ERM++ : An Improved Baseline for Domain Generalization Supplementary Materials

Piotr Teterwak[‡] Kuniaki Saito[‡] Theodoros Tsiligkaridis[†] Kate Saenko[‡] Bryan A. Plummer[‡]
Boston University[‡] MIT Lincoln Laboratory[†]
{piotrt,keisaito,saenko,bplum}@bu.edu ttsili@mit.edu

1. Additional Results

1.1. Cumulative Study

In addition to the ablative study of ERM++ components in the main paper, we add a cumulative study in Table 1. Similar to the ablative study, we can see that performance is enhanced by each component of ERM++, in some cases by over 2% (Experiment 2, MPA).

1.2. MealV2 Distillation Results

In Table 2, 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. We can see that the distillation process, while dramatically improving ImageNet performance, only slightly changes DG 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 [2] and larger models [1] improve DG performance.

1.3. Weight space regularization

In Table 3 we explore 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 [16]. Following [14], we first train a generalist model on all source domains for 28k steps, then train specialist models for 28k steps (1 model per domain), before averaging parameters. Although averaging specialists improves over ERM by 2%, it underperforms averaging iterates of a generalist by 2%. One possible explanation is that each domain has too little data to create a good specialist model

1.4. Per-dataset details

In Tables 4 (OfficeHome), 5 (DomainNet), 6 (VLCS), 7 (TerraIncognita), 8 (PACS), we expand results for the datasets and report accuracies for each held-out domain. We compare ResNet-50 ERM++ with reported performances of ERM [9], DIWA [16], SWAD, [5], and MIRO [6]. 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.

1.5. Validation-Set Accuracy Curves

In Figures 10,11,12,13, and 14, we provide source-validation accuracies for each of the 5 datasets, for 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 4.1 of the main paper, extending training lengths with ***Auto-LR*** improves performance by 0.5% on average.

2. Dataset Visualizations

In Figures 1 (OfficeHome), 3 (DomainNet), 4 (VLCS), 5 (TerraIncognita), 6 (PACS), 7 (FMoW), and 8 (PCAM) 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 [18] Figure 1. This dataset focuses on household objects. The domain shifts are in low-level style mostly, and there is little spatial bias.

DomainNet [15] Figure 3. While the real domain is quite similar to what one might expect in ImageNet, the distribution shifts are quite substantial in other domains. Quick-

Cumul. study (#6 is full ERM++)							OfficeHome	PACS	VLCS	DomNet	TerraInc	Avg.
#	MPA	FD	WS	Auto-LR	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	✓	✓	✓	✗	✗	✓	72.6±0.1	88.8±0.1	77.0±0.1	48.6±0.0	49.3±0.3	67.3
5	✓	✓	✓	✓	✗	✓	72.6±0.1	88.8±0.1	**78.7**±0.0	48.6±0.0	49.2±0.3	67.6
6	✓	✓	✓	✓	✓	✓	**74.7**±0.0	**89.8**±0.3	78.0±0.1	**50.8**±0.0	**51.2**±0.3	**68.9**

Table 1. We present the overall cumulative study for ERM++. ERM++ corresponds to experiment 6. (1) ERM [9] baseline with unfrozen BN. (2) MPA: Model parameter averaging, which uniformly improves results. (3) FD: training on the full data. (4) WS: Warm-starting the classification layer especially improves OfficeHome and PACS. (5) Auto-LR: Learning the lr schedule, which ensures convergence improves performance by an additional half percent. (6) S.Init: Initializing the initial parameters to those trained with AugMix brings performance to state of the art.

	Painting	Clipart	Info	Real	Quickdraw	Sketch	Avg
Aug [10]	**57.3**	**68.8**	**25.6**	70.2	**17.1**	**59.8**	**49.8**
MV2 [17]	**57.3**	68.5	25.4	**70.9**	16.1	59.0	49.5

Table 2. **Model distillation's effect on DG:** 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. We can see that the distillation process, while dramatically improving ImageNet performance, only slightly changes DG performance.

draw and Infograph are particularly challenging, so the 1-3% gains of ERM++ on these domains is meaningful (Table 5).

VLCS [8]: Figure 4. Low-level statistics are quite similar between domains in this dataset, however spatial biases differ between domains. For example, 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. Lastly, there are many ambiguous images in the LabelMe domain (see Figure 9), raising questions about the validity of trying to improve performance on this dataset.

TerraIncognita [4]: Figure 5 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 [13] Figure 6. 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 [7, 12]: Figure 7. 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 webscraped data (see Section 5 of the main paper).

PCAM [3, 12]: Figure 8. The images are difficult to parse for an untrained human, but without tumors the images seems to have smaller and more dense cell structure. We also use this to test the generalization of ERM++ to perform on non-webscraped data (see Section 5 of the main paper) Images from Figure in [12].

