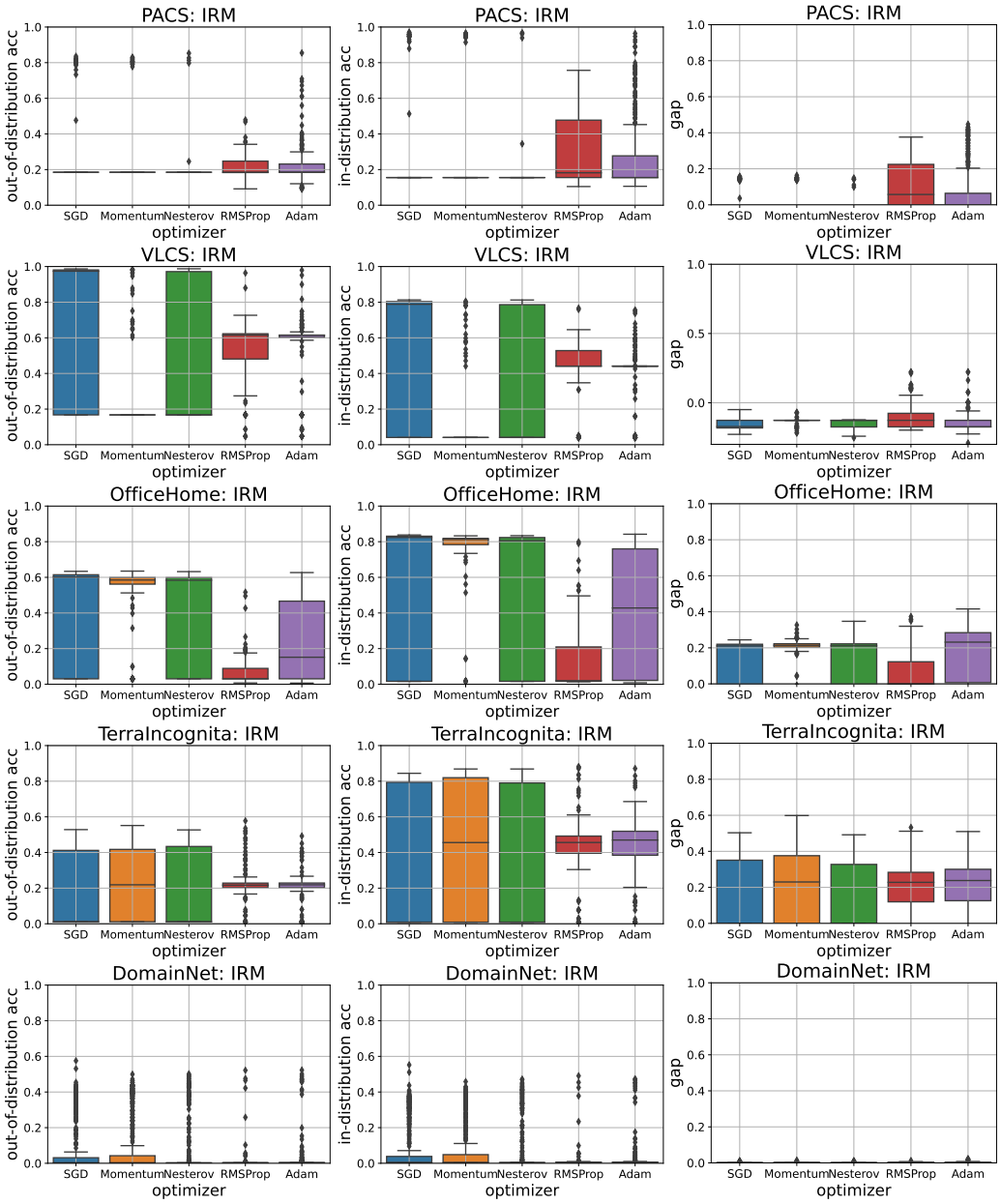


Figure 17. Non-Filtered Results of PACS, VLCS, OfficeHome, TerraIncognita, and DomainNet in DomainBed: Comparison of the validation accuracy and the test accuracy of IRM across five optimizers.

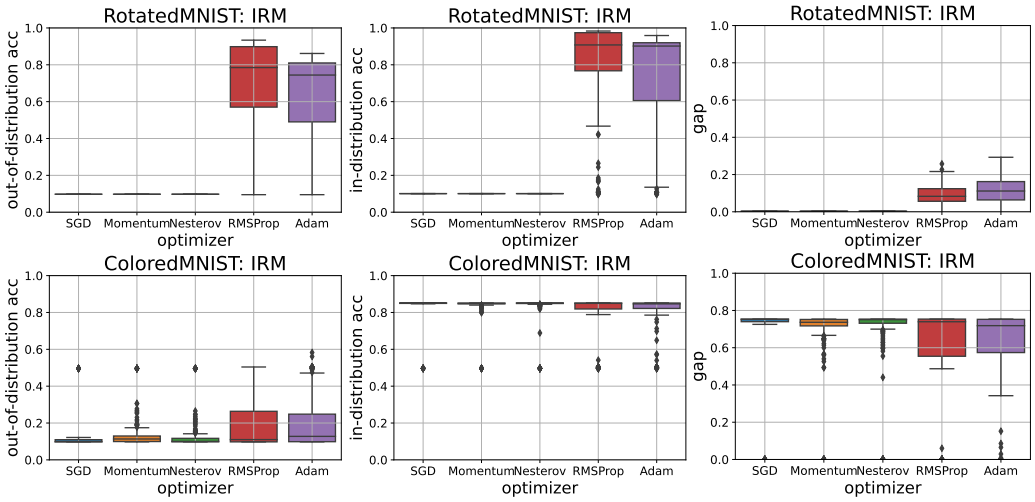


Figure 18. Non-Filtered Results of RotatedMNIST and ColoredMNIST in DomainBed: Comparison of the validation accuracy and the test accuracy of IRM across five optimizers.

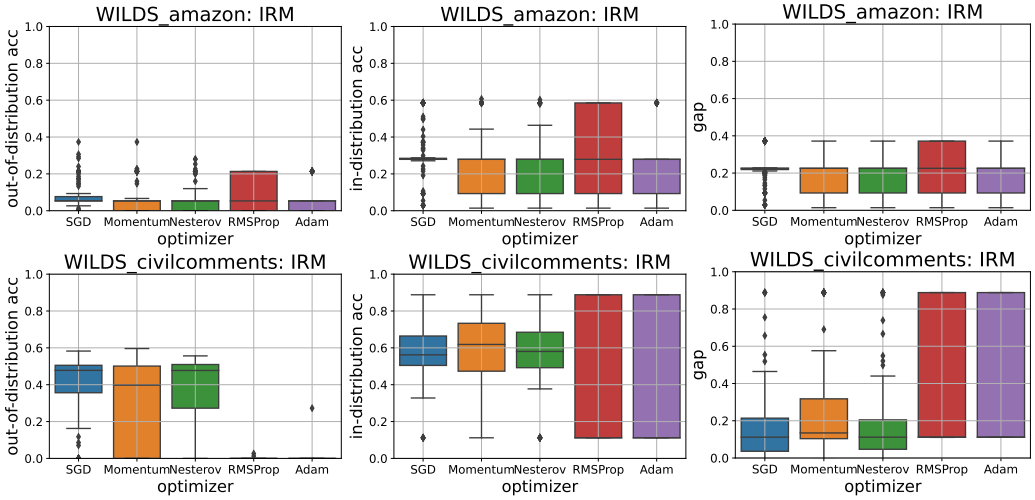


Figure 19. Non-Filtered Results of Amazon-WILDS and CivilComments-WILDS: Comparison of the validation accuracy and the test accuracy of IRM across five optimizers.

H.2 Full Results of Reliability Diagram Like Plots

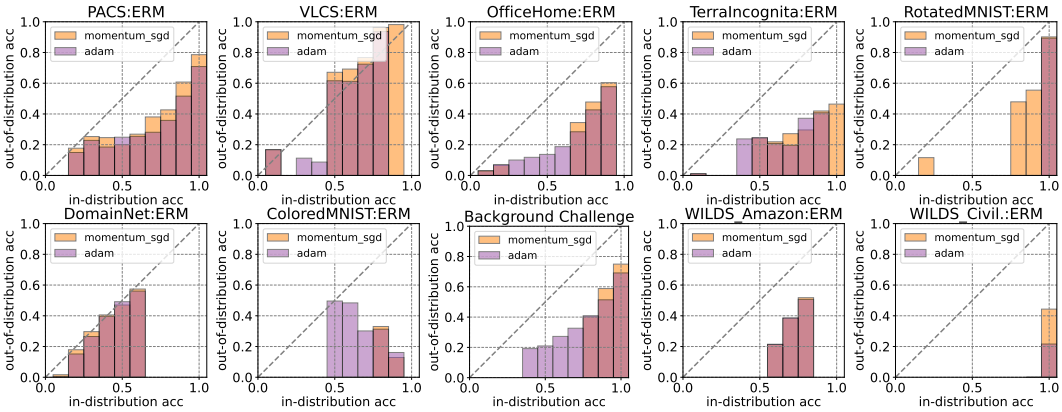


Figure 20. DomainBed, Backgrounds Challenge, Amazon-WILDS, and CivilComments-WILDS: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of ERM between Momentum SGD and Adam. Since Adam showed better OOD performance than RMSProp, Adam is presented as a representative of adaptive methods. Momentum SGD shows competitive performance in OOD with Vanilla SGD and Nesterov Momentum and is a representative of non-adaptive methods.

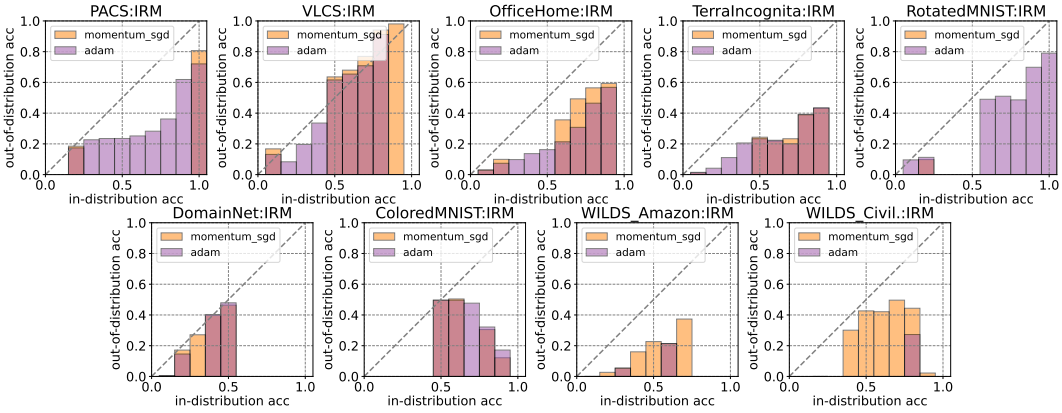


Figure 21. DomainBed, Amazon-WILDS, and CivilComments-WILDS: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of IRM between Momentum SGD and Adam. Since Adam showed better OOD performance than RMSProp, Adam is presented as a representative of adaptive methods. Momentum SGD shows competitive performance in OOD with Vanilla SGD and Nesterov Momentum and is a representative of non-adaptive methods.

