ERM++: An Improved Baseline for Domain Generalization

Piotr Teterwak [1] **Kuniaki Saito** [1] **Theodoros Tsiligkaridis** [2] **Kate Saenko** [1 3] **Bryan A. Plummer** [1]

Abstract

Multi-source Domain Generalization (DG) measures a classifier's ability to generalize to new distributions of data it was not trained on, given several training domains. While several multi-source DG methods have been proposed, they incur additional complexity during training by using domain labels. Recent work has shown that a well-tuned Empirical Risk Minimization (ERM) training procedure, that is simply minimizing the empirical risk on the source domains, can outperform most existing DG methods. We identify several key candidate techniques to further improve ERM performance, such as better utilization of training data, model parameter selection, and weight-space regularization. We call the resulting method ERM++, and show it significantly improves the performance of DG on five multi-source datasets by over 5% compared to standard ERM, and beats state-of-the-art despite being less computationally expensive. We hope that ERM++ becomes a default baseline for DG. Code is released at https://github.com/piotr-teterwak/erm_plusplus.

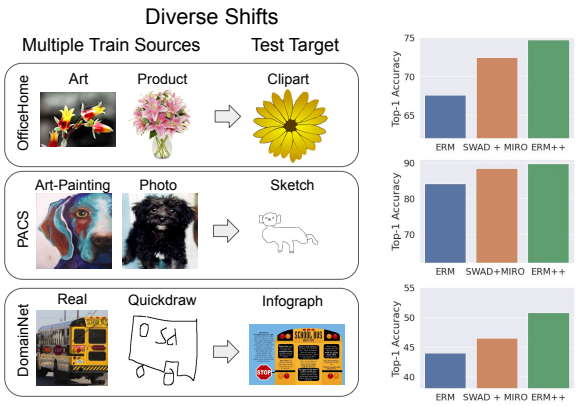

Figure 1. **ERM++:** We tackle the task of Multi-Source Domain Generalization, where a model is trained on several source domains and evaluated on a different target domain. We do this by improving the classic, and already strong, ERM (Gulrajani & Lopez-Paz, 2020) algorithm with known methodologies. We verify our method on a diverse set of domain shifts, and show that it improves over the best reported numbers in the literature.

1. Introduction

Domain Generalization (DG) is a crucial problem in the field of machine learning, as it addresses the challenge of building models that perform well on unseen (target) data distributions, without using target data to update the model (Blanchard et al., 2011; Muandet et al., 2013; Zhou et al., 2021). This is important in many real-world applications, where the distribution of data may vary between settings, and it is not always feasible to collect and label a large amount of data for each new domain. Similarly, it is not always known a-priori how the distribution on which the model is deployed differs from the training distribution. In multi-source domain generalization, each training sample is labelled as being part of one of several domains. Many advanced methods leverage domain membership explicitly. For example, DANN (Ganin et al., 2016) uses an adversarial loss to match feature distributions across source domains. Adaptive Risk Minimization (Zhang et al., 2021) meta-learns parameters which adapt a model to newly seen distribution shift. Yet, recently DomainBed (Gulrajani & Lopez-Paz, 2020) holistically evaluated methods and found that ERM (Empirical Risk Minimization), outperforms most prior work for DG in a setting where hyper-parameters are tuned. This is all the more impressive since ERM only leverages domain labels in a very weak way; by oversampling minority domains to balance domain sizes in the training data. Advanced techniques do not beat ERM (Gulrajani & Lopez-Paz, 2020) despite strong inductive biases and additional complexities (and hyper-parameters to tune).

Our goal is to revisit the framework used to benchmark multisource domain generalization problems to ensure that we maximize the performance of baseline methods. As illustrated in Figure 1, our new baseline, ERM++, outperforms

*Equal contribution [1]Boston University [2]MIT Lincoln Laboratory [3]Meta AI. Correspondence to: Piotr Teterwak <piotrt@bu.edu>.

Published at the ICML 2023 Workshop on Spurious Correlations, Invariance, and Stability. Honolulu, Hawaii, USA.

the state-of-the-art without the need for domain labels, architecture changes or complex training strategies. Instead, we critically evaluate the components of the training pipeline.

2. Revisiting training procedures to create ERM++ for Domain Generalization

We study the problem of Multi-Source Domain Generalization for classification. We train a model on training data consisting of multiple domains and evaluate it on data from unseen domains. More formally, let us consider training domains $d \in \{d_1, ..., d_n\}$. A training dataset is constructed using all sample, label pairs in all training domains $D = \{(X^{d_1}, Y^{d_1})...(X^{d_n}, Y^{d_n})\}$. After training classifier f on D, it is tested on a held-out testing domain d_{test}. As stated in previous sections, approaches utilizing invariance of the domain or regularization of features can complicate the training. Instead we perform simple empirical risk minimization (ERM), formalized as minimizing the average loss over all samples $\frac{1}{n}\sum_{i \in D} \ell(x_i, y_i)$, and shown to be successful on diverse tasks (Rame et al., 2022).

Our goal is to investigate the general training components that go into creating an ERM model to help ensure we have maximized its performance. These components include how to effectively use the source data (Section 2.1), considerations when selecting and using pretrained weights (Section 2.2), and weight-space regularization methods that help prevent overfitting to the source domains (Section 2.3). We refer to our new stronger baseline as ERM++.

2.1. Improved Data Utilization

A key component of training any neural network is utilizing the (often limited) training data effectively. A common practice in the domain generalization literature is to split source datasets into (often 80%/20%) train/validation sets under a fixed number of iterations for each dataset (*e.g.*, (Gulrajani & Lopez-Paz, 2020; Cha et al., 2021; Rame et al., 2022; Arpit et al., 2021)). The validation data is used to set hyperparameters and perform checkpoint (no. training steps) selection. This approach has two major drawbacks. First, by creating a separate validation set we are sacrificing a significant portion of our labeled data. Second, by training under a fixed (relatively small) number of iterations we ignore the varying convergence rates of different models, which may result in a model underperforming its true ability.

To counteract these issues, when setting hyperparameters in the first stage, we include a new parameter ϕ that sets the training length (*Early Stopping*, Table 2 Experiment 6). Once we have set all the hyperparameters (including ϕ), we train our deployment model using the full dataset as noted earlier, selecting the final checkpoint as our model. More concretely, we continue training until we no longer observe significant performance gains, which we refer to as Long Training (*LT*, Table 2, Experiment 4). This uses training labels more efficiently by training on the Full-Dataset (*FD*, Table 2 Experiment 3). The full ERM++ baseline enables *Early Stopping, LT* and *Full Data*.

2.2. Leveraging Pretrained Model Weights

Most domain generalization methods do not train a model from scratch, but rather transfer the weights of an existing model, typically pretrained on ImageNet (Deng et al., 2009). There are three decisions that we explore further: selecting what model weights to transfer, determining what weights to fine-tune or keep frozen, and how to initialize new weights .

Model Weight Selection: Recent work has shown that better ImageNet models have better domain generalization properties for both single-source and multi-source DG (Kim et al., 2022; Angarano et al., 2022). However, this has been explored in the context of varying model size. Therefore, performance gains can be either from a.) improved pre-training dataset (upstream) performance resulting in improved DG or b.) larger models resulting in improved DG performance, regardless of upstream performance. We explore the effect of different initializations for the same model architecture, specifically a ResNet-50 (He et al., 2016). We test standard TorchVision model weights, AugMix-trained weights (Hendrycks et al., 2020), ResNet A1 weights (Wightman et al., 2021), and finally a distilled ensemble ((Shen & Savvides, 2020)). Results are shown in Table 4 and more details are present in the Appendix. We find Augmix weights the strongest, and call AugMix inititialization Strong Init. (Table 2, Experiment 7). The full ERM++ baseline uses *Strong Init.*

Finetuning or freezing model weights: It has been shown that what parameters to update during fine-tuning a pretrained model, and when, can have substantial effects on downstream performance. Surgical fine-tuning (Lee et al., 2022) shows that only updating some blocks results in improved performance, but that different datasets require the unfreezing of different blocks. Most domain generalization methods will fine-tune most layer weights, with the exception of BatchNorm parameters, which are sometimes kept frozen. We compare the effect freezing or finetuning the BatchNorm parameters on performance, and refer to unfreezing them as *UBN* (Table 2, Experiment 9). The full ERM++ baseline uses *UBN*.