2.1. Attention Tuning Visualization

We visualize attention tuning attention maps, and compare them to ERM++ w/out attention tuning and a pre-trained DINOv2 model in Figure 2. We find attention tuning can pick up discriminative, but occluded, features in samples where ERM++ w/out attention tuning.

3. Runtime Comparisons

As discussed in the main paper Section 4.4; ERM++ achieves higher predictive performance than competing methods MIRO [6] and DIWA [16] despite lower computational cost for training. The reason is reduced cost of hyper-parameter search; we use fixed hyper-parameters, borrowed from the DomainBed framework, (see Section 4.2 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 the training step cap in *Auto-lr*, 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

	Painting	Infograph	Quickdraw	Sketch	Real	Clipart	Avg
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 3. **Weight Space Regularization:** We experiment with 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 outperform ERM, **MPA** outperforms **SMPA**.

	art	clipart	product	real	avg
ERM [9]	63.1	51.9	77.2	78.1	67.6
ERM + SWAD [5]	66.1	57.7	78.4	80.2	70.6
DIWA [16]	69.2	59	81.7	82.2	72.8
ERM + MIRO + SWAD [6]	-	-	-	-	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 4. **OfficeHome:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. [6] 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.

better performance. In particular, this configuration represents Experiment 8 in Table 5 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.

4. Reproducibility

We provide code in a zip file along with this supplementary, and will open-source the code upon acceptance.

4.1. 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.

4.2. Training details

We follow the DomainBed [9] training procedure and add additional components from ERM++. In particular, we use the default hyper-parameters from DomainBed [9], *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 [11] optimizer with β and ϵ values set default values from Pytorch 1.12. We extend the training cap to 4x the initial learning when ***Auto-lr*** is used. 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 [9].

ViT Training Details: We follow a similar recipe for ViTs, with a few changes. First, we don't extend the training step cap by 4x on account of ViT's being over-parameterized and easy to overfit, relative to ResNet-50. Second, we use the LARS optimizer, which adjusts the learning rate according to the weight norm per-layer, and with a larger 1e-1 learning rate for the linear classifier during warmup ($1e-1$). The LARS optimizer decreases weight updates to stabilize training, and ViTs are unstable to train.

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 (2500 for ViT), and start *MPA* 100 steps after that. For the Specialist Model Parameter Averaging(*SMPA*) experiments (Table 9 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 re-computing of BN stats makes sure the averaged model has

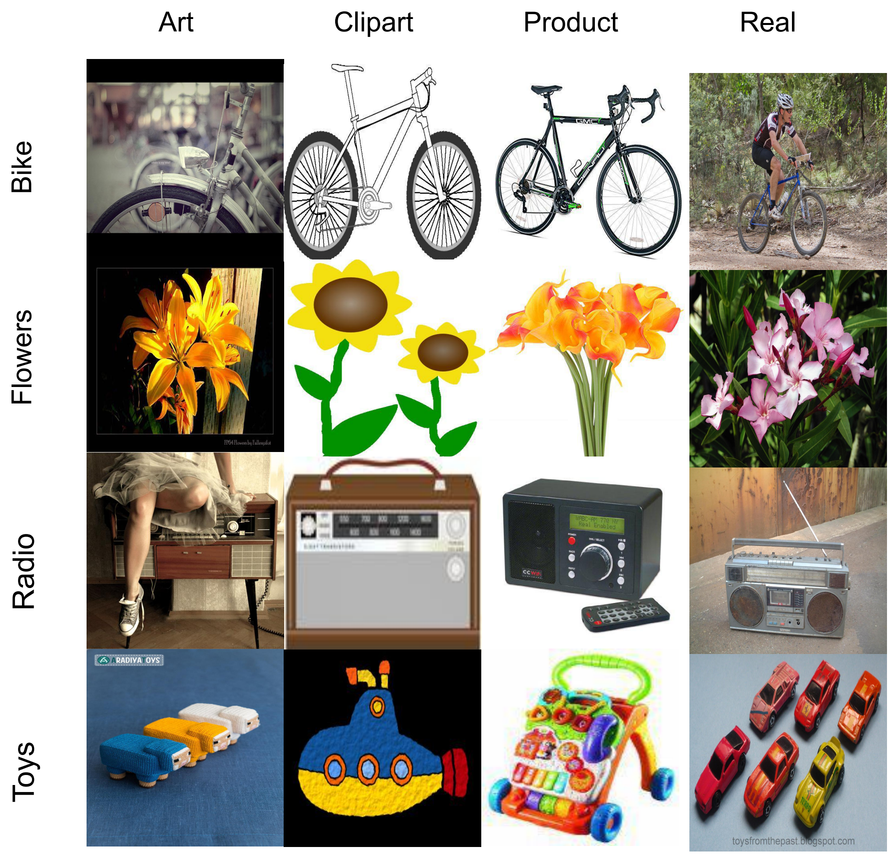

Figure 1. **OfficeHome**: Samples from the OfficeHome [18] dataset, from each domain and selected classes. The dataset focuses on household objects. The domain shifts are in low-level style mostly, and there is little spatial bias.