H.3 Full Results of Scatter Plots

The results of the comparison of OOD accuracy and in-distribution accuracy shown in Section 3.3 are shown below for all benchmarks.

The box plots shown in Section H.1 and the Reliability Diagram Like Plot shown in Section H.2 are based on the data shown in the Scatter Plot shown below.

H.3.1 ERM

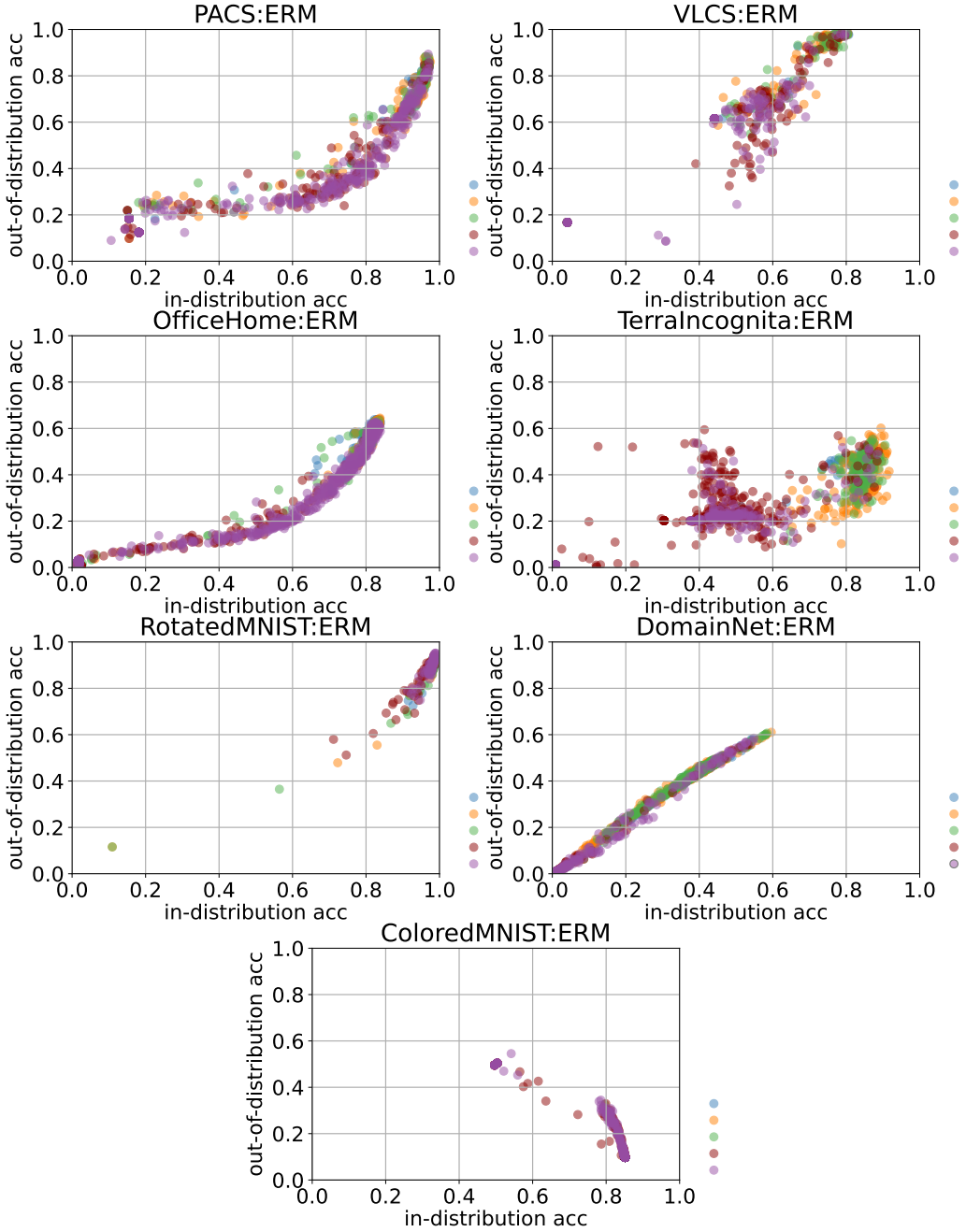


Figure 22. DomainBed: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of ERM across optimizers. The legend circles on the right side of each figure show, in order, VanillaSGD, Momentum SGD, Nesterov Momentum, RMProp, and Adam. The difference in each data point indicates the difference in hyperparameter configuration.

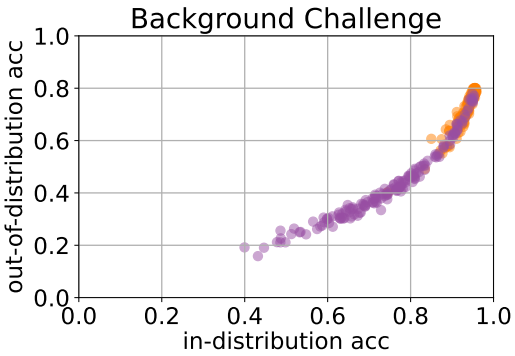


Figure 23. Backgrounds Challenge: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of ERM across optimizers. The legend circles on the right side of each figure show, in order, VanillaSGD, Momentum SGD, Nesterov Momentum, RMProp, and Adam. The difference in each data point indicates the difference in hyperparameter configuration.

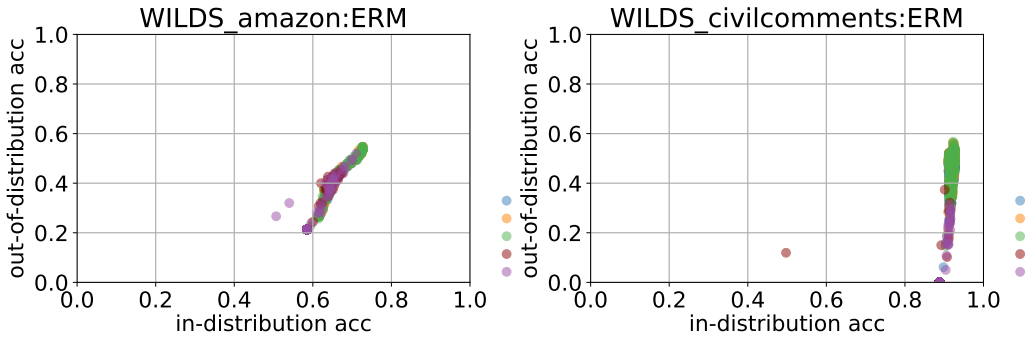


Figure 24. WILDS: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of ERM across optimizers. The legend circles on the right side of each figure show, in order, VanillaSGD, Momentum SGD, Nesterov Momentum, RMProp, and Adam. The difference in each data point indicates the difference in hyperparameter configuration.

H.3.2 IRM

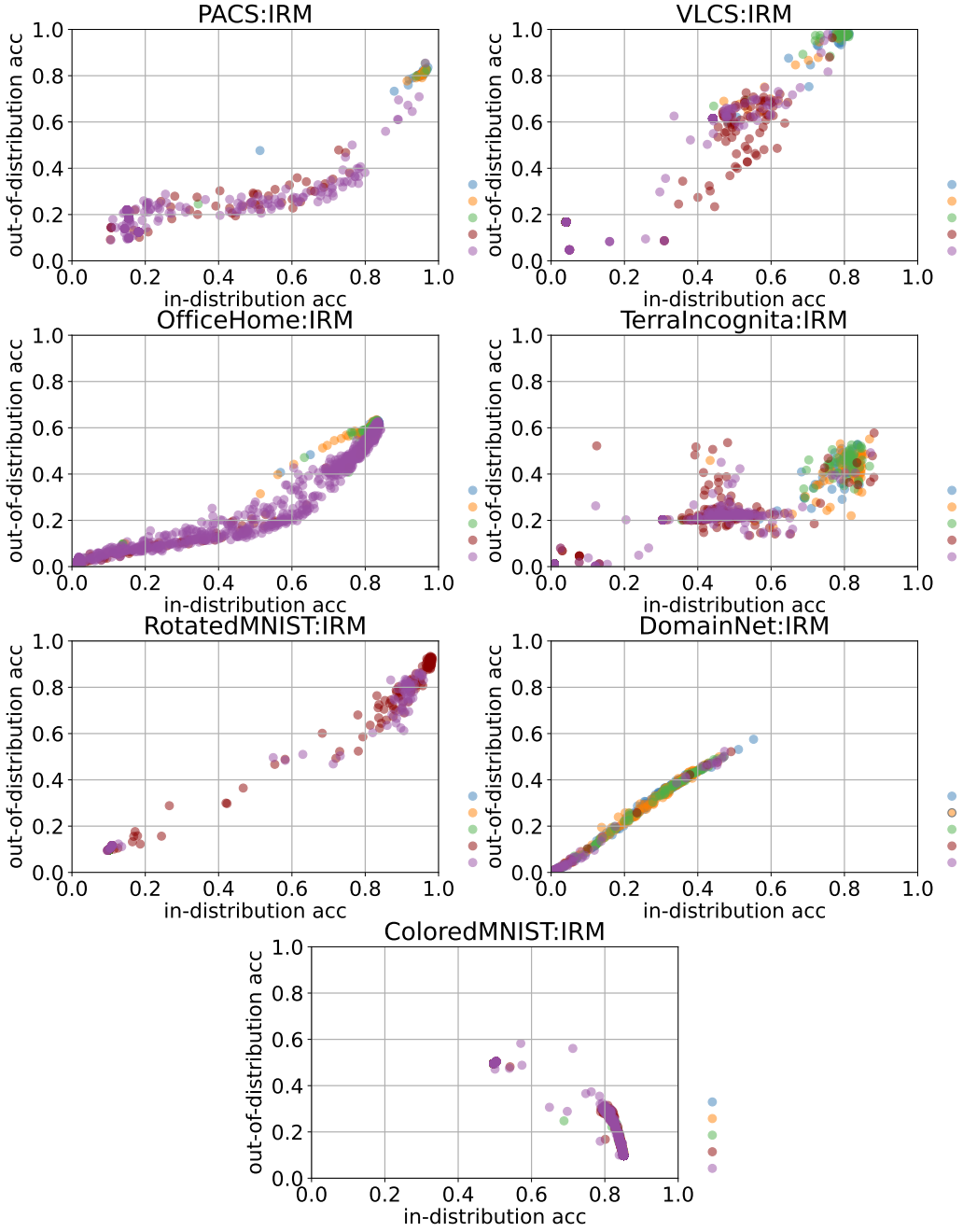


Figure 25. DomainBed: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of IRM across optimizers. The legend circles on the right side of each figure show, in order, VanillaSGD, Momentum SGD, Nesterov Momentum, RMProp, and Adam. The difference in each data point indicates the difference in hyperparameter configuration.