Initializaing new layer weights: Initializing new layer weights is typically accomplished by giving the new layer weights a random initialization and then training them on the target datasets. However, a recurring observation made by many researchers over the years is that your model may suffer from catastrophic forgetting of the pre-trained features due to the noisy gradients from the newly initialized

	OfficeHome	PACS	DomainNet	TerraIncognita	VLCS	Avg.
ERM (Vapnik, 1999)	67.6±0.2	84.2±0.1	44.0±0.1	47.8±0.6	77.3±0.1	64.2
CORAL (Sun & Saenko, 2016)	68.7±0.3	86.2±0.3	41.5±0.1	47.6±1.0	78.8±0.6	64.5
ERM + MIRO (Cha et al., 2022)	70.5±0.4	85.4±0.4	44.3±0.2	50.4±1.1	79.0±0.0	65.9
ERM + SWAD (Cha et al., 2021)	70.6±0.2	88.1±0.1	46.5±0.1	50.0±0.3	79.1±0.1	66.9
CORAL + SWAD (Sun & Saenko, 2016)	71.3±0.1	88.3±0.1	46.8±0.0	51.0±0.1	78.9±0.1	67.3
DIWA (Rame et al., 2022)	72.8	89.0	47.7	51.9	78.6	68.0
ERM + MIRO + SWAD (Cha et al., 2022)	72.4±0.1	88.4±0.1	47.0±0.0	**52.9**±0.2	**79.6**±0.2	68.1
ERM++ (Ours)	**74.7**±0.0	**89.8**±0.3	**50.8**±0.0	51.2±0.3	78.0±0.1	**68.9**

Table 1. **Comparison to recent methods:** Performance of recent methods as reported by (Cha et al., 2022). ERM outperforms almost all prior work, especially when combined with techniques such as SWAD and MIRO. ERM++ outperforms all prior work on average.

ERM++ Components (#7 represents full ERM++)								OfficeHome	PACS	VLCS	DomainNet	TerraInc	Avg.
#	MPA	FD	LT	WS	ES	S. Init	UBN	15K	10K	11K	590K	25K	
1	✗	✗	✗	✗	✗	✗	✓	67.1±0.2	85.1±0.3	76.9±0.6	44.1±0.15	45.2±0.6	63.7
2	✓	✗	✗	✗	✗	✗	✓	70.2±0.3	85.7±0.2	78.5±0.3	46.4±0.0	49.4±0.4	66.0
3	✓	✓	✗	✗	✗	✗	✓	71.5±0.1	87.3±0.2	77.4±0.1	46.8±0.0	49.8±0.5	66.5
4	✓	✓	✓	✗	✗	✗	✓	71.7±0.1	88.7±0.2	76.9±0.1	48.3±0.0	49.6±0.4	67.0
5	✓	✓	✓	✓	✗	✗	✓	72.6±0.1	88.8±0.1	77.0±0.1	48.6±0.0	49.3±0.3	67.3
6	✓	✓	✓	✓	✓	✗	✓	72.6±0.1	88.8±0.1	**78.7**±0.0	48.6±0.0	49.2±0.3	67.6
7	✓	✓	✓	✓	✓	✓	✓	**74.7**±0.0	89.8±0.3	78.0±0.1	**50.8**±0.0	**51.2**±0.3	**68.9**
8	✓	✓	✗	✓	✓	✓	✓	74.6±0.1	87.9±0.2	78.6±0.1	49.8±0.0	51.1±0.8	68.4
9	✓	✓	✓	✓	✓	✓	✗	**74.7**±0.2	**90.1**±0.0	78.6±0.1	49.9±0.0	49.0±0.4	68.3

Table 2. We present the overall ablation for ERM++, with standard errors. ERM++ corresponds to experiment 7. (1) ERM (Gulrajani & Lopez-Paz, 2020) baseline with unfrozen BN. (2) MPA: Model parameter averaging, which uniformly improves results. (3) FD: training on the full data. (4) LT: Training for 4x longer, which ensures convergence improves. (5) WS: Warm-starting the classification layer. (6) ES: Splitting off validation data to find a training length yields substantial gains. (7) S.Init: Initializing the initial parameters to those trained with AugMix brings performance to state of the art. (8) Removing LT from (7) still results in state-of-the-art performance with half of the training cost of MIRO. (9) UBN: When we freeze the BN parameters, we see that performance substantially degrades.

	OH	PA	VL	DN	TI	Avg
MIRO + SWAD	72.4	88.4	**79.6**	47.0	52.9	68.1
DIWA	72.8	89.0	78.6	47.7	52.9	68.0
ERM++	74.7	89.8	78.0	50.8	51.2	68.9
DIWA + ERM++	75.1	**90.0**	78.6	**51.5**	51.4	69.3
MIRO + ERM++	**76.3**	88.8	77.9	50.4	**53.4**	**69.4**

Table 3. We combine ERM++ with MIRO (Cha et al., 2022) and DIWA(Rame et al., 2022) Both DIWA and MIRO improve performance, validating that DIWA and MIRO are effective methods even when built on top of a stronger baseline.

layer (Goyal et al., 2017; He et al., 2016; Kumar et al., 2022; Rame et al., 2022). To address this, researchers would begin training by Warmstart (*WS*, Table 2, Experiment 5) (Kanavati & Tsuneki, 2021; Zhai & Wu, 2018) (also commonly referred to as warmup), where the new layer weights are trained with all pretrained weights kept frozen for a few epochs. After this short training cycle, new and old layer weights are finetuned together (sometimes except for BN). The full ERM++ baseline uses *WS*.

2.3. Weight-Space Regularization

Averaging model parameter iterates has a long history within machine learning (Arpit et al., 2021; Cha et al., 2021; Izmailov et al., 2018; Ruppert, 1988; Wortsman et al., 2022a;b; Rame et al., 2022; Li et al., 2022), and improves generalization by converging to flatter minima (Izmailov et al., 2018). Methods can roughly be divided into those which average within a single trajectory (Arpit et al., 2021; Izmailov et al., 2018; Cha et al., 2021), or those between different trajectories originating from a single parent (Li et al., 2022; Wortsman et al., 2022a; Rame et al., 2022). Because the model parameters averaged can be interpreted as independent members of an ensemble (Wortsman et al., 2022a), most prior work takes care to ensure the averaged models are sufficiently diverse and each member has strong performance. This is accomplished by cyclic learning rates (Izmailov et al.,

	OfficeHome	PACS	VLCS	DomainNet	TerraIncognita	Average	ImageNet Accuracy
TorchVision Weights	72.2	85.9	78.5	46.9	49.7	66.6	76.1
AugMix Trained Weights	74.6	**87.9**	78.6	**49.8**	**51.0**	**68.4**	79.0
Meal V2	**75.5**	86.7	**79.1**	49.5	50.9	68.3	**80.7**
ResNet A1	70.8	82.8	77.7	43.0	37.3	62.3	80.4

Table 4. **Top-1 Accuracy with different ResNet-50 initialization**: We investigate initialization weights from different pre-training procedures. The differences between different initializations are very substantial, up to about 6%. Interestingly, improved ImageNet accuracy does not strongly correlate with improved performance. The very strong ResNet A1(Wightman et al., 2021) performs a full 6% worse than the AugMix weights on DG tasks.

	P	C	I	R	Q	S	Av
Aug	**57.3**	**68.8**	**25.6**	70.2	**17.1**	**59.8**	**49.8**
MV2	**57.3**	68.5	25.4	**70.9**	16.1	59.0	49.5

Table 5. **Model distillation's effect on domain generalization:** We look at the per-domain accuracy on DomainNet, comparing Augmix training (Aug) and MealV2 (MV2).

2018) or searching ranges over which to average (Cha et al., 2021). Most recently, Arpit et al. (Arpit et al., 2021) revisit a simple method for parameter averaging where simply all iterates are averaged (*MPA*). We verify that this works in combination with other techniques present in ERM++ in Table 2, Experiment 2. Therefore *MPA* is part of ERM++.

2.4. ERM++ Computational Cost

ERM++ induces less training cost overhead compared to competing methods. ERM (Gulrajani & Lopez-Paz, 2020), DIWA (Rame et al., 2022), and MIRO (Cha et al., 2022) all use expensive hyper-parameter searches, while we simply use reasonable default ones. For example, MIRO (Cha et al., 2022) searches over 4 λ regularization weight parameters to obtain SOTA results, and DIWA (Rame et al., 2022) averages 20-60 independent runs. Overall, without long training, ERM++ achieves SOTA accuracy with 50% of the training compute of MIRO and 5% of the compute of DIWA (Rame et al., 2022), while retaining the same inference overhead. This is even accounting for the two training passes needed for Early Stopping with the full data; we do two passes over the data instead of the 4 used to obtain MIRO's result. Further hyper-parameter tuning would likely improve any method, but for a given training budget, ERM++ outperforms prior work.