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.

Pseudocode for auto-lr:

In Figure 15, we show pseudo-code for 'Auto-lr' ERM++ component. The learning rate is decreased by a factor of 0.1 when the validation loss does not decrease. Training is stopped after the third consecutive non-decreasing validation loss

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 [19] [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

[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

	painting	clipart	info	real	quickdraw	sketch	avg
ERM [9]	50.1	63.0	21.2	63.7	13.9	52.9	44.0
ERM + SWAD [5]	53.5	66.0	22.4	65.8	16.1	55.5	46.5
DIWA [16]	55.4	66.2	23.3	68.7	16.5	56	47.7
ERM + MIRO + SWAD [6]	-	-	-	-	-	-	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 5. **DomainNet:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. [6] 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 4), 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 [9]. 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 [9]	97.7	**64.3**	73.4	74.6	77.3
ERM + SWAD [5]	98.8	63.3	**75.3**	**79.2**	79.1
DIWA [16]	**98.9**	62.4	73.9	78.9	78.6
ERM + MIRO + SWAD [5]	-	-	-	-	**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

Table 6. **VLCS:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. [6] 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 9), raising questions about the usefulness of trying to train on this domain.

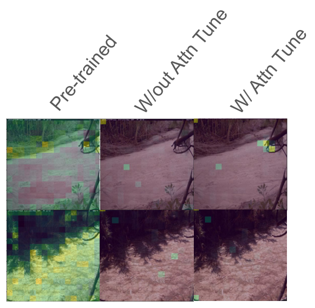

Figure 2. Examples of Attention Tuning Visualization of DINOv2 model. We average over all attention heads in the final attention block. On a pretrained model, attention is scattered. On both an attention tuned and full fine-tuned model, attention is more focused than with a pre-trained model. However, on some samples (representive samples pictured here) full fine-tuning misses discriminative but occluded animal features. On the top-right images, the attention tuning picks up the dog. In the bottom-right, the attention-tuned model picks up a tail in the lower-left corner.

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 implementation and borrow the lambda weights values from [6] 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 [16] algorithm due to computational reasons; we average the parameters of the three seeds of ERM++, with shared initialization of the linear classifier. The authors of DIWA show that about half of the performance boost comes from the first few models averaged (Figure 4 of [16]), therefore this is a reasonable approximation of the method.

SWAD Implementation: We directly follow the SWAD implementation and hyper-parameters from [5].