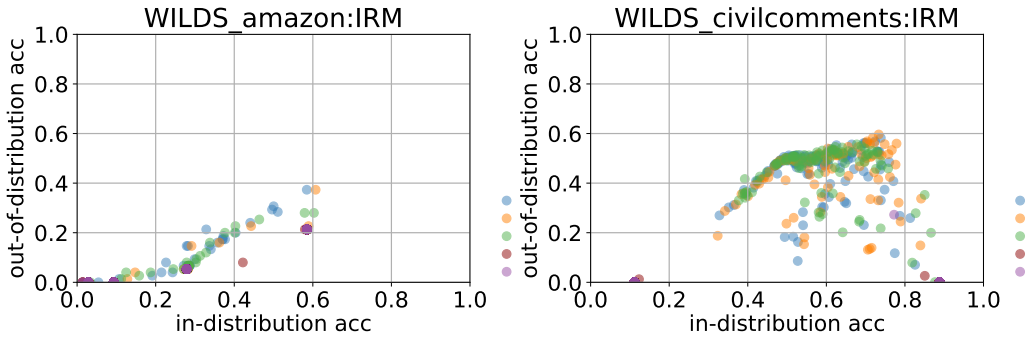


Figure 26. WILDS: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of IRM across optimizers. The legend circles on the right side of each figure show, in order, VanillaSGD, Momentum SGD, Nesterov Momentum, RMProp, and Adam. The difference in each data point indicates the difference in hyperparameter configuration.

I Additional Results of Experiments

In addition to the experimental results presented in the main paper, we provide a more in-depth analysis, which is shown in this chapter.

First of all, We show that the pattern of linear correlation of accuracy claimed by [Mil+21] does not necessarily occur even when the correlation patterns we observed, such as diminishing returns, are probit transformed. The results of the probit transform of the scatter plots shown in Appendix H.3 are shown, following the method of [Mil+21]. This may be due to the fact that our experimental setup uses a larger number of data sets and takes IRM and other factors into account.

In the second section, we present results comparing the performance improvement of OOD with a larger trial budget in the search for hyperparameters by Bayesian optimization.

Thirdly, We investigate why Adam shows better OOD performance than SGD in the negatively correlated case seen in Figure 7 for ColoredMNIST by considering the learning curve.

Finally, we examine the effectiveness of the early stopping for each optimizer to study if adaptive optimizers overfit because of their speed of convergence.

I.1 Probit Transformed Scatter Plot

As shown in Section 3.3, the work of [Mil+21] compares the accuracy of OOD with the accuracy of in-distribution by showing a plot on a probit scale. Their comparison shows that the correlation of accuracy is linear.

In this section, we convert the scatter plots identified in Section H.3 to probit scale and confirm that they do not necessarily show a linear correlation (Figure 27, 28, 29, 30, 30, and 31).

I.1.1 ERM

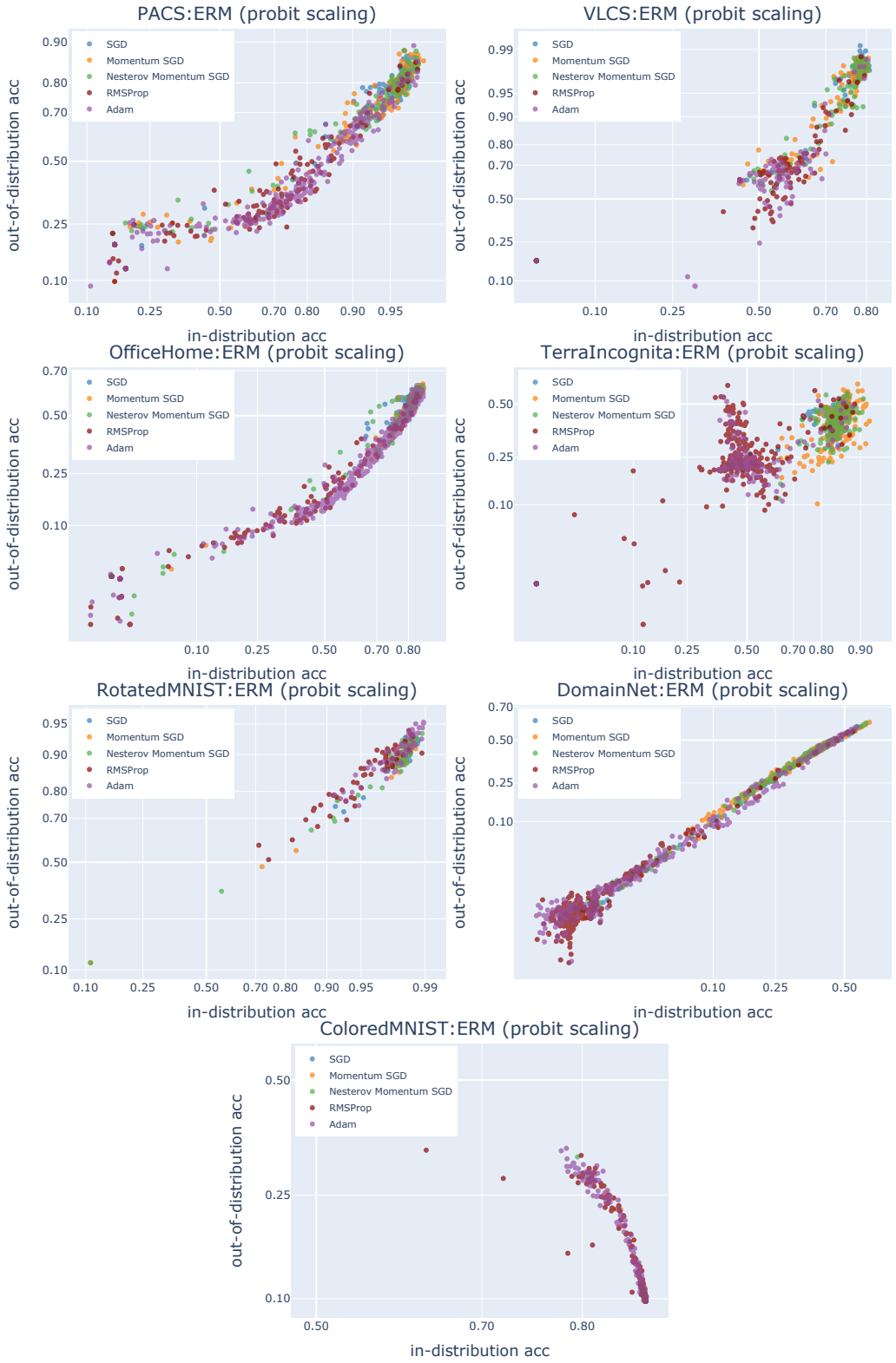


Figure 27. Probit Scale / DomainBed: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of ERM across optimizers. The difference in each data point indicates the difference in hyperparameter configuration.

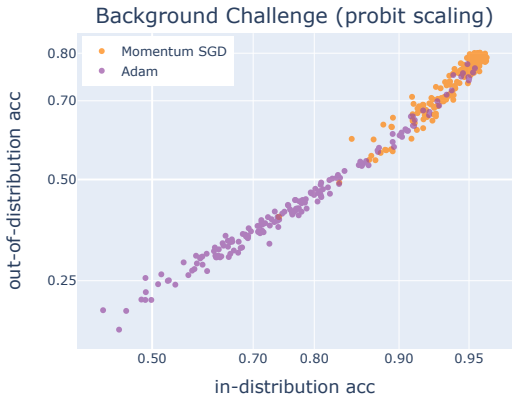


Figure 28. Probit Scale / Backgrounds Challenge: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of ERM across optimizers. The difference in each data point indicates the difference in hyperparameter configuration.

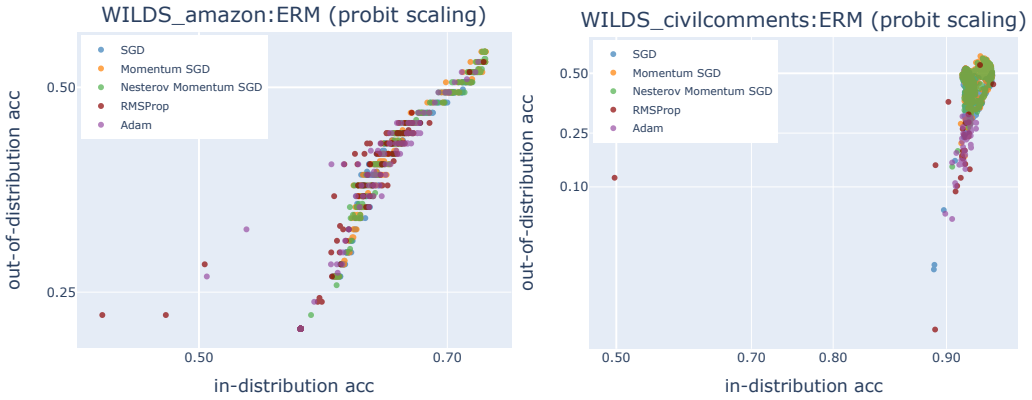


Figure 29. Probit Scale / WILDS: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of ERM across optimizers. The difference in each data point indicates the difference in hyperparameter configuration.