3. Experiments

Experimental Setup. We follow the DomainBed training procedure and add additional components from ERM++. In particular, we use the default hyper-parameters from DomainBed (Gulrajani & Lopez-Paz, 2020), *e.g.*, a batch size of 32 (per-domain), a learning rate of 5e-5, a ResNet dropout value of 0, and a weight decay of 0. Unless we specify that the "Long Training" component is added, we train models for 15000 steps on DomainNet (following SWAD(Cha et al., 2021)) and 5000 steps for other datasets, which corresponds to a variable number of epochs dependent on dataset size. If Long Training is used, we extend training by 4x. We train on all source domains except for one, validate the model on held-out data from the sources every 300 steps (1000 for DomainNet), and evaluate on the held-out domain.

Results. Table 1 compares ERM++ to prior work, where we outperform the state-of-the-art across five DomainBed datasets by an average of 1%. The single largest gain was on DomainNet (3% gain), with OfficeHome and PACS obtaining still substantial gains of 1.5-2%. Table 2 ablates the importance of each piece, and show that each component of ERM++ is important to achieve the best final performance. *Strong Init* (1.3%) and *Model Parameter Averaging*(2.3%) contribute the most to improved performance, though all pieces are important.

Table 3 demonstrates our training procedure's ability to generalize, where we combine our approach with the two highest performing methods in prior work (DIWA (Rame et al., 2022) and MIRO (Cha et al., 2022)). We find that our approach is able to boost the performance of both methods by over 1%. This validaties that DIWA and MIRO are effective methods even when built on top of a stronger baseline.

Finally, Table 4 show the impact of different initialization. Surprisingly, it is clear that the most generalizable model is not necessarily the one which achieves the best ImageNet performance. Model Distillation, which strongly improves source accuracy, does not increase overall DG performance. Meal-V2 is a distillation of the ensemble if two very strong ImageNet models into a ResNet-50. Interestingly, the student in Meal-V2 is initialized with the same AugMix trained network as we do in our experiments. Therefore, the differences in performance can be strictly attributed to the effects of model distillation. In Table 5, we can see that performance on ImageNet-like domains improves while performance on other domains suffers. This suggests that the distillation process effectively matches the student to the

teacher over the data used in the distillation process, at the price of function smoothness away from the distillation data.

We provide additional analysis and results of ERM++ components in the Appendix.

4. Conclusion

This paper develops a strong baseline, ERM++, that is critically missing from the literature. By identifying easy-to-implement techniques, ERM++ achieves significant gains in DG performance, reporting a 5% average boost over ERM on the challenging DomainBed evaluation datasets. We recommend that future works compare new methods to ERM++ instead of ERM as a baseline, and that they extend ERM++ instead of ERM when relevant. When we replace ERM with ERM++ for DIWA(Rame et al., 2022) and MIRO(Cha et al., 2022), we find that performance improves for both. Our results highlight the importance of improving the training procedure for better DG performance.

Acknowledgment

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

References

Angarano, S., Martini, M., Salvetti, F., Mazzia, V., and Chiaberge, M. Back-to-bones: Rediscovering the role of backbones in domain generalization. *arXiv preprint arXiv:2209.01121*, 2022.

Arpit, D., Wang, H., Zhou, Y., and Xiong, C. Ensemble of averages: Improving model selection and boosting performance in domain generalization. *arXiv preprint arXiv:2110.10832*, 2021.

Beery, S., Van Horn, G., and Perona, P. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 456–473, 2018.

Blanchard, G., Lee, G., and Scott, C. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 24, 2011.

Cha, J., Chun, S., Lee, K., Cho, H.-C., Park, S., Lee, Y., and Park, S. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418, 2021.

Cha, J., Lee, K., Park, S., and Chun, S. Domain generalization by mutual-information regularization with pre-trained models. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII*, pp. 440–457. Springer, 2022.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Fang, A., Kornblith, S., and Schmidt, L. Does progress on imagenet transfer to real-world datasets? *arXiv preprint arXiv:2301.04644*, 2023.

Fang, C., Xu, Y., and Rockmore, D. N. 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*, pp. 1657–1664, 2013.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Diverse weight averaging for out-of-distribution generalization. *arXiv preprint arXiv:1706.02677*, 2017.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. AugMix: A simple data processing method to improve robustness and uncertainty. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Kanavati, F. and Tsuneki, M. Partial transfusion: on the expressive influence of trainable batch norm parameters for transfer learning. In *Medical Imaging with Deep Learning*, pp. 338–353. PMLR, 2021.

Kim, D., Wang, K., Sclaroff, S., and Saenko, K. A broad study of pre-training for domain generalization and adaptation. *arXiv preprint arXiv:2203.11819*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kumar, A., Raghunathan, A., Jones, R., Ma, T., and Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

Lee, Y., Chen, A. S., Tajwar, F., Kumar, A., Yao, H., Liang, P., and Finn, C. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

Li, M., Gururangan, S., Dettmers, T., Lewis, M., Althoff, T., Smith, N. A., and Zettlemoyer, L. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*, 2022.

Muandet, K., Balduzzi, D., and Schölkopf, B. Domain generalization via invariant feature representation. In *International conference on machine learning*, pp. 10–18. PMLR, 2013.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019.

Rame, A., Kirchmeyer, M., Rahier, T., Rakotomamonjy, A., Gallinari, P., and Cord, M. Diverse weight averaging for out-of-distribution generalization. *arXiv preprint arXiv:2205.09739*, 2022.

Ruppert, D. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

Shen, Z. and Savvides, M. Meal v2: Boosting vanilla resnet-50 to 80%+ top-1 accuracy on imagenet without tricks. *arXiv preprint arXiv:2009.08453*, 2020.

Sun, B. and Saenko, K. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pp. 443–450. Springer, 2016.

Vapnik, V. N. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5018–5027, 2017.

Wightman, R. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Wightman, R., Touvron, H., and Jégou, H. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022a.

Wortsman, M., Ilharco, G., Kim, J. W., Li, M., Kornblith, S., Roelofs, R., Lopes, R. G., Hajishirzi, H., Farhadi, A., Namkoong, H., et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7959–7971, 2022b.

Zhai, A. and Wu, H.-Y. Classification is a strong baseline for deep metric learning. *arXiv preprint arXiv:1811.12649*, 2018.

Zhang, M., Marklund, H., Dhawan, N., Gupta, A., Levine, S., and Finn, C. Adaptive risk minimization: Learning to adapt to domain shift. *Advances in Neural Information Processing Systems*, 34:23664–23678, 2021.

Zhou, K., Yang, Y., Qiao, Y., and Xiang, T. Domain generalization with mixstyle. *arXiv preprint arXiv:2104.02008*, 2021.

	R0	R1	R2	R3	R4	Av.
ERM	34.9	47.1	38.6	43.8	53.8	43.6
ERM++	**41.5**	50.3	40.5	50.4	57.7	48.1
- Strong Init	39.2	50.1	39.5	49.5	58.7	47.4
- WS	41.3	**50.4**	**41.0**	**50.6**	**59.3**	**48.5**
- UBN	39.6	49.1	38.9	49.1	58.1	47.0

Table 6. **WILDS-FMOW Top-1 Accuracy:** We show that ERM++ outperforms ERM on this on the challenging WILDS-FMOW classification dataset. We also ablate several components of ERM++. UBN (Unfrozen Batch Norm) and Strong Init (from Augmix) improve performance, while surprisingly WS (warmstart) decreases performance in this particular scenario. We emphasize that ERM++ overall improves over ERM(Gulrajani & Lopez-Paz, 2020).

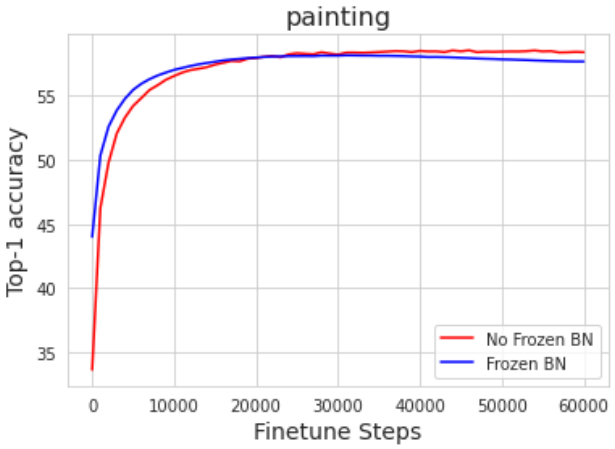

Figure 2. **Unfreezing Batchnorm:** Here we show the test curves of the fine-tuning on the held-out painting domain of DomainNet. With frozen BatchNorm, the initial training is faster but it overfits.