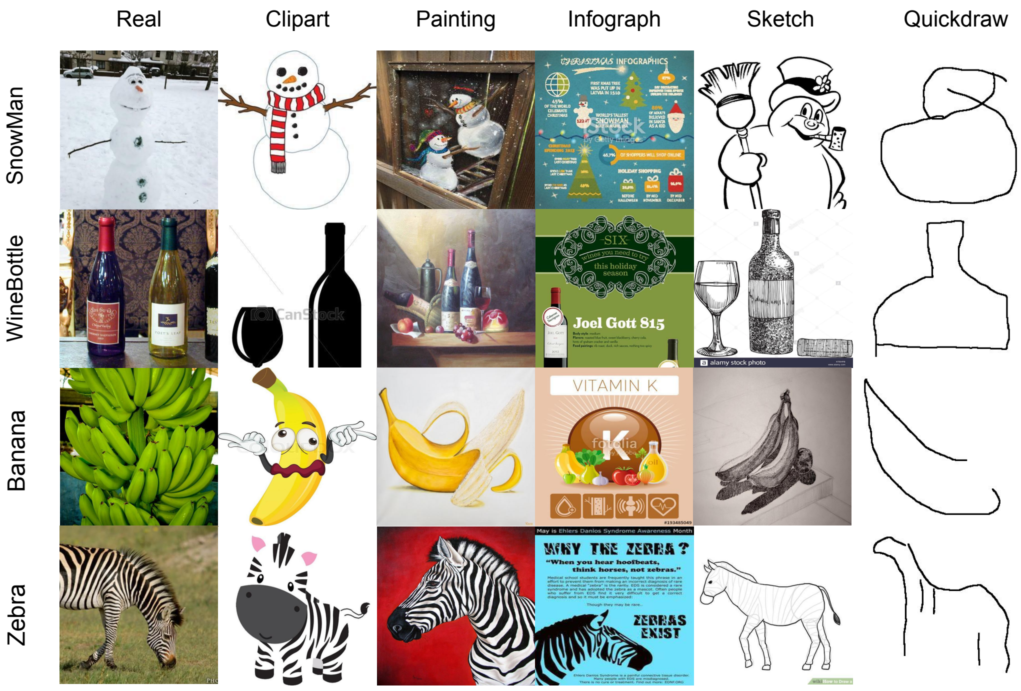

Figure 3. **DomainNet:** Samples from the DomainNet [15] dataset. While the real domain is quite similar to what one might expect in ImageNet, the distribution shifts are quite substantial in other domains. Quickdraw and Infograph are particularly challenging, so the 1-3% gains of ERM++ on these domains is meaningful (Table 5). While most domains contain primarily shifts in low level statistics (for example, real to painting), Infograph also has many non-centered objects.

	Loc. 100	Loc. 38	Loc. 43	Loc. 46	Average
ERM [9]	54.3	42.5	55.6	38.8	47.8
ERM + SWAD [5]	55.4	44.9	59.7	39.9	50.0
DIWA [16]	**57.2**	50.1	60.3	39.8	51.9
ERM + MIRO + SWAD [6]	-	-	-	-	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 7. **TerraIncognita:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. [6] 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%.

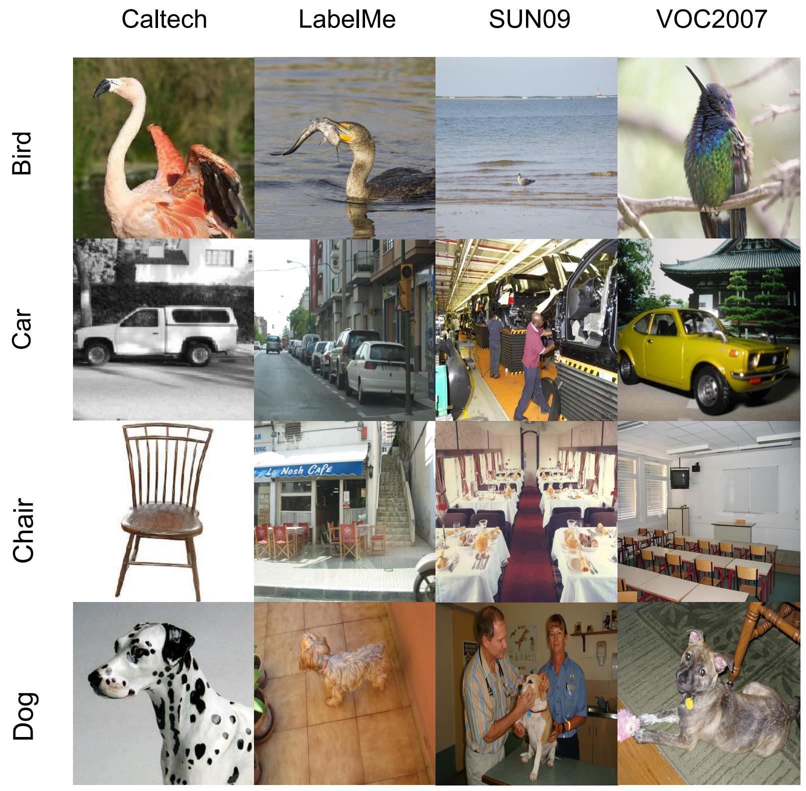

Figure 4. **VLCS:** Sample from the VLCS dataset [8] The low-level statistics are quite similar between domains, however spatial biases differ between domains. Caltech objects are quite centered, while other domains do not have this trait. For example the LabelMe domain has cars along the side of the image, and there are many chairs in the VOC2007 domain. Furthermore, in some cases the size of the objects differs dramatically. Finally, there are many ambiguous images in the LabelMe domain (see Figure 9), raising questions about the usefulness of trying to train on this domain.

	art_painting	cartoon	photo	sketch	avg
ERM [9]	84.7	80.8	97.2	79.3	84.2
ERM + SWAD [5]	89.3	83.4	97.3	82.5	88.1
DIWA [16]	90.6	83.4	98.2	83.8	89
ERM + MIRO + SWAD [6]	-	-	-	-	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 8. **PACS:** Per-domain top-1 accuracy against reported results of recent top-performing methods SWAD, DIWA, and MIRO. [6] 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.