I.1.2 IRM

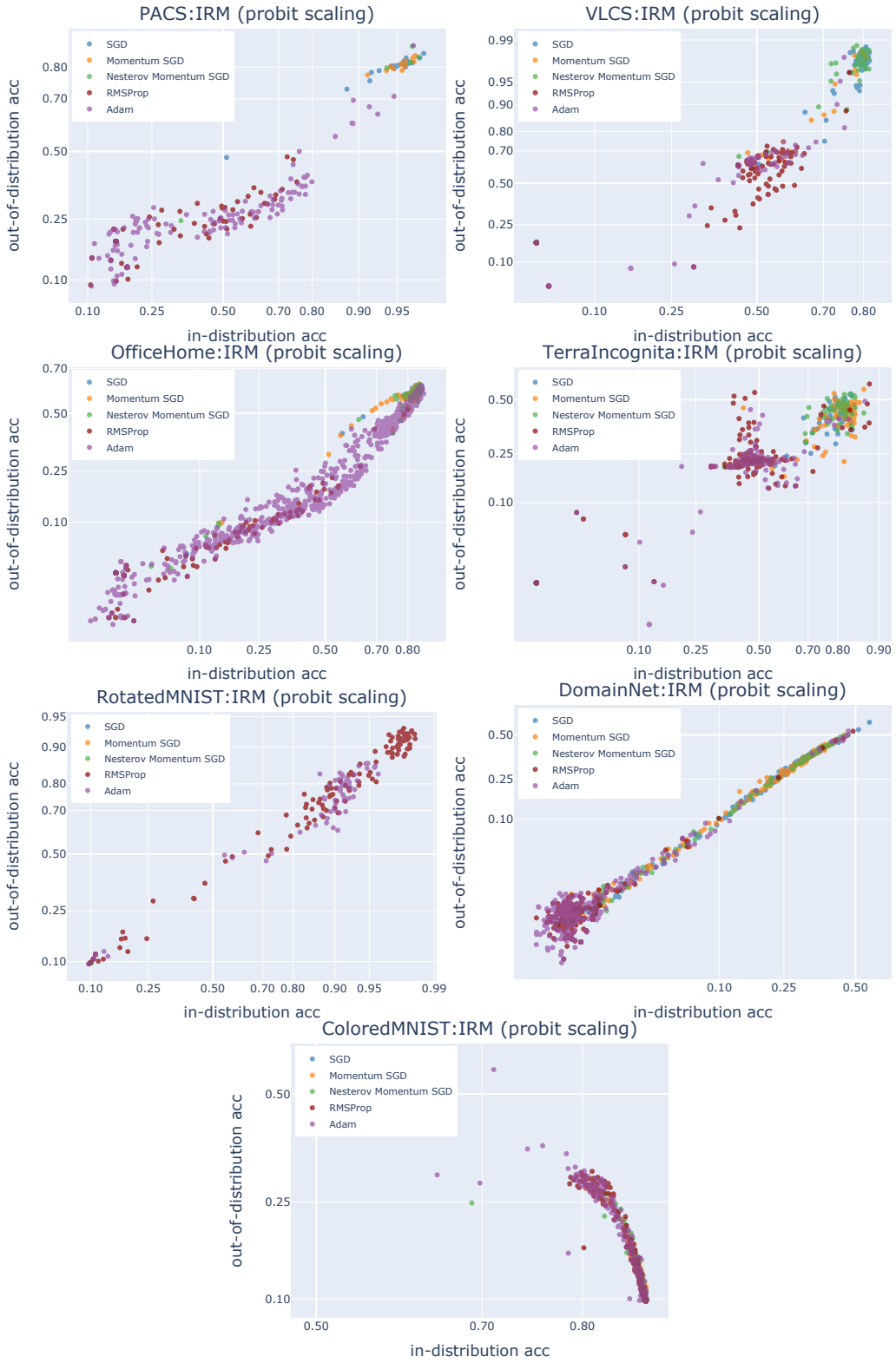


Figure 30. Probit Scale / DomainBed: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of IRM across optimizers. The difference in each data point indicates the difference in hyperparameter configuration.

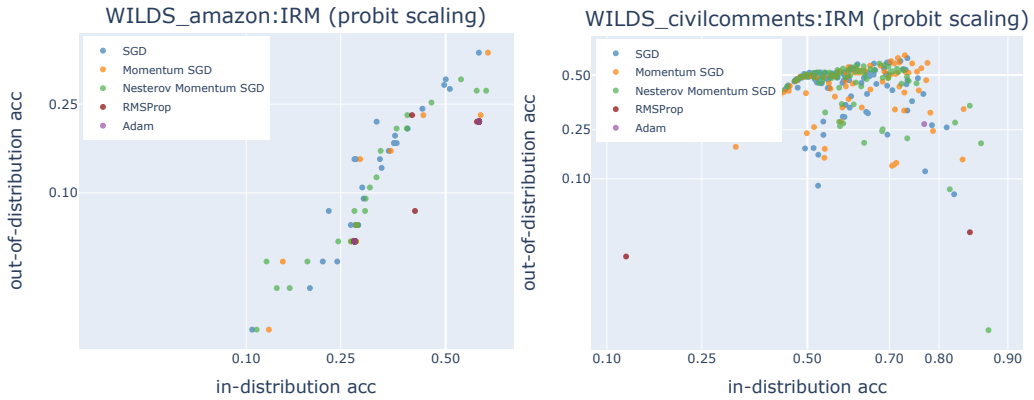


Figure 31. Probit Scale / WILDS: Comparison of the in-distribution accuracy and the out-of-distribution accuracy of IRM across optimizers. The difference in each data point indicates the difference in hyperparameter configuration.

I.2 Results of OOD Accuracy when horizontal axis is the trial budget

For practitioners, we show how much the trial budget for hyperparameter optimization affects OOD generalization when trained with ERM and IRM.

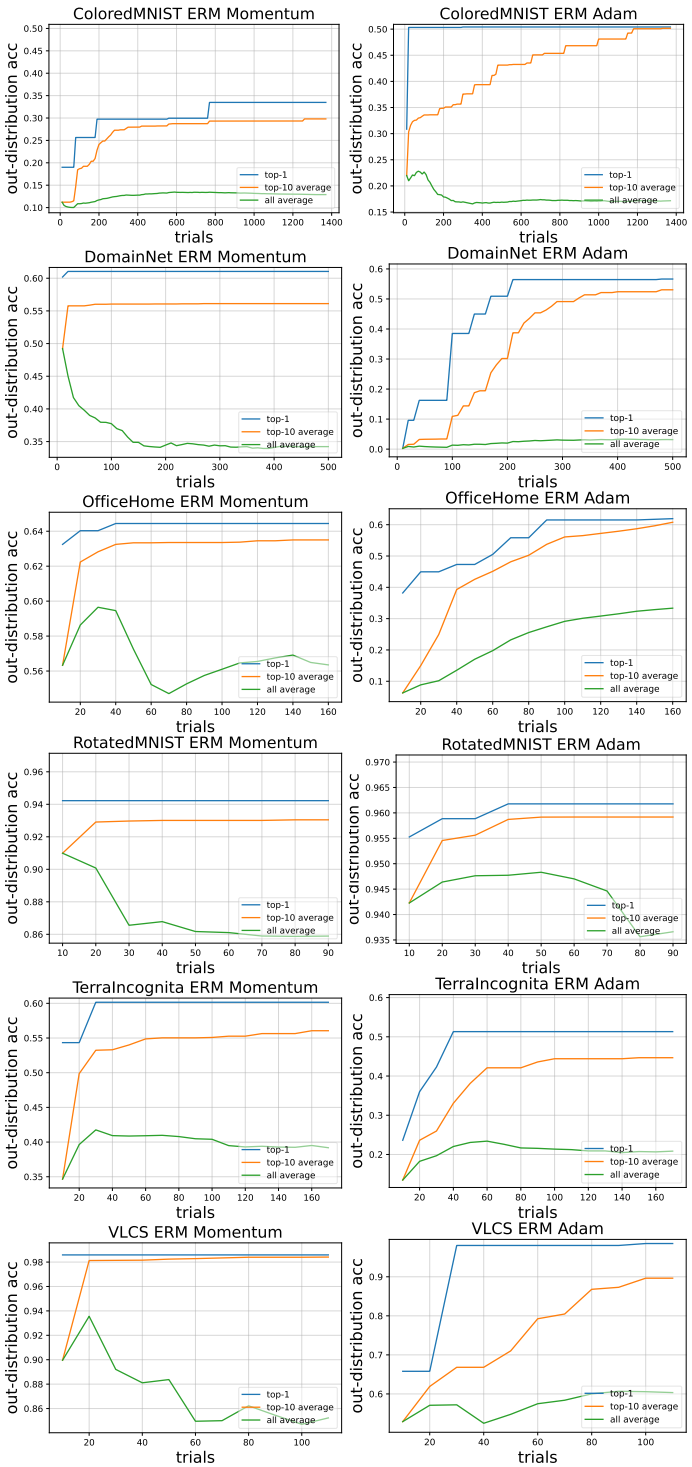


Figure 32. ERM Domainbed / OOD accuracy when horizontal axis is the trial budget

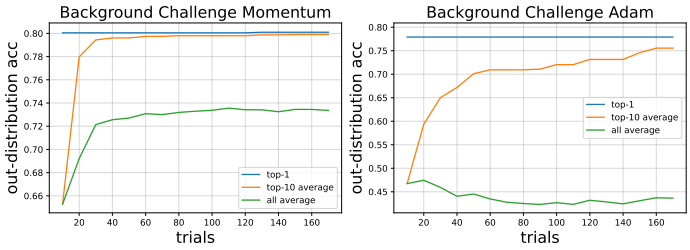


Figure 33. ERM Background Challenge / OOD accuracy when horizontal axis is the trial budget

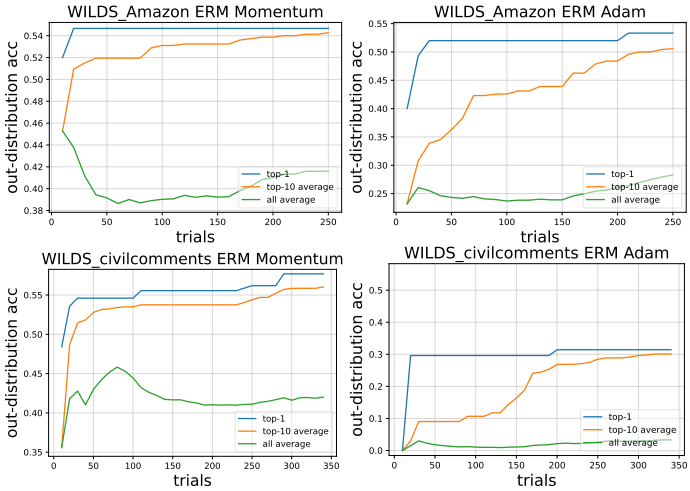
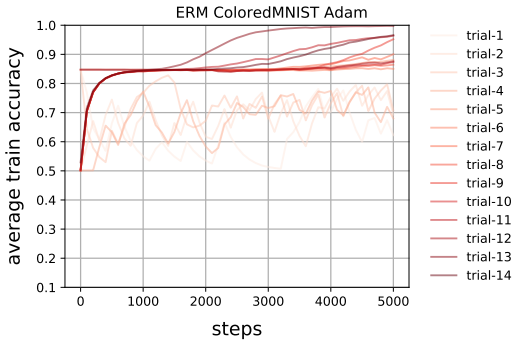


Figure 34. ERM WILDS / OOD accuracy when horizontal axis is the trial budget

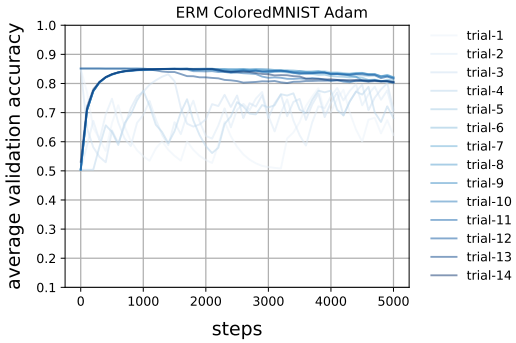
I.3 Learning Curve of ColoredMNIST

In Section G.1, we conjecture that the better performance of Adam in Colored MNIST classification may come from overfitting to training data. To confirm this hypothesis, we plot the averaged training accuracy, the averaged validation accuracy, and the averaged test accuracy throughout the training. We pick the top-14 results in terms of test accuracy and use them for the plot. We show the result in Figure 35.