A. Additional Results

A.1. Generalizing Beyond Web-scraped Datasets

We have demonstrated that ERM++ is a highly effective recipe for DG on several datasets: OfficeHome, PACS, DomainNet, and TerraIncognita. These datasets are diverse and represent a strong evaluation of ERM++. However, (Fang et al., 2023) show that on datasets not consisting of web-scraped data, the correlation between ImageNet performance and transfer performance is quite weak. To verify that this is not the case for ERM++, we perform an ablation study on WILDS-FMOW, a land-use classification dataset, and see that the ERM++ substantially improves over ERM (Table 6).

A.2. Per-dataset details

In Tables 7 (OfficeHome), 9 (DomainNet), 10 (VLCS), 11 (TerraIncognita), 12 (PACS), we expand results for the datasets and report accuracies for each held-out domain. We compare ERM++ with reported performances of ERM (Gulrajani & Lopez-Paz, 2020), DIWA (Rame et al., 2022), SWAD, (Cha et al., 2021), and MIRO (Cha et al., 2022). ERM + SWAD + MIRO and DIWA are the current SOTA for ResNet-50 models for this set of datasets. Overall trends include ERM++ being especially effective at sketch-like domains, indicating a lowered texture bias. On the sketch and clipart domains in DomainNet, ERM++ outperforms prior best performance by over 4%. When we additionally combine MIRO with ERM++, we see much improved performance on OfficeHome and TerraIncognita without much affecting the performance on the other datasets.

A.3. Additional Analysis: Data Utilization

Using the full data (*FD*): The most common ERM (Gulrajani & Lopez-Paz, 2020) implementation splits off 80% of the source domains for training, and keeps the remaining 20% for hyper-parameter validation and checkpoint selection. By comparing Table 2 in experiments 2 and 3, we show that training on the full data improves over checkpoint selection on a validation set on all datasets except for VLCS. Early Stopping (*ES*) below helps us recover VLCS performance.

Long training (*LT*): Prior work has shown that training to proper convergence can have large impacts on transfer learning performance (Chen et al., 2020). To explore this setting for DG, we extended training by 4x for each dataset. In other words, DomainNet models are trained for 60K steps while the other datasets are trained for 20K steps. This training length is one where we observe source validation accuracies start to saturate for most datasets (Section A.6). We present the results in Table 2, experiment 4. We find that training for longer, on average, increases performance by 0.5%.

Early Stopping (*ES*): Although the training pieces presented so far improve DG performance on the datasets considered on average, one consistent pattern is that VLCS performance degrades in experiments number 3 (Full-Data), 4 (Long training). This suggests that VLCS is a dataset which is prone to overfitting. We observe that this is true even on a validation set constructed from the source domains. Therefore, we propose an additional step where we use a random 20% of the training data as validation data in order to search for the proper number of training steps, and then retrain using the full data. In Table 2, Experiment 6, we see this dramatically improves performance on VLCS without affecting other datasets.

	art	clipart	product	real	avg
ERM (Gulrajani & Lopez-Paz, 2020)	63.1	51.9	77.2	78.1	67.6
ERM + SWAD (Cha et al., 2021)	66.1	57.7	78.4	80.2	70.6
DIWA (Rame et al., 2022)	69.2	59	81.7	82.2	72.8
ERM + MIRO + SWAD (Cha et al., 2022)	-	-	-	-	72.4
ERM++	70.7	**62.2**	81.8	84.0	74.7
ERM++ + MIRO	**74.0**	61.5	**83.8**	**85.7**	**76.3**

Table 7. **OfficeHome:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. (Cha et al., 2022) does not report per-domain performance for MIRO, so we only show average for that case. DIWA doesn't report standard errors. ERM++ not only greatly increases performance relative to SWAD, DIWA, and MIRO but also reduce variance between runs. The largest gains are on the held-out domain with the largest domain shift(clipart), illustrating the ability of ERM++ to improve performance on difficult DG tasks.

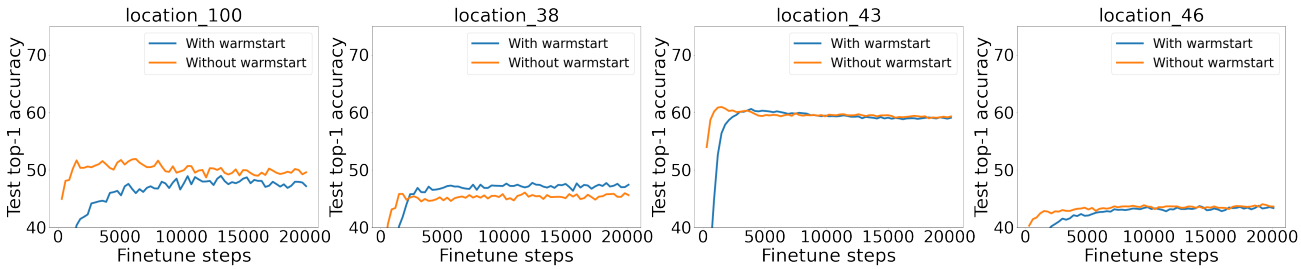

Figure 3. **Oracle test performances** : We plot the top-1 accuracy on held-out test domains of TerraIncognita as a function of fine-tuning epochs with and without warmstart. Warmstart substantially decreases overfitting to the source domains.

A.4. Additional Analysis: Pretrained Model Weight Usage

Warmstart (*WS*): In Table 2, we compare to training using a random initialization for the new classification layer (Experiment 4) or by using Warmstart (Experiment 5). We find WS provides a small but consistent boost on average across datasets. We find this is likely due to a decrease in overfitting to the source domains. For example, in Figure 3, we show accuracies plotted across fine-tuning steps for models with and without warm-start for several domains of TerraIncognita. Without Warmstart, performance quickly plateaus and in some cases, *e.g.*, location 100 and location 43, performance even decreases. This kind of performance decrease is not benign; it is impossible to detect without access to the test data. Therefore, for reliable deployments of systems which generalize well, training procedures which do not overfit to source domains are important. We verify that WS has a regularization effect by measuring the L2 distance of the final model from initialization (the pre-trained model) and find that the trained weights were more than twice as far without using WS (58.1 with and 122.5 w/o).

Unfreezing the Batchnorm (*UBN*): BatchNorm is commonly frozen in current DG recipes for reasons not well justified. However, we find that frozen batch normalization leads to quick over-fitting in the long-training regime. In Figure 2 we can see that frozen batch normalization results in overfitting. In contrast, without frozen batch normalization this is not an issue. As seen in Table 2, Experiment 9, this freezing BN also results in lower performance. It can therefore be concluded that unfrozen BatchNorm, gives an effective regularization effect by randomizing shifting and scaling of features.

Stronger initializations (*S. Init*): One of the key components of the standard DG training scheme is initializing the model parameters with a pre-trained model. The effect of the strong initialization for our model is shown in Table 2, experiment 7, where we achieve a 1% boost an average. However, selecting a model takes care. Table 4 compares ResNet-50 models of varying ImageNet performance. We summarize our findings below:

- Stronger ImageNet performance does not necessarily correspond to better DG performance. In particular, both the ResNet-50 A1 and Meal V2 weights achieves much better ImageNet Top-1 Accuracy than the standard TorchVision weights, but achieve worse DG performance. However, the overall consistency of the AugMix weights across all 5 datasets makes it a reasonable choice.
- Model Distillation, which strongly improves source accu-

	P	C	I	R	Q	S	Av
Aug	**57.3**	**68.8**	**25.6**	70.2	**17.1**	**59.8**	**49.8**
MV2	**57.3**	68.5	25.4	**70.9**	16.1	59.0	49.5

Table 8. **Model distillation's effect on domain generalization:** We look at the per-domain accuracy on DomainNet, comparing Augmix training (Aug) and MealV2 (MV2). MealV2 is a method used to distill a large ensemble into a student ResNet-50, where the student is initialized to AugMix weights. The held-out domains considered are (P)ainting, (C)lipart, (I)nfo, (R)eal, (Q)uickdraw, and (S)ketch. We can see that the distillation process, while dramatically improving ImageNet performance, only slightly changes generalization performance. In particular, generalization gets slightly worse for all domains except for (R)eal, which is the most similar to ImageNet. This is surprising, since it has been shown that both ensembles (Arpit et al., 2021) and larger model (Angarano et al., 2022) improve domain generalization performance. The distillation process seems to match the teacher function poorly on OOD data.

racy, does not increase overall DG performance. Meal-V2 is a distillation of the ensemble if two very strong ImageNet models into a ResNet-50. Interestingly, the student in Meal-V2 is initialized with the same AugMix trained network as we do in our experiments. Therefore, the differences in performance can be strictly attributed to the effects of model distillation. Looking at the results in more detail, as in Table 8, we can see that performance on ImageNet-like domains improves while performance on other domains suffers. This suggests that the distillation process effectively matches the student to the teacher over the data used in the distillation process, at the price of function smoothness away from the distillation data.