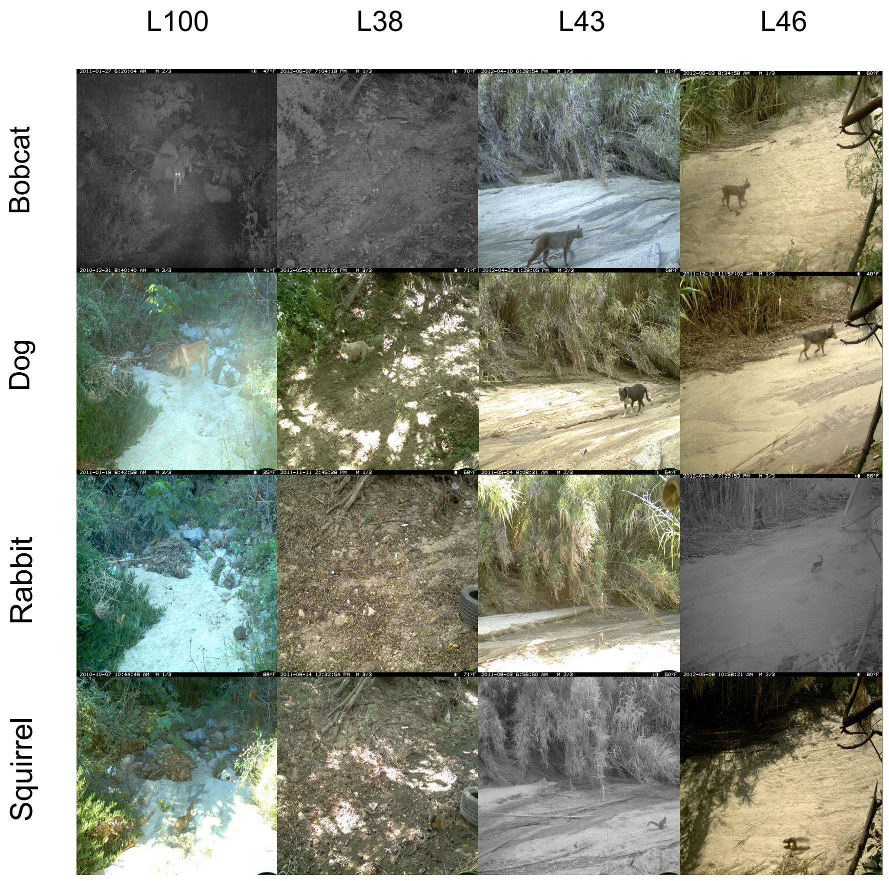

Figure 5. **TerraIncognita**: Samples from the TerraIncognita [4] dataset, from each domain and selected classes. The background stays consistent, and the animal object frequently takes up a small portion of the frame. At night the images are black-and-white. This dataset matches realistic deployment scenarios well.

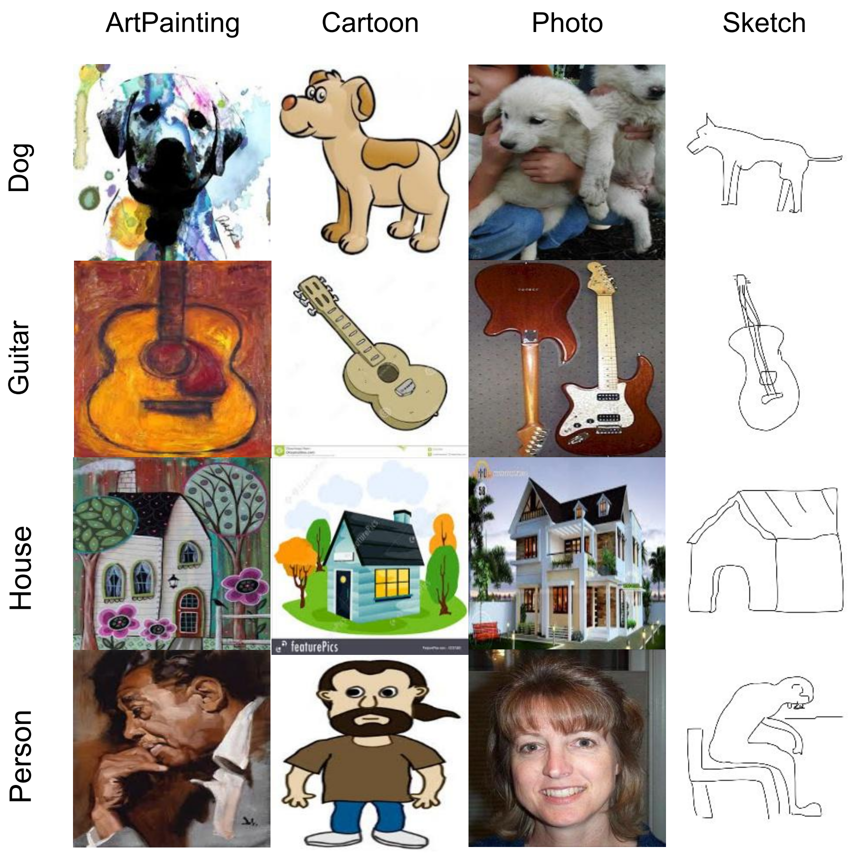

Figure 6. **PACS:** Samples from the PACS dataset [13], from each domain and selected classes. The subjects tend to be centered, and the sketches are more realistic than the quickdraw setting in DomainNet. Though the domians are similar to that of DomainNet, PACS has fewer than 10000 samples compared to 586000 of DomainNet. Therefore PACS tests the capabilities of ERM++ on smaller data.