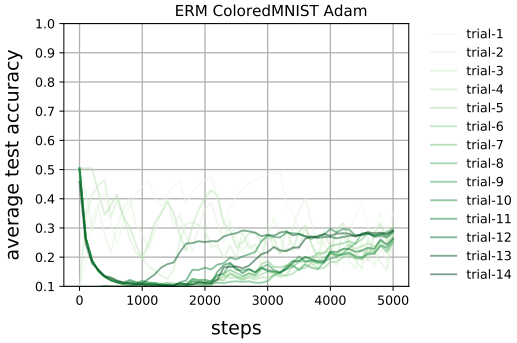
As is evident from Figure 35 (a) and Figure 35 (b), we observe that training accuracy increases while the validation accuracy keeps unchanged. This indicates that overfitting occurs in the training. However, Figure 35 (c) indicates that test accuracy gradually improves as the model is overfitting. On the other hand, SGD shows no such overfitting and OOD generalization improvement, as shown in Figure 36. Thus, we can empirically support our conjecture that Adam produces the better OOD generalization performance by overfitting training data.



(a) ERM: Averaged training accuracy of Adam on Colored MNIST throughout the training.

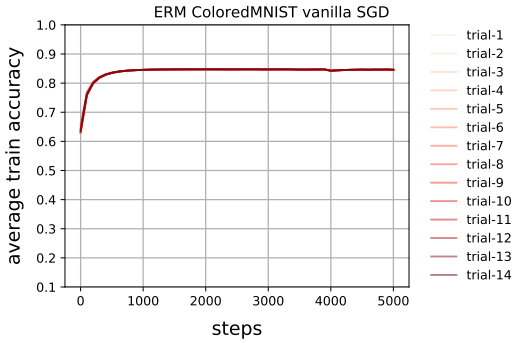


(b) ERM: Averaged validation accuracy of Adam on Colored MNIST throughout the training.

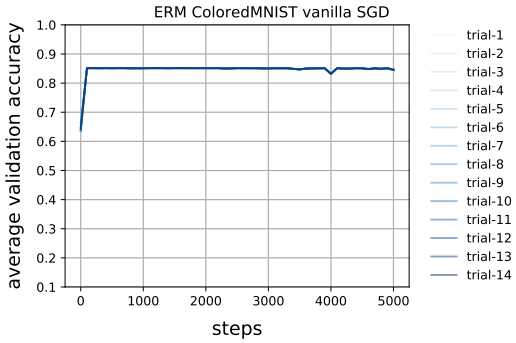


(c) ERM: Averaged test accuracy of Adam on Colored MNIST throughout the training.

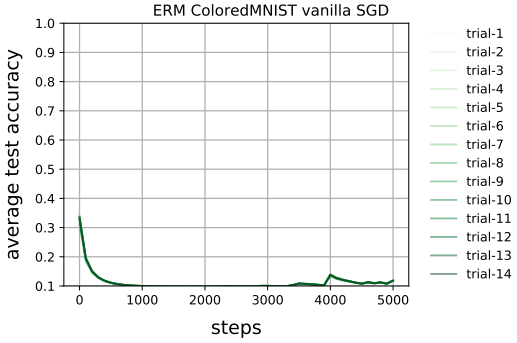
Figure 35. Adam: Comparison of averaged training accuracy, averaged validation accuracy, and averaged test accuracy throughout the training. We plot these values of check points whose train accuracy is in the top-14.



(a) ERM: Averaged training accuracy of SGD on Colored MNIST throughout the training.



(b) ERM: Averaged validation accuracy of SGD on Colored MNIST throughout the training.



(c) ERM: Averaged test accuracy of SGD on Colored MNIST throughout the training.

Figure 36. SGD: Comparison of averaged training accuracy, averaged validation accuracy, and averaged test accuracy of SGD throughout the training. We plot these values of checkpoints whose train accuracy is in the top-14.

I.4 Early Stopping

In Section G.1, figures 1 shows that adaptive optimizers tend to overfit to training domain. A possible reason is that the training speed of adaptive methods is faster than non-adaptive ones. That is, in the same steps budget, adaptive optimizers converge faster in effect. To validate if this is the case, we investigate whether early stopping improves the OOD generalization of adaptive optimizers.

In particular, we compute the difference between averaged accuracy at early stopping and at last epoch for test accuracy and validation accuracy, respectively:

$$\text{difference of accuracy} = \text{average accuracy at early stopping} - \text{average accuracy at last epoch}. \tag{15}$$

If the average accuracy at early stopping is larger, the difference is positive and vice versa.

Figures 37 to 42 are the results of this comparison for each dataset and algorithm. The y-axis is the difference of test accuracy and the x-axis is the difference of validation accuracy. The color indicates the epoch of early stopping. The darker color indicates that early stopping is conducted in relatively earlier epochs.

For PACS, both differences are positive and lighter colors are concentrated at points of small difference of the validation accuracy, as indicated in Figures 37 and 38. Therefore, we can conclude that when the validation accuracy deteriorates by further training, the test accuracy also gets worse. However, the effect of further training is less evident for Adam. In other words, early stopping does not influence Adam so much but keeps test accuracy from decreasing for SGD.

The result of VLCS shows a similar pattern as PACS (Figures 39 and 40). If anything, Further training after early stopping makes adaptive optimizes be likely to result in better test accuracy than SGD, though they have a large variance. With respect to SGD, additional training degrades the test accuracy as much as it degrades validation accuracy.

Office-Home shows similar results as PACS, as presented in Figures 41 and 42.

In summary, we find that early stopping does not influence adaptive optimizers. Therefore, we can conclude that adaptive optimizer overfits not because it trains faster than non-adaptive methods. The fact that early stopping does not affect the adaptive optimizer means that adjusting the number of epochs instead of a fixed number of epochs will not change the result. We have followed previous studies and experimented with a fixed number of epochs in the present study, and our results suggest that this does not have a serious impact on the comparison of adaptive and non-adaptive optimizers. Thus, we can see that using a fixed epoch number does not undermine the validity of our optimizer comparison experiment in this sense.

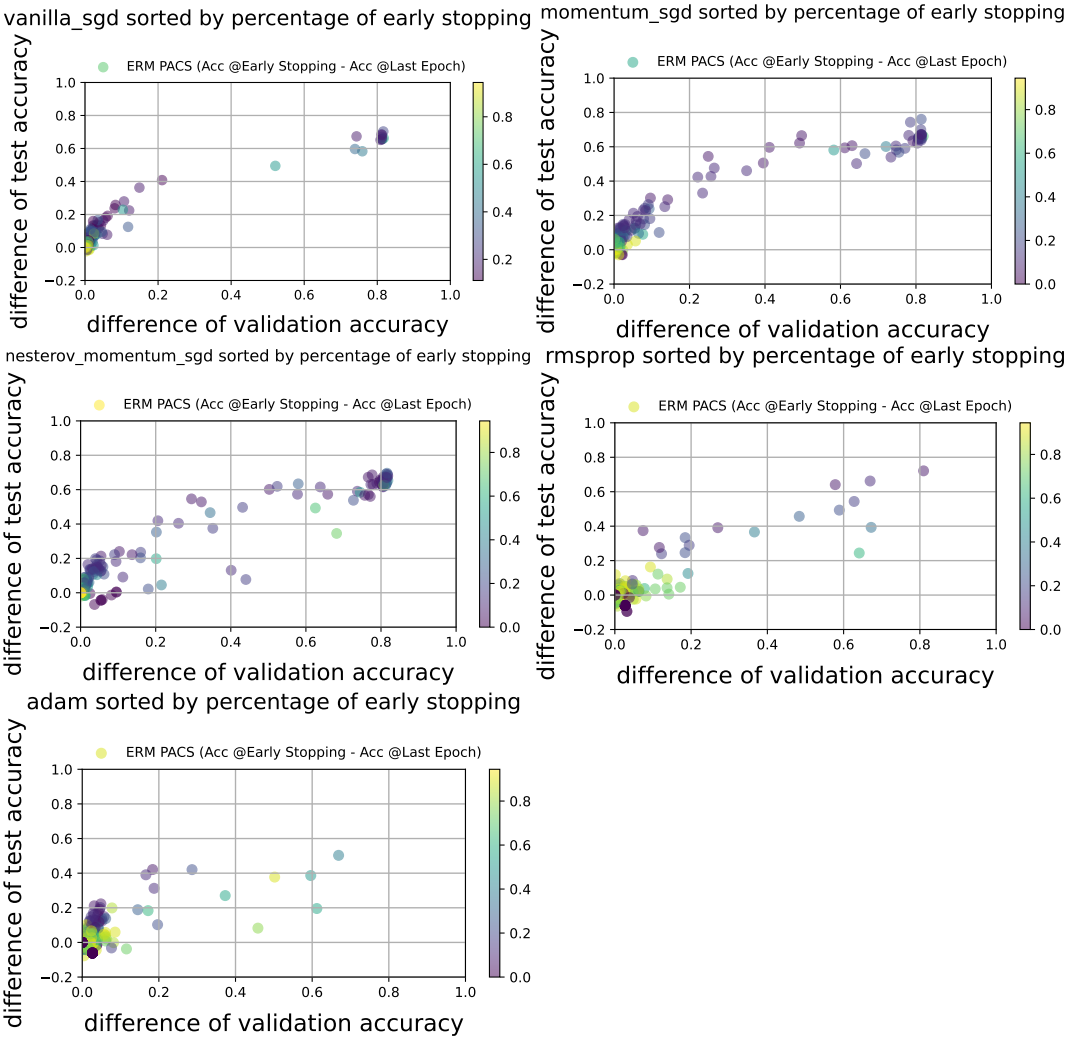


Figure 37. ERM/PACS: Difference between accuracy at early stopping and at last epoch. The y-axis is the difference of test accuracy and the x-axis is the difference of validation accuracy. The color indicates the epoch of early stopping. The darker color indicates that early stopping is conducted in relatively earlier epochs.