- AugMix is a model trained with generalization to synthetic corruptions as a goal and results in a very strong DG performance. Therefore, while ImageNet Top-1 accuracy is not a good indicator for DG performance, investigating the correlation between synthetic corruption performance and DG performance is promising.

A.5. Additional Analysis: Weight Space Regularization

Generalist Model Parameter Averaging (*MPA*): We confirm that regularizing model parameters by averaging iterates is an important tool in improving domain generalization performance; in Table 2 (Experiments 1 and 2) we compare models trained with and without parameter averaging across timesteps. Specifically, we average the parameters of all training steps after an initial burn-in period of 100 steps. We confirm that such model parameter averaging consistently and substantially improves domain generalization.

Specialist Model Parameter Averaging (*SMPA*): We also explored a setting where instead of averaging model weights, we attempt to include diversity between the models being averaged as this has been shown to boost performance (Rame et al., 2022). Following (Li et al., 2022), we first train a generalist model on all source domains for 5 epochs, then train specialist models for 5 epochs, before averaging parameters. Results on the DomainNet dataset are reported in Table 13. Although averaging specialists improves over ERM, it does not improve over averaging model iterates of a generalist.

Figure 4. **Sample from LabelMe Domain in VLCS:** Is this a dog, person, or chair? Many samples in the LabelMe domain of VLCS are ambigrous but assigned a label (in this case, dog). This raises questions about the usefulness of training on this domain.

A.6. Validation-Set Accuracy Curves

In Figures 5,6,7,8, and 9, we provide source-validation accuracies for each of the 5 datasets, for the number of steps corresponding to *long training*, which is 20000 steps for most datasets except for the larger DomainNet, which is 60000 steps. As one can see, at this point, validation accuracy is saturated for most domains in most datasets, so this training length is reasonable. Prior training lengths are denoted as red vertical lines in these figures, and one can see that for many datasets this is not a sufficient training length. As we describe in Section 5.1 of the main paper, this improves performance by 0.5% on average.

B. Dataset Visualizations

In Figures 10 (OfficeHome), 11 (DomainNet), 12 (VLCS), 13 (TerraIncognita), 14 (PACS), 15 (FMoW) we show samples of a few classes from each of the datasets, and each domain. As one can see, both the datasets and distribution shifts are quite diverse, highlighting the flexibility of our method. We present some key attributes of the datasets below.

OfficeHome (Venkateswara et al., 2017) Figure 10. This dataset focuses on household objects. The domain shifts are in low-level style mostly, and there is little spatial bias.

	painting	clipart	info	real	quickdraw	sketch	avg
ERM (Gulrajani & Lopez-Paz, 2020)	50.1	63.0	21.2	63.7	13.9	52.9	44.0
ERM + SWAD (Cha et al., 2021)	53.5	66.0	22.4	65.8	16.1	55.5	46.5
DIWA (Rame et al., 2022)	55.4	66.2	23.3	68.7	16.5	56	47.7
ERM + MIRO + SWAD (Cha et al., 2022)	-	-	-	-	-	-	47.0
ERM++	58.4	**71.5**	26.2	70.7	**17.3**	**60.5**	**50.8**
ERM++ + MIRO	**58.5**	71.0	**26.5**	**71.1**	15.9	59.5	50.4

Table 9. **DomainNet:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. (Cha et al., 2022) does not per-domain performance for MIRO, so we only show average for that case. DIWA doesn't report standard errors. ERM++ not only greatly increases performance relative to SWAD, DIWA, and MIRO but also reduce variance between runs. Similar to results on OfficeHome (Table 7), the largest performance gains(of larger than 4%) are on domains very different from the source domain(clipart and sketch). This suggests ERM++ is less sensitive to texture bias than ERM (Gulrajani & Lopez-Paz, 2020). The bias of MIRO to the pre-trained weights manifests in slightly higher performance on close to ImageNet domains like real when combined with ERM++, at the slight expense of performance on other domains.

	caltech101	labelme	sun09	voc2007	avg
ERM (Gulrajani & Lopez-Paz, 2020)	97.7	**64.3**	73.4	74.6	77.3
ERM + SWAD (Cha et al., 2021)	98.8	63.3	**75.3**	**79.2**	79.1
DIWA (Rame et al., 2022)	**98.9**	62.4	73.9	78.9	78.6
ERM + MIRO + SWAD (Cha et al., 2021)	-	-	-	-	**79.6**
ERM++	98.7	63.2	71.6	78.7	78.0
ERM++ + MIRO	99.0	62.4	71.8	78.3	77.9

d

Table 10. **VLCS:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. (Cha et al., 2022) does not per-domain performance for MIRO, so we only show average for that case. DIWA doesn't report standard errors. Although overall performance on VLCS is lower than competing methods, we can see that this drop primarily comes from lower performance on sun09. Furthermore, there are many ambiguous images in the LabelMe domain (see Figure 4), raising questions about the usefulness of trying to train on this domain.

DomainNet (Peng et al., 2019) Figure 11. While the real domain is quite similar to what one might expect in ImageNet, the distribution shifts are quite substantial in other domains. Quickdraw and Infograph are particularly challenging, so the 1-3% gains of ERM++ on these domains is meaningful (Table 9).

VLCS (Fang et al., 2013): Figure 12. Low-level statistics are quite similar between domains in this dataset, however spatial biases differ between domains. For example, Caltech objects are quite centered, while other domains do not have this trait. For example the LabelMe domain has cars along the side of the image, and there are many chairs in the VOC2007 domain. Furthermore, in some cases the size of the objects differs dramatically. Lastly, there are many ambiguous images in the LabelMe domain (see Figure 4), raising questions about the validity of trying to improve performance on this dataset.

TerraIncognita (Beery et al., 2018): Figure 13 The background stays consistent, and the animal object frequently takes up a small portion of the frame. At night the images are black-and-white. This is a very realistic dataset, on which is good to test.

PACS (Li et al., 2017) Figure 14. The subjects tend to be centered, and the sketches are more realistic than the quickdraw setting in DomainNet. Though the domains are similar to that of DomainNet, PACS has fewer than 10000 samples compared to 586000 of DomainNet. Therefore PACS tests the capabilities of ERM++ on smaller data.

FMoW: Figure 15. The images differ in region but also in resolution and scale. The distribution shift between FMoW and the pretraining data is large, therefore FmoW represents the ability of ERM++ to perform on non web-scraped data (see Section 5.4 of the main paper).

C. Runtime Comparisons

As discussed in the main paper Section 3.4; ERM++ achieves higher predictive performance than competing methods MIRO (Cha et al., 2022) and DIWA (Rame et al., 2022) despite lower computational cost for training. The rea-

	Location 100	Location 38	Location 43	Location 46	Average
ERM (Gulrajani & Lopez-Paz, 2020)	54.3	42.5	55.6	38.8	47.8
ERM + SWAD (Cha et al., 2021)	55.4	44.9	59.7	39.9	50.0
DIWA (Rame et al., 2022)	**57.2**	50.1	60.3	39.8	51.9
ERM + MIRO + SWAD (Cha et al., 2022)	-	-	-	-	52.9
ERM++	48.3	**50.7**	**61.8**	**43.9**	51.2
ERM++ + MIRO	**60.81**	48.8	61.1	42.7	**53.4**

Table 11. **TerraIncognita:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. (Cha et al., 2022) does not per-domain performance for MIRO, so we only show average for that case. DIWA doesn't report standard errors. ERM++ outperforms other methods on 3 out of 4 held out domains despite slighly underperforming on average. However, we point out that ERM++ w/MIRO outperforms both DIWA and MIRO, and improves ERM++ by a further 2%.

	art_painting	cartoon	photo	sketch	avg
ERM (Gulrajani & Lopez-Paz, 2020)	84.7	80.8	97.2	79.3	84.2
ERM + SWAD (Cha et al., 2021)	89.3	83.4	97.3	82.5	88.1
DIWA (Rame et al., 2022)	90.6	83.4	98.2	83.8	89
ERM + MIRO + SWAD (Cha et al., 2022)	-	-	-	-	88.4
ERM++	**90.6**	83.7	98.1	**86.6**	**89.8**
ERM++ + MIRO	90.2	**83.8**	**98.6**	82.4	88.8

Table 12. **PACS:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. (Cha et al., 2022) does not per-domain performance for MIRO, so we only show average for that case. DIWA doesn't report standard errors. ERM++ leads to substantial improvement over prior work. As in other dataset (OfficeHome, DomainNet), large performance gains are made on the sketch domain.