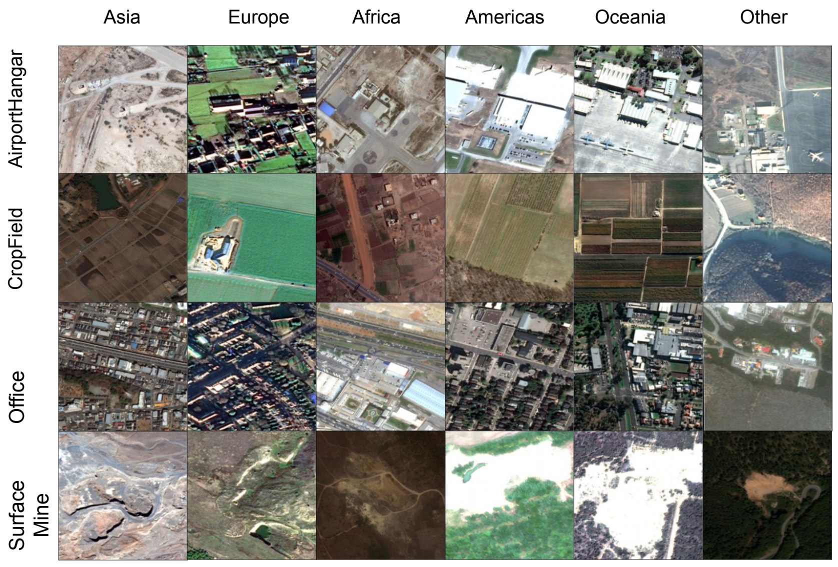

Figure 7. **FMoW:** Samples from the FMoW [7, 12] dataset, from each domain and selected classes. The images differ in region but also in resolution and scale. The distribution shift between FMoW and the pretraining data is large, therefore FmoW represents the ability of ERM++ to perform on non web-scraped data (see Section 5.4 of the main paper).

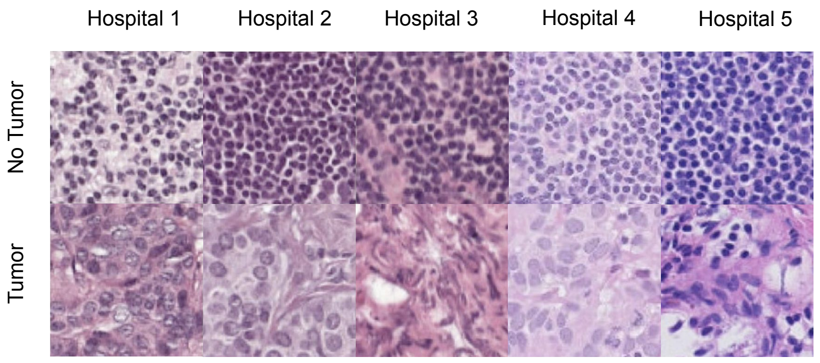

Figure 8. **PCAM:** Samples from the PatchCamelyon [3, 12] dataset, from each domain and both classes. The images are difficult to parse for an untrained human, but without tumors the images seems to have smaller and more dense cell structure. Images from [12] paper figure.