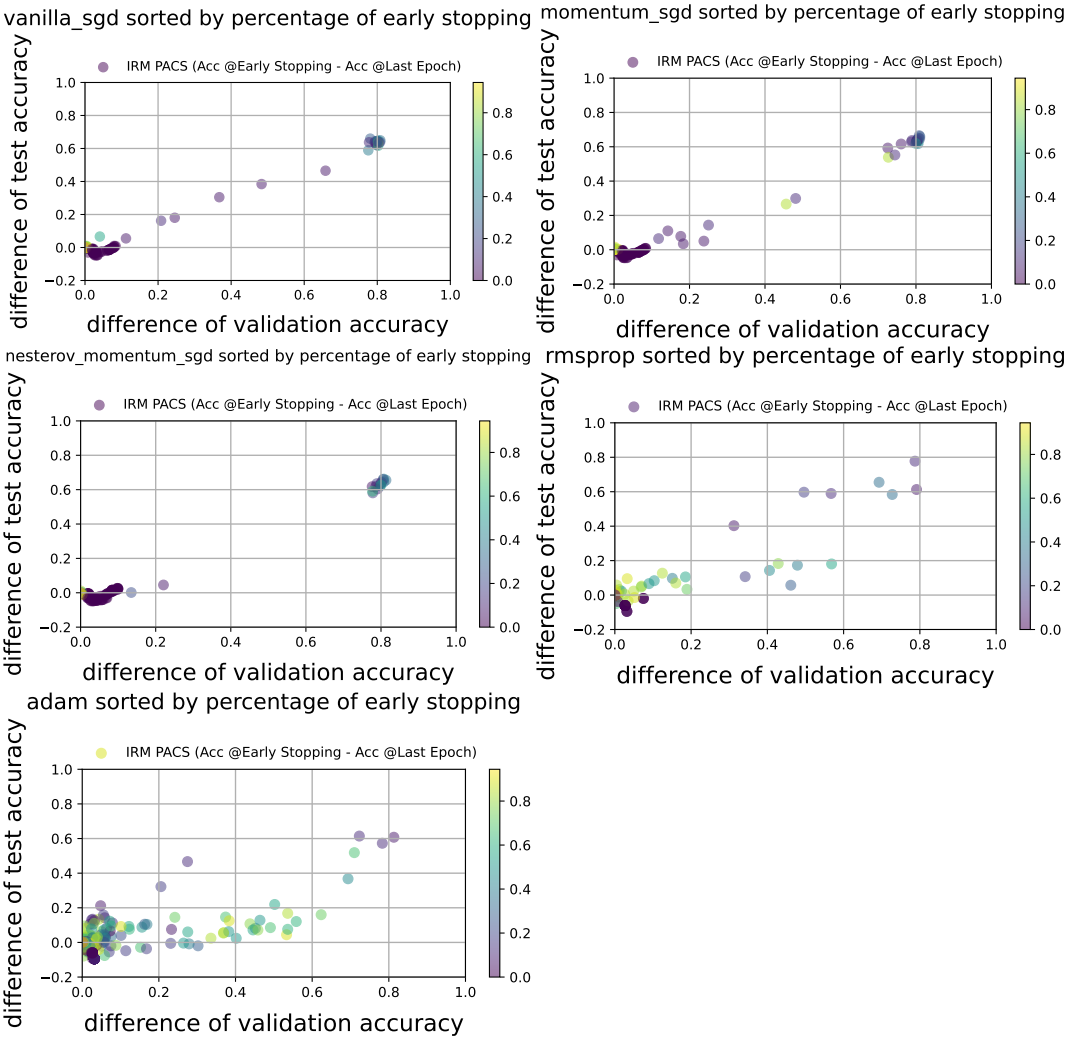


Figure 38. IRM/PACS: Difference between accuracy at early stopping and at last epoch. The y-axis is the difference of test accuracy and the x-axis is the difference of validation accuracy. The color indicates the epoch of early stopping. The darker color indicates that early stopping is conducted in relatively earlier epochs.

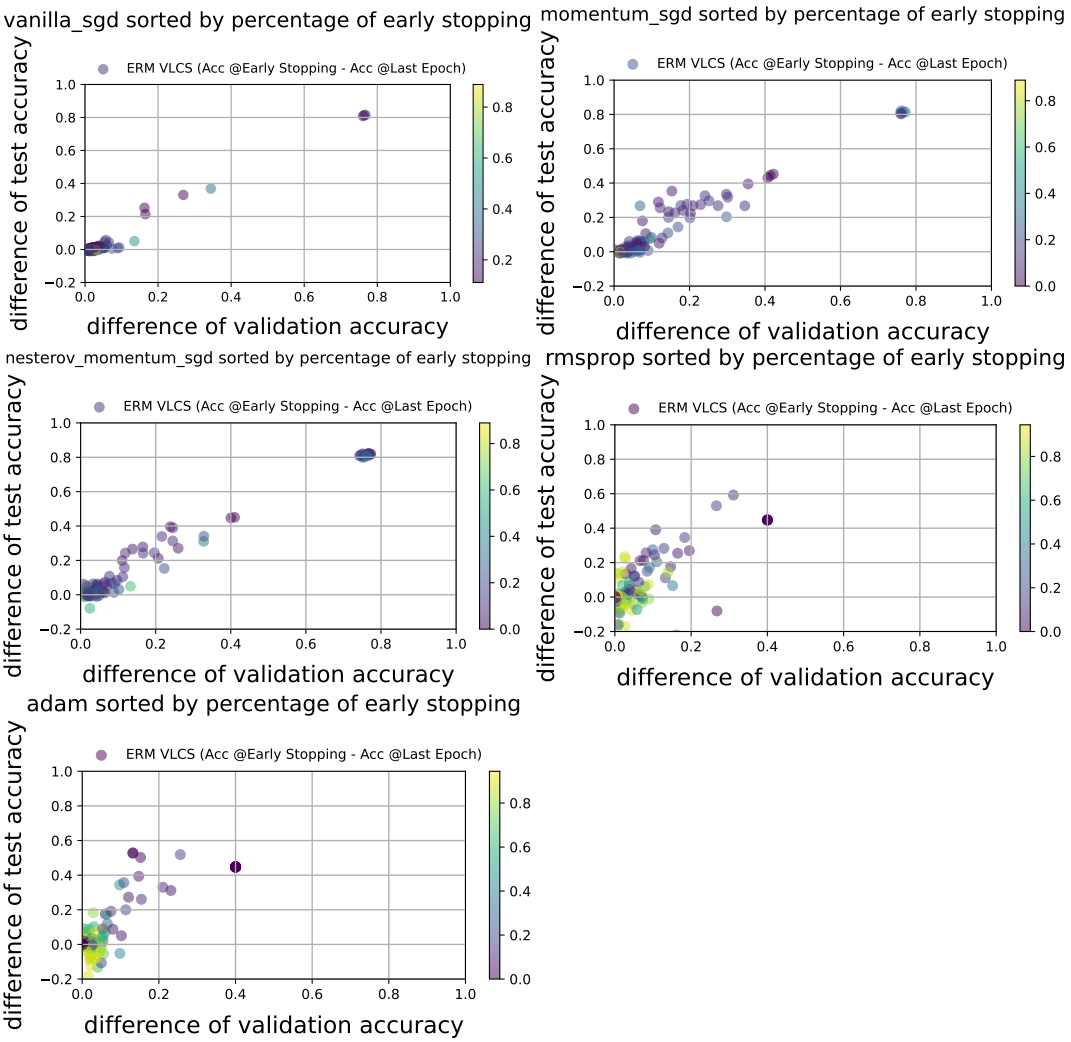


Figure 39. ERM/VLCS: Difference between accuracy at early stopping and at last epoch. The y-axis is the difference of test accuracy and the x-axis is the difference of validation accuracy. The color indicates the epoch of early stopping. The darker color indicates that early stopping is conducted in relatively earlier epochs.

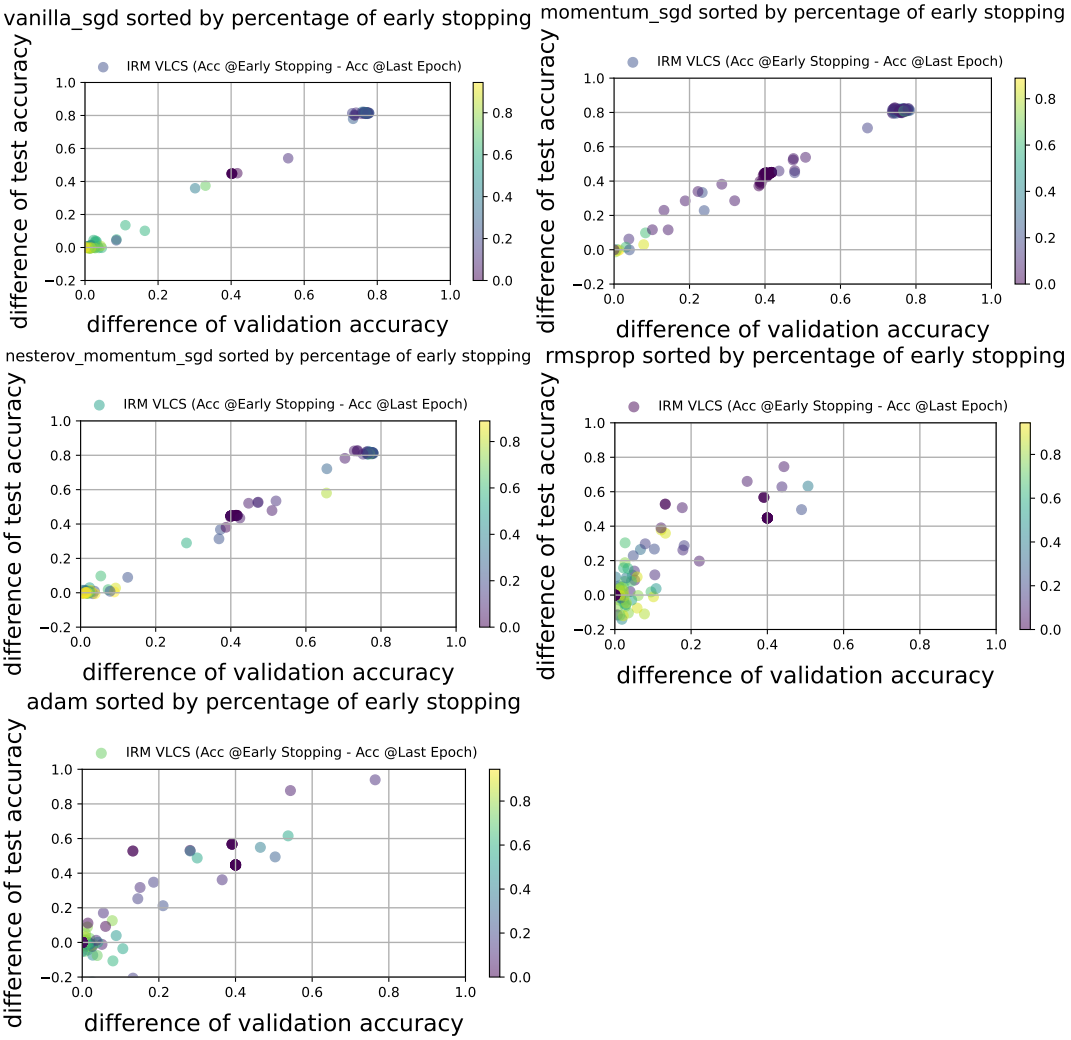


Figure 40. IRM/VLCS: Difference between accuracy at early stopping and at last epoch. The y-axis is the difference of test accuracy and the x-axis is the difference of validation accuracy. The color indicates the epoch of early stopping. The darker color indicates that early stopping is conducted in relatively earlier epochs.