	P	I	Q	S	R	C	Av
ERM	51.1	21.2	13.9	52.0	63.7	63.0	44.1
SMPA	52.9	**27.2**	14.3	51.3	65.6	65.2	46.1
MPA	**55.2**	24.0	**16.7**	**57.4**	**67.0**	**67.49**	**48.0**

Table 13. Weight Space Regularization: We show experiments different types of parameter averaging for weight regularization on DomainNet. **SMPA** is a specialized model parameter averaging, where we average parameters of domain specialists, while **MPA** averages parameters within a single training trajectory. While both **MPA** and **SMPA** outperform ERM, **MPA** outperforms **SMPA**.

son is reduced cost of hyper-parameter search; we use fixed hyper-parameters, borrowed from the DomainBed framework, (see Section D.2 for more details) while DIWA averages 20-60 models and MIRO search for *4* λ weight regularization values in each experiment. Assuming the worst case scenario of training two full passes (one on validation data for number of training steps for *Early Stopping*, and one on full training data with validation data folded in *Full Data*), and the same number of training steps as MIRO; ERM++ costs $\frac{1}{2}$ that of MIRO while obtaining better performance. In particular, this configuration represents Experiment 8 in Table 3 of the main paper.

For each forward step MIRO there is an additional forward pass of the data through the model which is absent in ERM++. On the other hand, ERM++ does take a forward pass through the running average model to update batch normalization statistics, which is not done in former methods. This means that each forward pass is compute-equivalent for ERM++ and MIRO, for a given architecture.

D. Reproducibility

D.1. Infrastructure

We train on a heterogeneous cluster, primarily on NVIDIA A6000 GPU's. Each experiment is conducted on a single GPU with 4 CPUs. A single run could range from 12-48 hours, depending on number of steps trained.

D.2. Training details

We follow the DomainBed (Gulrajani & Lopez-Paz, 2020) training procedure and add additional components from ERM++. In particular, we use the default hyper-parameters from DomainBed (Gulrajani & Lopez-Paz, 2020), *e.g.*, a batch size of 32 (per-domain), a learning rate of 5e-5, a ResNet dropout value of 0, and a weight decay of 0. We use the ADAM optimizer (Kingma & Ba, 2014) optimizer with β and ϵ values set default values from Pytorch 1.12. Unless we specify that the "Long Training" component

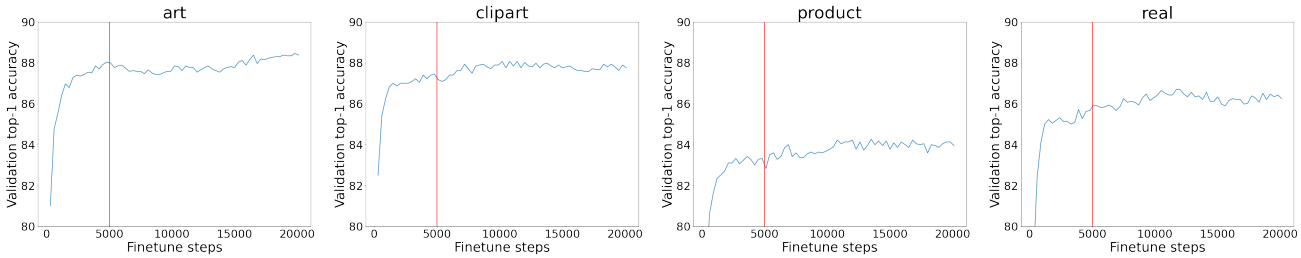

Figure 5. **OfficeHome:** Source validation accuracies. The validation accuracy saturates by 20000 steps, which corresponds to number of steps in *Long Training*(Section 5.1 of the main paper). Training length used in prior works is denoted as a red line, and the training is not yet converged.

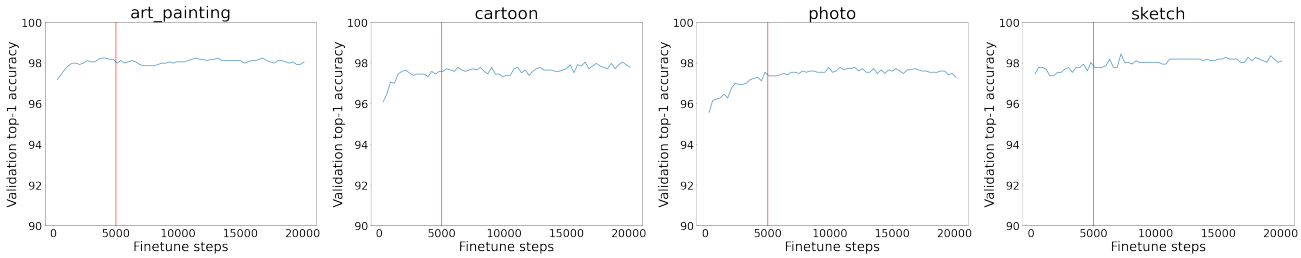

Figure 6. **PACS**: Source validation accuracies. The validation accuracy saturates by 20000 steps, which corresponds to number of steps in *Long Training*(Section 5.1 of the main paper).Training length used in prior works is denoted as a red line, and the training is not yet converged.

is added, we train models for 15000 steps on DomainNet (following SWAD(Cha et al., 2021)) and 5000 steps for other datasets, which corresponds to a variable number of epochs dependent on dataset size. If Long Training is used, we extend training by 4x. We train on all source domains except for one, validate the model on held-out data from the sources every 300 steps(20% of the source data), and evaluate on the held-out domain. If using *Full Data* we retrain using the full data. We use the same data augmentation techniques as ERM (Gulrajani & Lopez-Paz, 2020). We use the ResNet-50 architechture in all experiments.

Model Parameter Averaging details: If we use Model Parameter Averaging(*MPA*), we begin to keep a running average at the 100th step. If we additionally use warm-start, we only optimize the classification head for the first 500 steps, and start *MPA* 100 steps after that. For the Specialist Model Parameter Averaging(*SMPA*) experiments (Table 6 of main paper), we first train a generalist model for 15000 steps , then train an independent model for each domain for another 1500 steps. At the end, we average parameters and re-compute batch norm running statistics. This recomputing of BN stats makes sure the averaged model has accurately computed batch norm statistics which may not be a simple average of experts, due to the non-linearity of neural nets.

Batch Normalization details: With unfrozen batch normalization(*UBN*), we update the evaluation model BN statistics by averaging the model iterates first (from *MPA*), then then forward propagating the current batch at each step through the evaluation model. In this way, the BN running statistics and model used for inference match.

Sources of pre-trained weights: We use torchvision 0.13.1 for vanilla ResNet-50 initialization. For augmix and ResNet-A1 initialized weights, we leverage TIMM (Wightman, 2019) [1] [2] .

A note on hyper-parameter search: In this work, we focus on methodological improvements that do not depend on expensive hyper-parameter tuning, and as a result we use default learning rate, weight decay, etc. We demonstrate state-of-the-art performance despite this, and greatly reduce the computational cost of training as a result. However, we believe there is substantial headroom for improvement with further hyper-parameter tuning.

MIRO Implementation: We directly follow the MIRO

[1]Augmix Weights :https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-weights/resnet50_ram-a26f946b.pth

[2]ResNet-A1 Weights :https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-rsb-weights/resnet50_a1_0-14fe96d1.pth

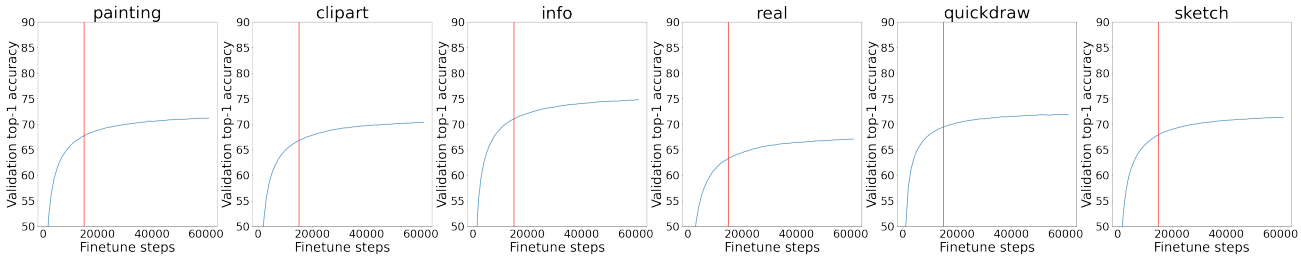

Figure 7. **DomainNet:** Source validation accuracies. The validation accuracy saturates by 60000 steps, which corresponds to number of steps in *Long Training*(Section 5.1 of the main paper). Training length used in prior works is denoted as a red line, and the training is not yet converged.