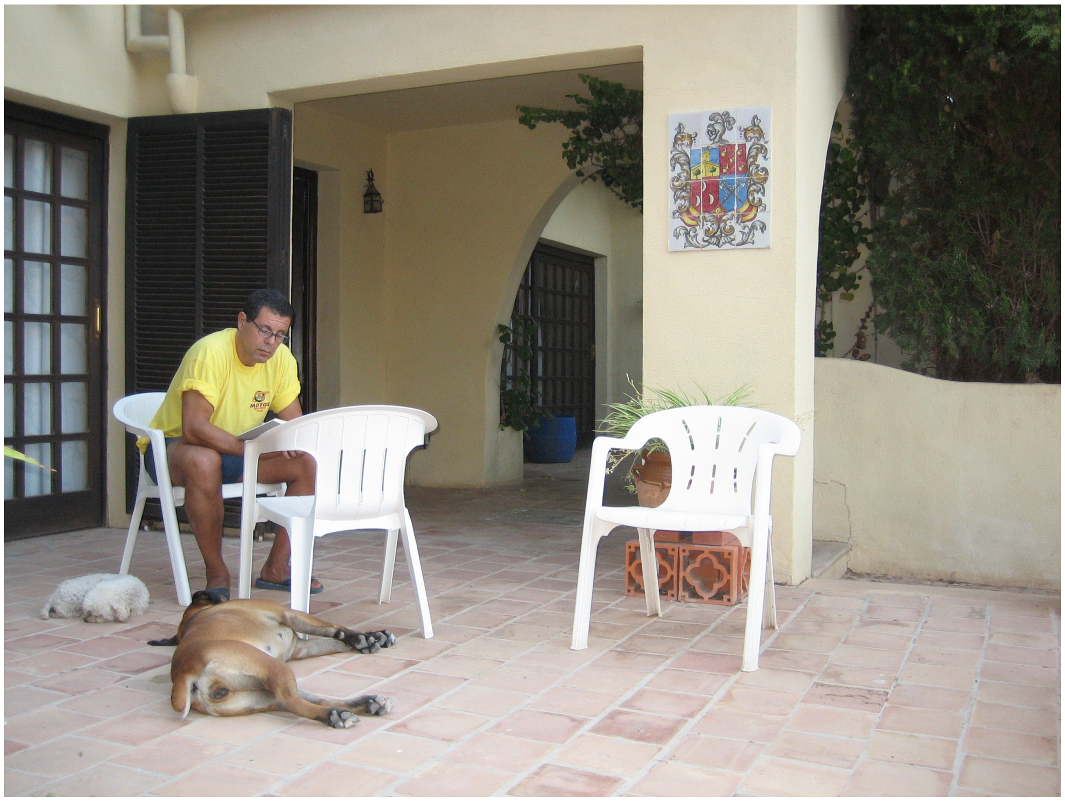

Figure 9. **Sample from LabelMe Domain in VLCS:** Is this a dog, person, or chair? Many samples in the LabelMe domain of VLCS are ambiguous but assigned a label (in this case, dog). This raises questions about the usefulness of training on this domain.

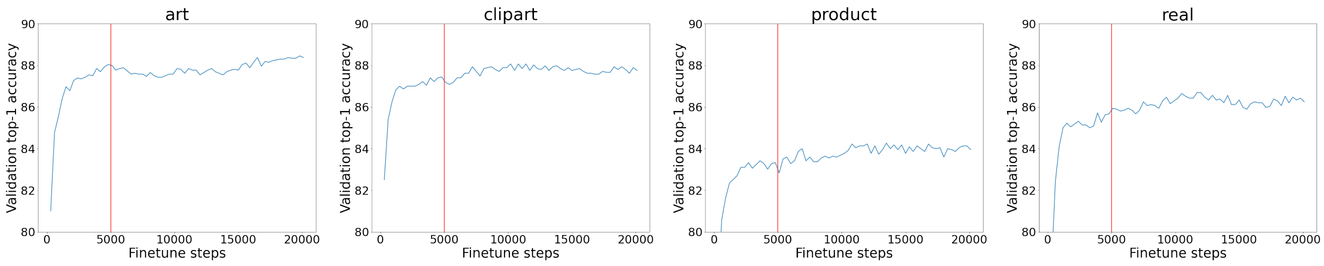

Figure 10. **OfficeHome:** Source validation accuracies. The validation accuracy saturates by 20000 steps. Training length used in prior works is denoted as a red line, and the training is not yet converged.

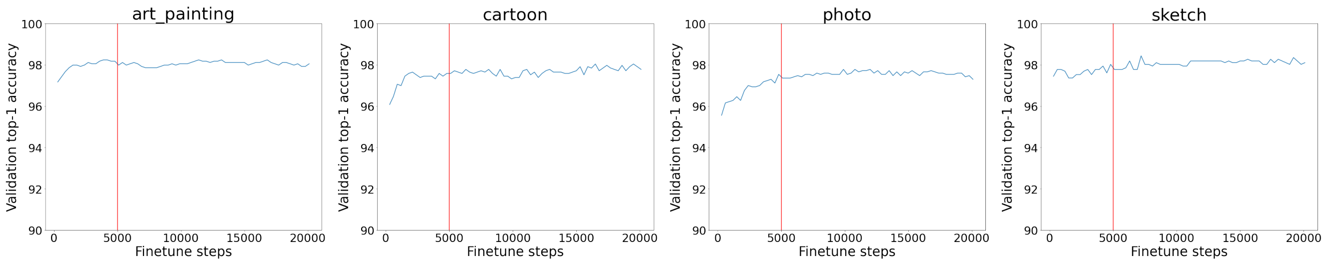

Figure 11. **PACS**: Source validation accuracies. The validation accuracy saturates by 20000 steps. Training length used in prior works is denoted as a red line, and the training is not yet converged.