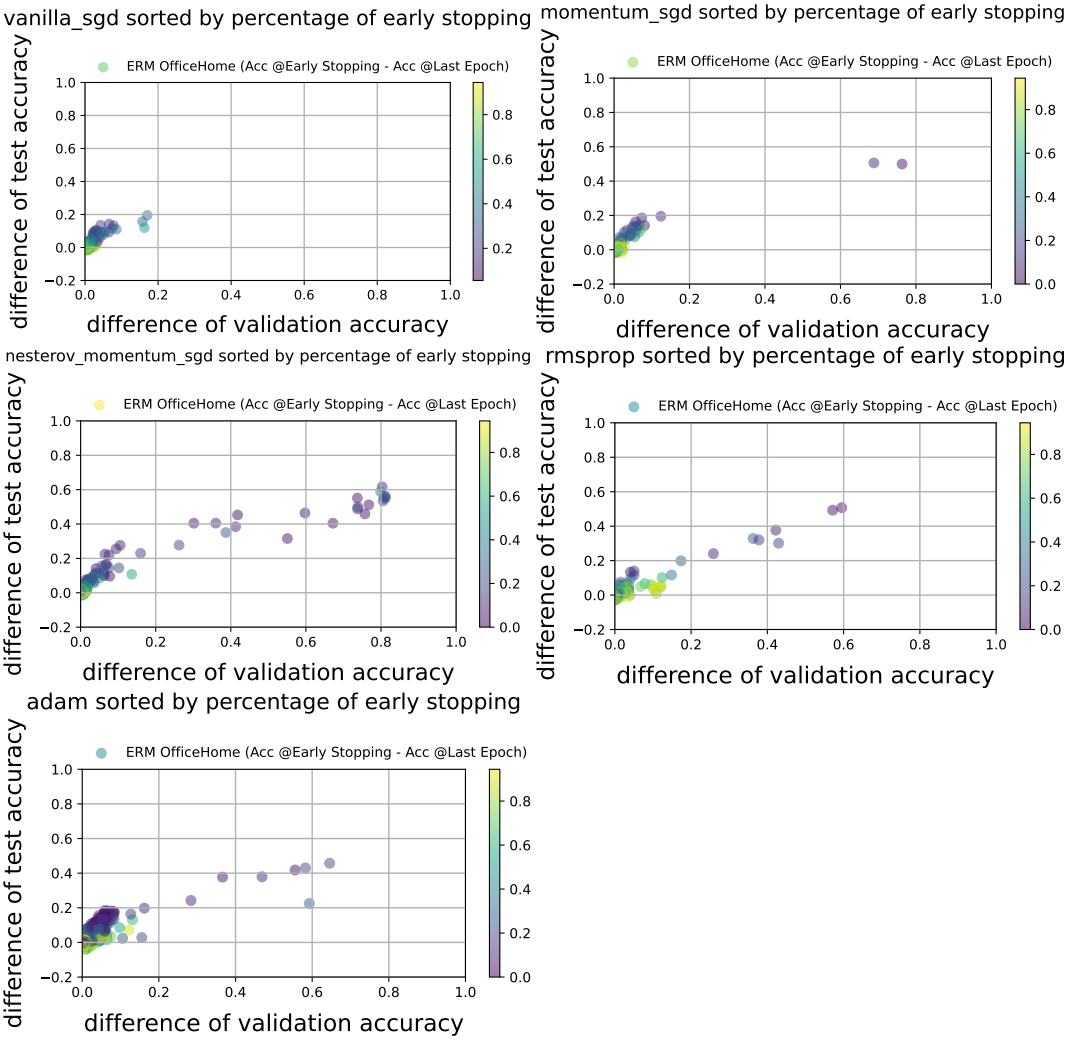


Figure 41. ERM/Office-Home: Difference between accuracy at early stopping and at last epoch. The y-axis is the difference of test accuracy and the x-axis is the difference of validation accuracy. The color indicates the epoch of early stopping. The darker color indicates that early stopping is conducted in relatively earlier epochs.

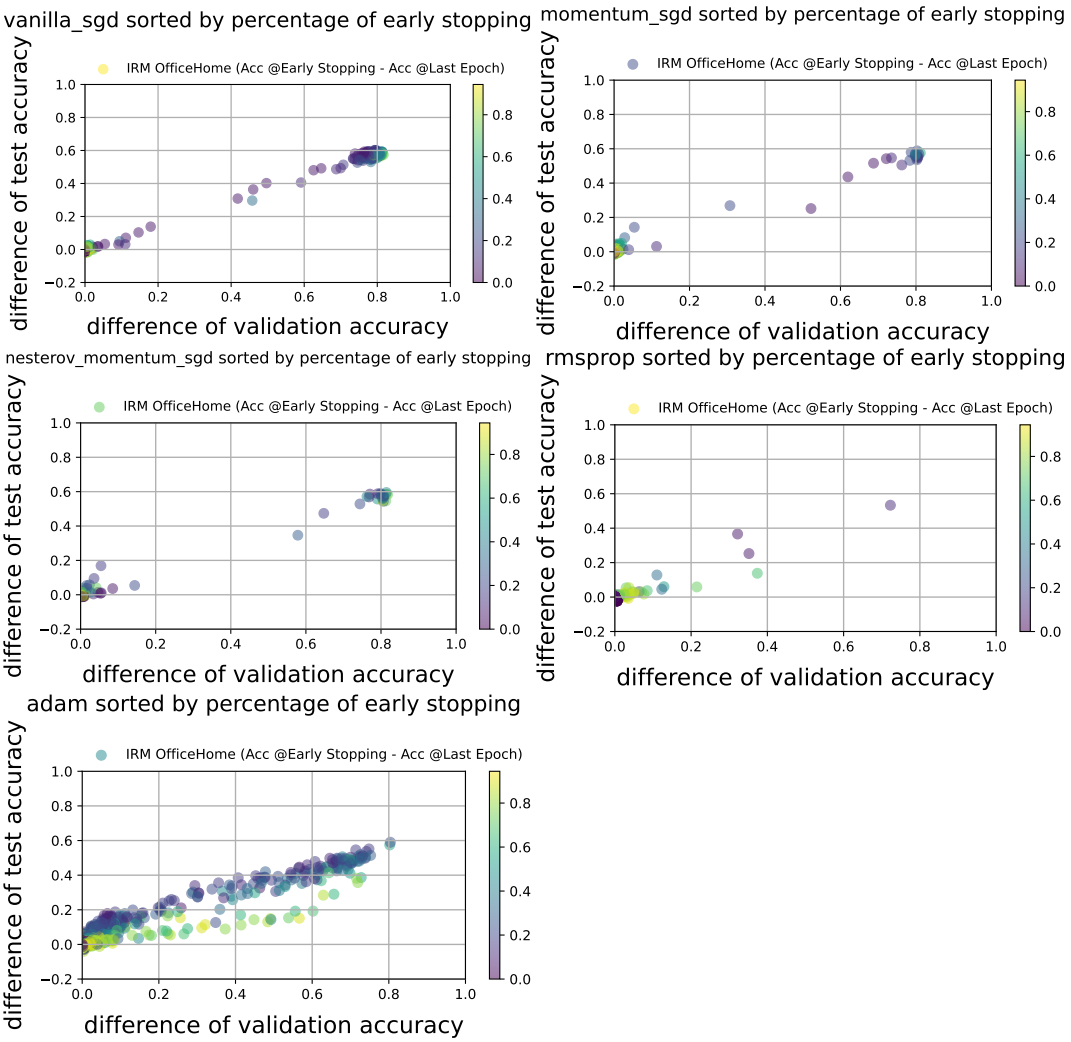


Figure 42. IRM/Office-Home: Difference between accuracy at early stopping and at last epoch. The y-axis is the difference of test accuracy and the x-axis is the difference of validation accuracy. The color indicates the epoch of early stopping. The darker color indicates that early stopping is conducted in relatively earlier epochs.

However, we find different results from Colored MNIST. As shown in Figures 43 and 44, we can observe that the difference of validation accuracy is positive and that of test accuracy is negative. That is further training after early stopping decreases validation accuracy but increases test accuracy on Colored MNIST, while there is no difference for SGD. This is consistent with the result of Appendix I.3.