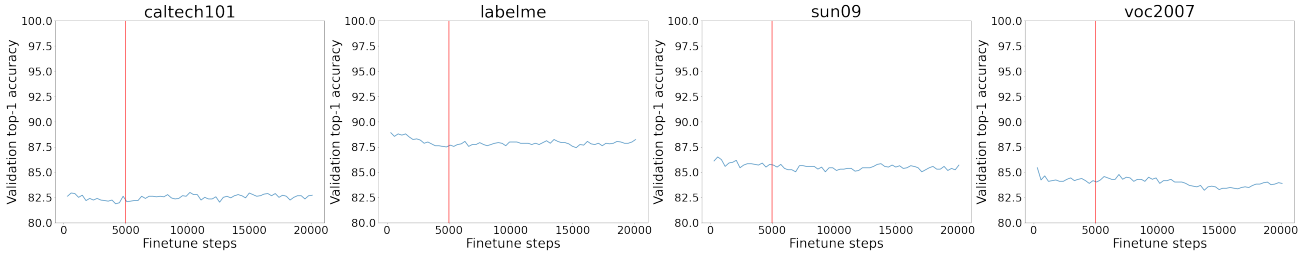

Figure 8. **VLCS:** Source validation accuracies. The validation accuracy saturates by 20000 steps, which corresponds to number of steps in *Long Training*(Section 5.1 of the main paper).Training length used in prior works is denoted as a red line. In the case of VLCS, it seems like longer training is not so helpful, and this is reflected in our ablations (Table 2)

.

implementation and borrow the lambda weights values from (Cha et al., 2022) when we combine MIRO with ERM++ in Table 2 of the main paper. ERM++ substantially improves the performance of MIRO.

DIWA Implementation: We follow a simplified version of the DIWA (Rame et al., 2022) algorithm due to computational reasons; we average the parameters of the three seeds of ERM++. The authors of DIWA show that about half of the performance boost comes from the first few models averaged (Figure 4 of (Rame et al., 2022)), therefore this is a reasonable approximation of the method. It is interesting that DIWA reduces performance of ERM++, but that ERM++ w/DIWA is still improved over DIWA as reported in (Rame et al., 2022).

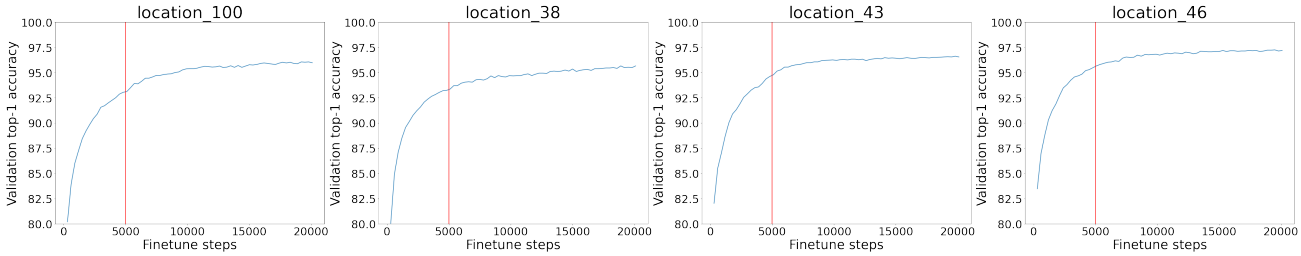

Figure 9. **TerraIncognita:** Source validation accuracies. The validation accuracy saturates by 20000 steps, which corresponds to number of steps in *Long Training*(Section 5.1 of the main paper). Training length used in prior works is denoted as a red line, and the training is not yet converged.

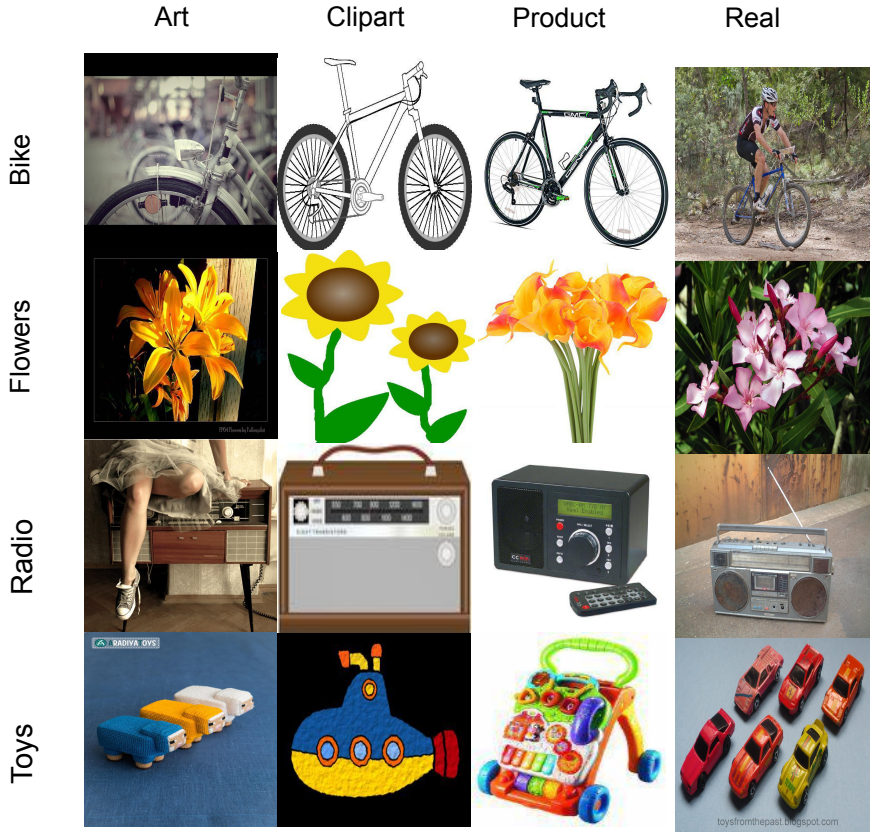

Figure 10. **OfficeHome**:Samples from the OfficeHome dataset, from each domain and selected classes. The dataset focuses on household objects. The domain shifts are in low-level style mostly, and there is little spatial bias.

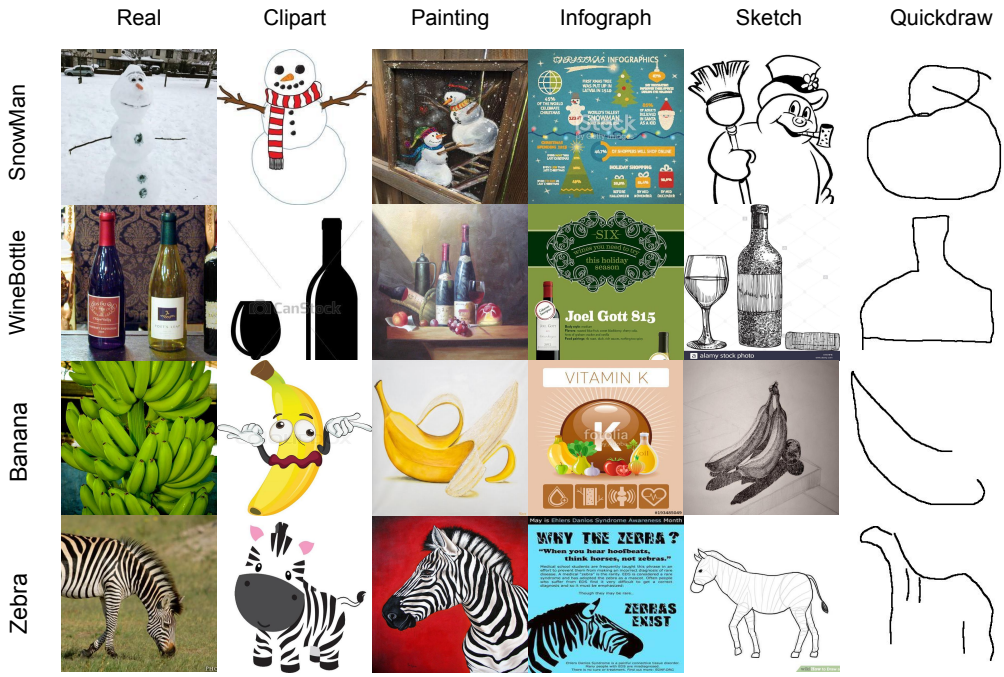

Figure 11. **DomainNet:** Samples from the DomainNet dataset. While the real domain is quite similar to what one might expect in ImageNet, the distribution shifts are quite substantial in other domains. Quickdraw and Infograph are particularly challenging, so the 1-3% gains of ERM++ on these domains is meaningful (Table 9). While most domains contain primarily shifts in low level statistics (for example, real to painting), Infograph also has many non-centered objects.