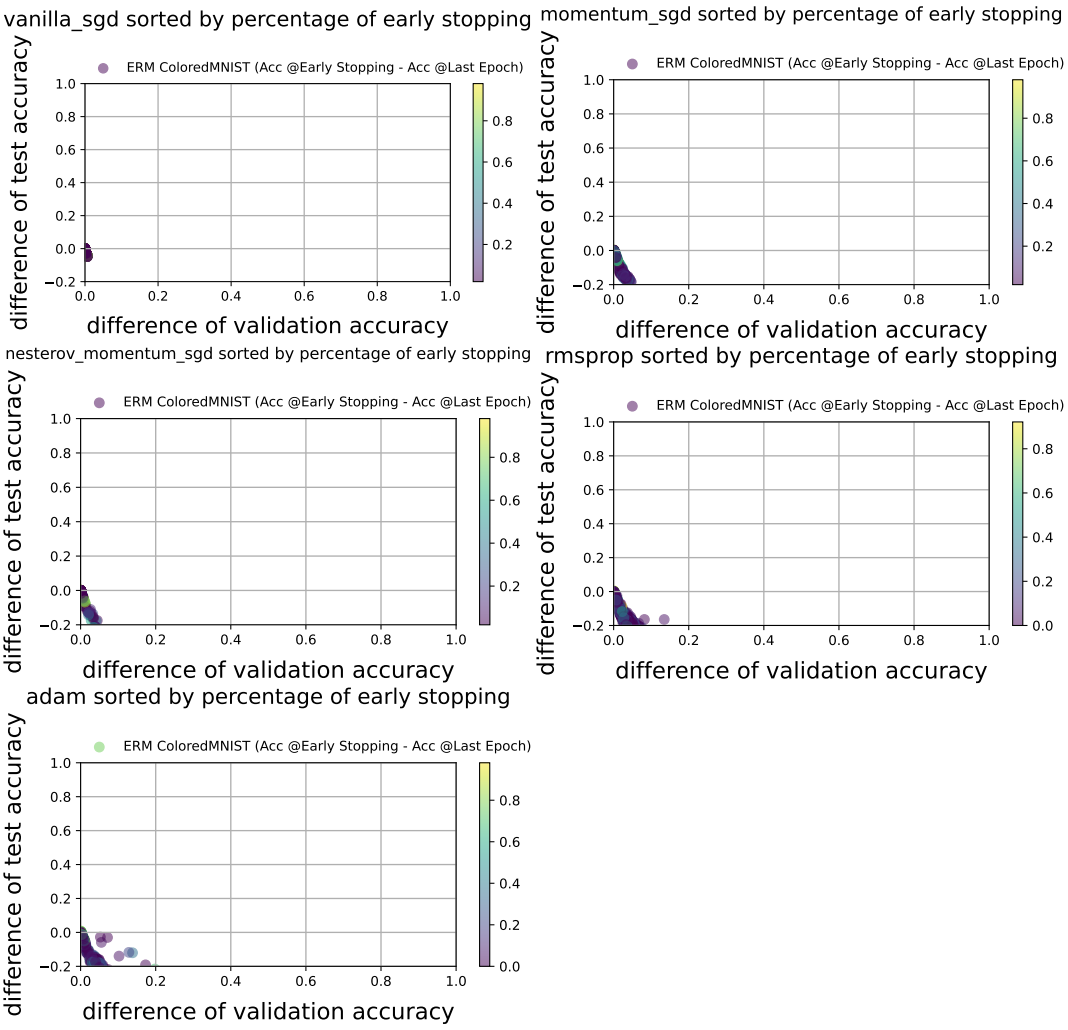


Figure 43. ERM/Colored MNIST: Difference between accuracy at early stopping and at last epoch. The y-axis is the difference of test accuracy and the x-axis is the difference of validation accuracy. The color indicates the epoch of early stopping. The darker color indicates that early stopping is conducted in relatively earlier epochs.

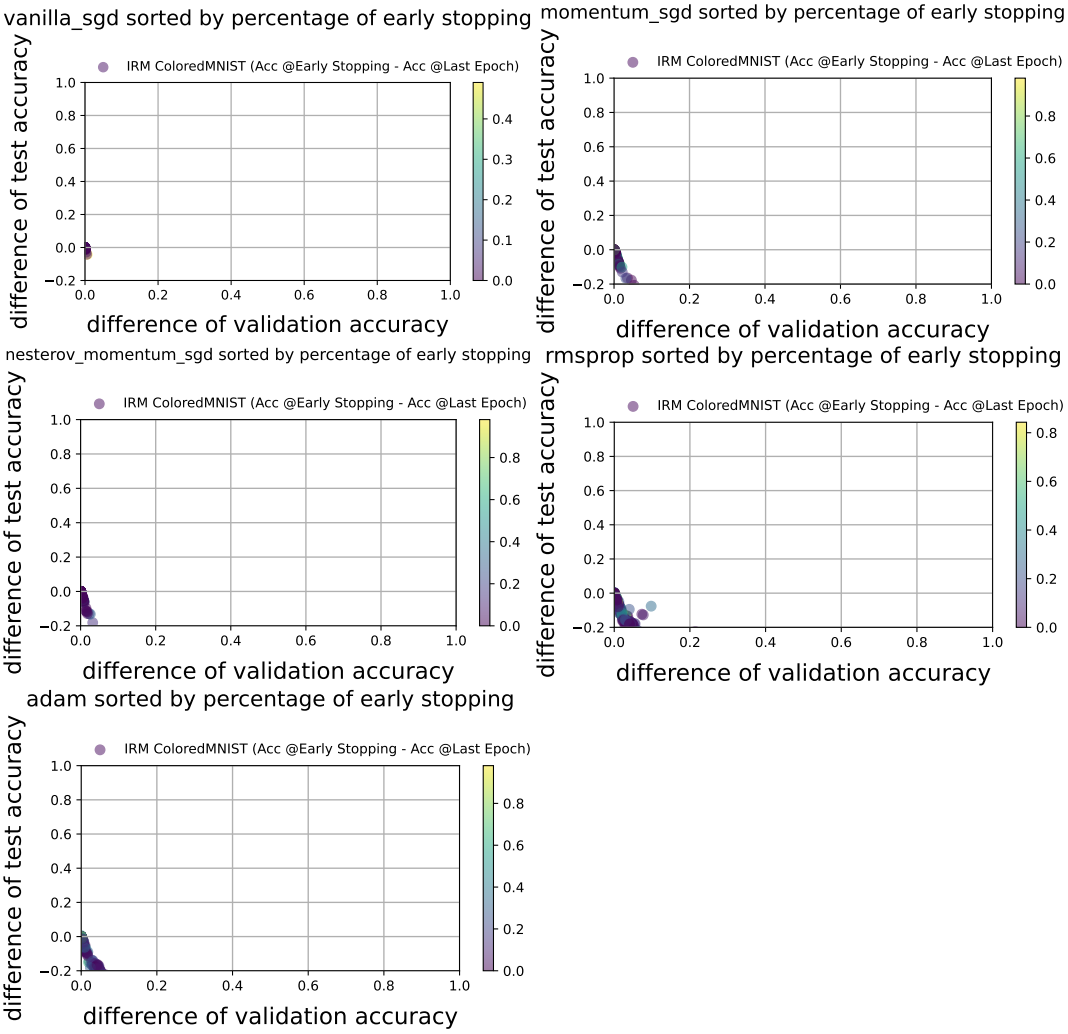


Figure 44. IRM/Colored MNIST: Difference between accuracy at early stopping and at last epoch. The y-axis is the difference of test accuracy and x-axis is the difference of validation accuracy. The color indicates the epoch of early stopping. The darker color indicates that early stopping is conducted in relatively earlier epochs.

J Soundness Check of Our Experiments

J.1 Histgram of Hyperparameters

We have shown the histgram of the hyperparameters to be used for training just for reference. In particular, we display a result for learning rate of Momentum SGD and Adam for the each dataset. We observe that we could sample hyperparameters from a reasonably wide range.

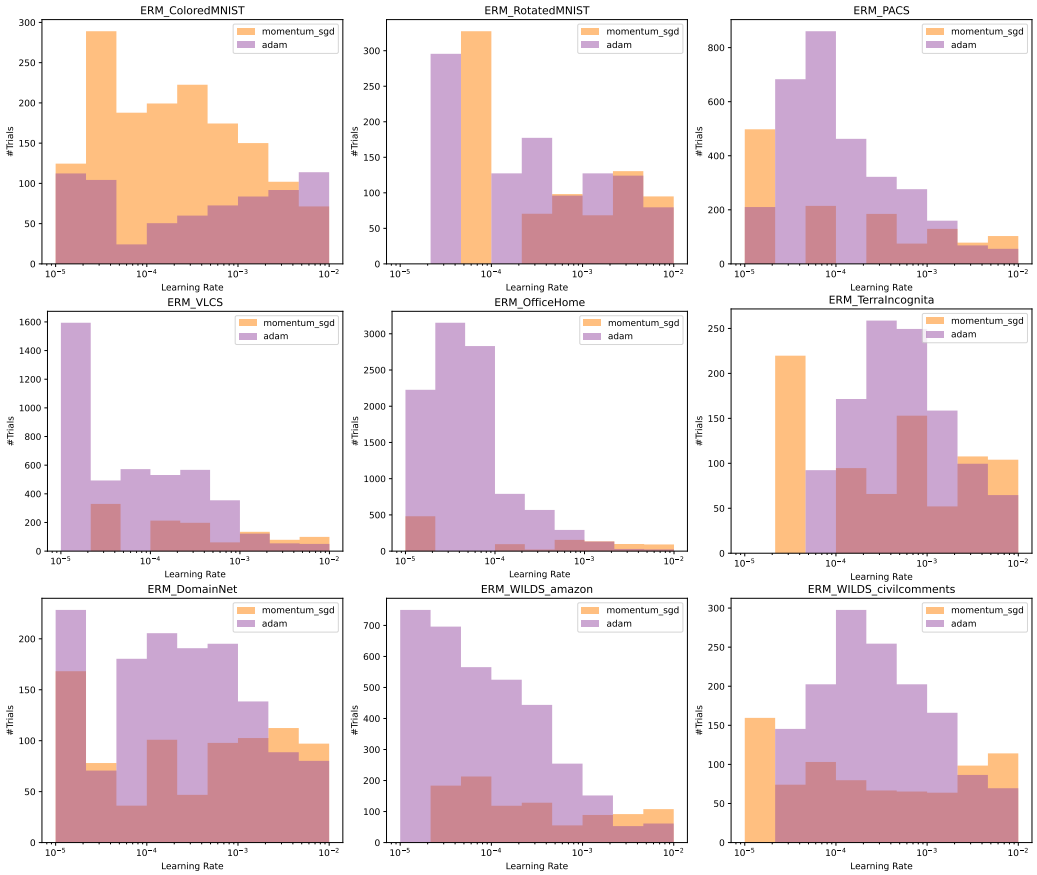


Figure 45. Histogram of explored hyperparameters (learning rate) in training of DomainBed, WILDS dataset in ERM. Momentum SGD and Adam results are included. Although uniform distribution is used as the prior distribution for hyperparameter optimization, the histogram results do not match the uniform distribution because Bayesian optimization is used.

J.2 Best OOD Performance Comparison against with Existing Benckmark

In order to confirm the soundness of our experiments, we compared our results with existing benchmarks to see how well they actually performed, in addition to the hyperparameter search ranges in the previous section. In particular, we compared our experimental results with those of DomainBed [GL21], an existing benchmark that uses Adam.

Dataset	OOD Domain	Existing Benchmark Results(Adam)	Our Results(Adam)
ColoredMNIST	0.9	10.0±0.1%	73.92%
RotatedMNIST	0	95.9±0.2%	96.40%
VLCS	C	97.8±0.0%	99.36%
PACS	A	83.9±1.6%	89.30%
OfficeHome	A	62.3±0.5%	63.12%
TerraIncognita	L100	47.5±0.2%	61.35%
DomainNet	clipart	56.0±1.1%	58.48%

Table 10: Comparison of our experimental results with the benchmark results reported in DomainBed [GL21]

J.3 The Number of Trials for Hyperparameter Search

For reference, we show the table of the number of trials for each dataset/optimizer combination on ERM in Table 11.

Dataset	#Trials (Momentum SGD)	#Trials (Adam)
Background Challenge	347	567
PACS	174	412
VLCS	132	324
OfficeHome	173	699
TerraIncognita	183	202
DomainNet	515	1137
ColoredMNIST	342	315
RotatedMNIST	156	341
WILDS_Civilcomments	543	554
WILDS_Amazon	438	466

Table 11: Number of Trials