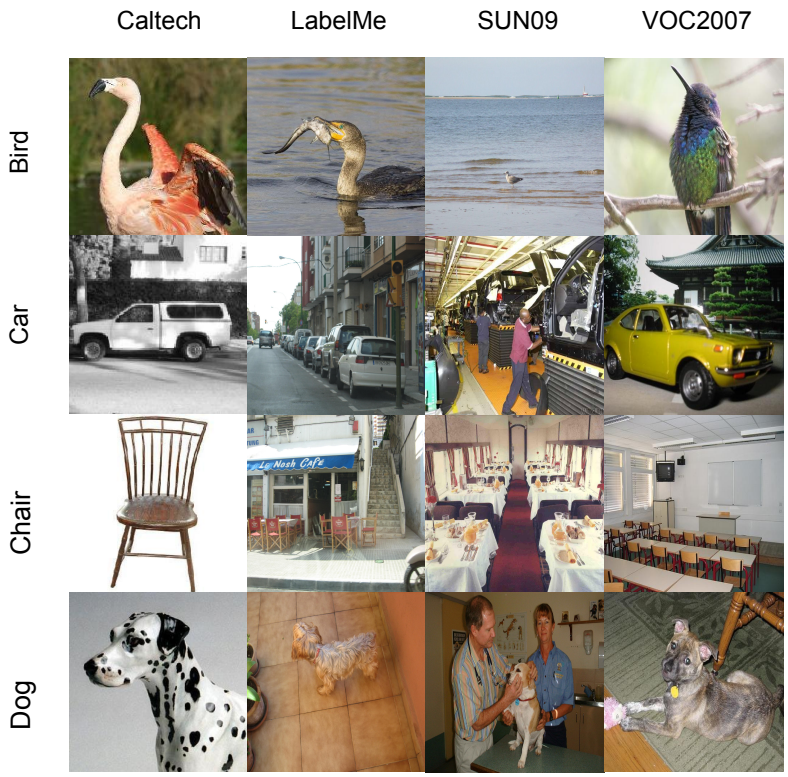

Figure 12. **VLCS:** The low-level statistics are quite similar between domains, however spatial biases differ between domains. Caltetch objects are quite centered, while other domains do not have this trait. For example the LabelMe domain has cars along the side of the image, and there are many chairs in the VOC2007 domain. Furthermore, in some cases the size of the objects differs dramatically. Finally, there are many ambiguous images in the LabelMe domain (see Figure 4), raising questions about the usefulness of trying to train on this domain.

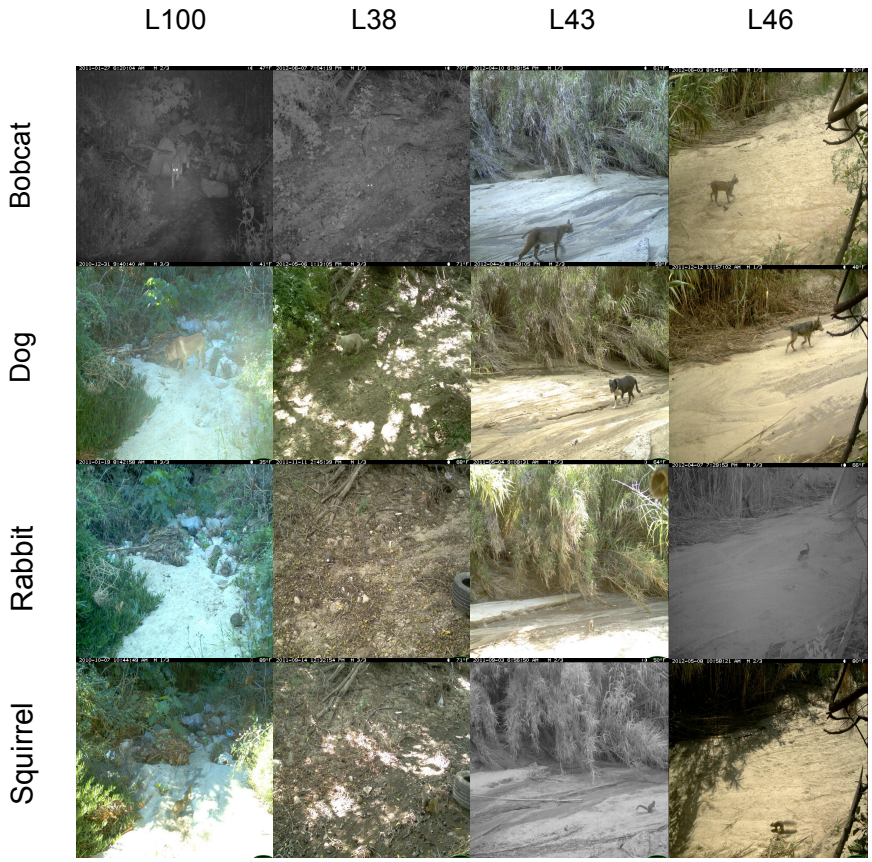

Figure 13. **TerraIncognita**: Samples from the TerraIncognita dataset, from each domain and selected classes. The background stays consistent, and the animal object frequently takes up a small portion of the frame. At night the images are black-and-white. This dataset matches realistic deployment scenarios well.

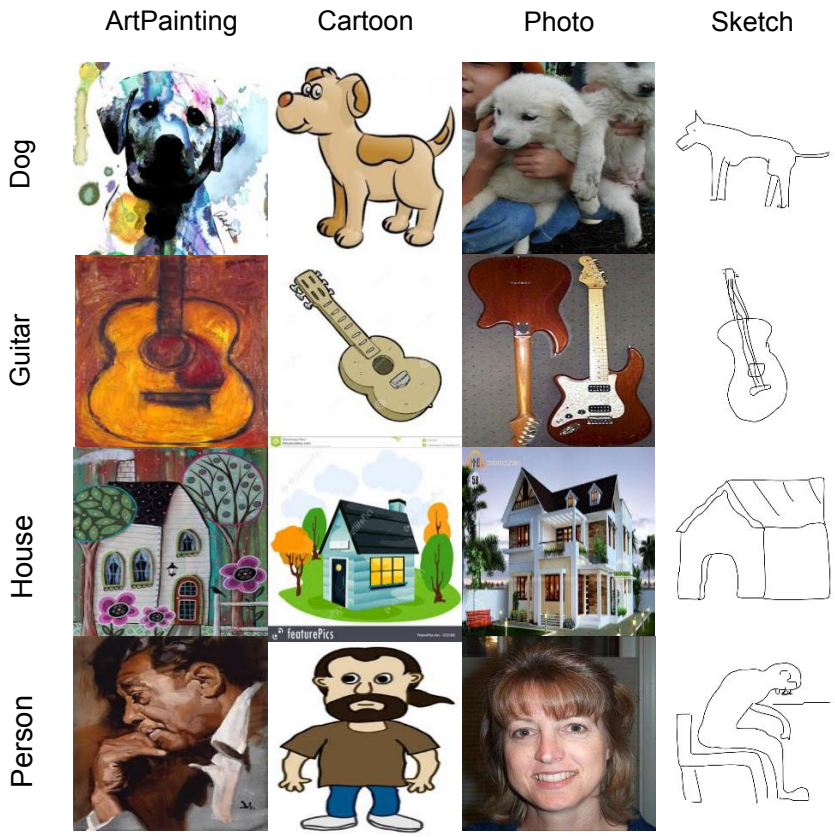

Figure 14. **PACS:** Samples from the PACS dataset, from each domain and selected classes. The subjects tend to be centered, and the sketches are more realistic than the quickdraw setting in DomainNet. Though the domians are similar to that of DomainNet, PACS has fewer than 10000 samples compared to 586000 of DomainNet. Therefore PACS tests the capabilities of ERM++ on smaller data.

Figure 15. **FMoW:**Samples from the TerraIncognita dataset, from each domain and selected classes. The images differ in region but also in resolution and scale. The distribution shift between FMoW and the pretraining data is large, therefore FmoW represents the ability of ERM++ to perform on non web-scraped data (see Section 5.4 of the main paper).