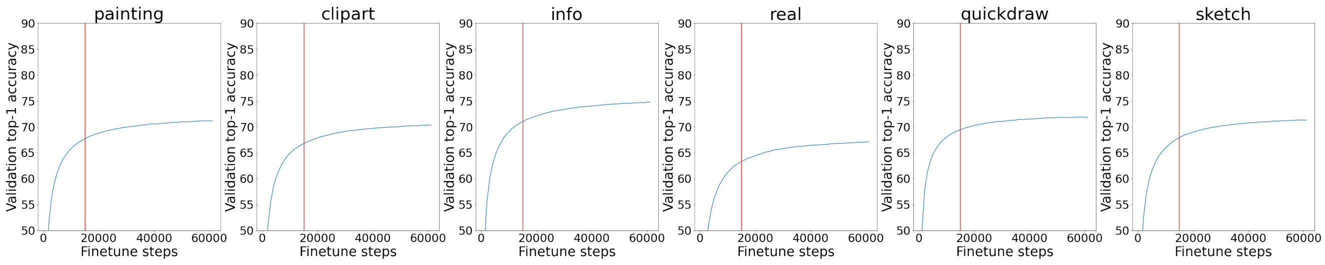

Figure 12. **DomainNet:** Source validation accuracies. The validation accuracy saturates by 60000 steps. Training length used in prior works is denoted as a red line, and the training is not yet converged.

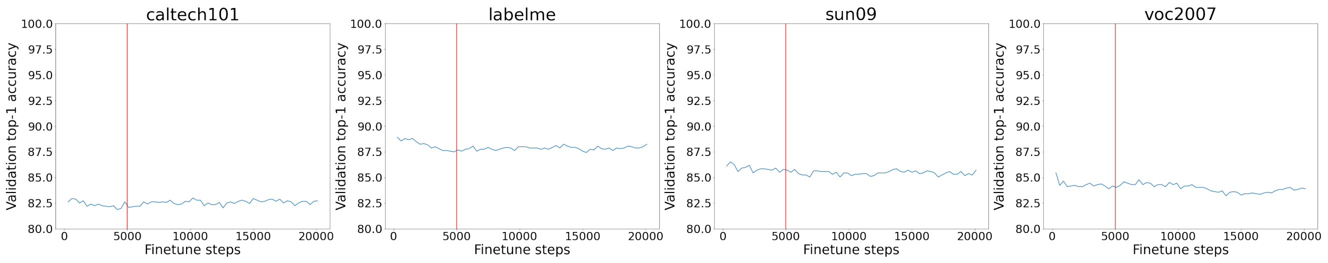

Figure 13. **VLCS:** Source validation accuracies. The validation accuracy saturates by 20000 steps. Training length used in prior works is denoted as a red line. In the case of VLCS, it seems like longer training is not so helpful, and this shows the need for ***Auto-lr***

.

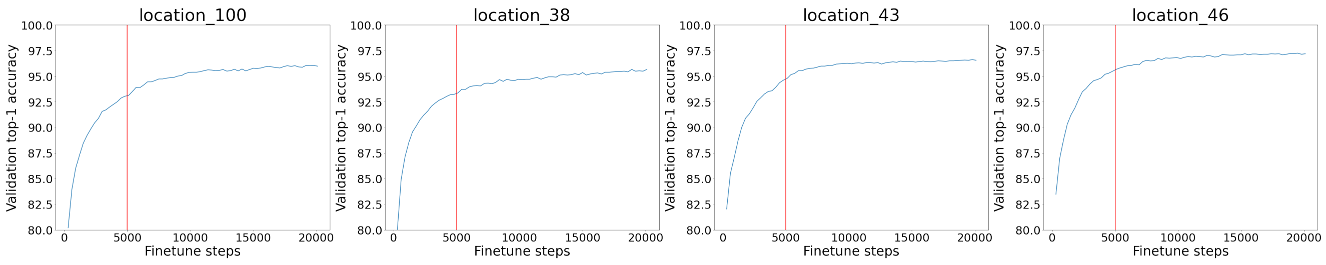

Figure 14. **TerraIncognita:** Source validation accuracies. The validation accuracy saturates by 20000 steps. Training length used in prior works is denoted as a red line, and the training is not yet converged.

```
import torch
import torch.optim as optim
import torch.nn as nn


# Initialize variables for learning rate decay
lr_decay_factor = 0.1
lr = initial_lr
max_decay_steps = 3
decay_count = 0

# Training loop
for step in range(num_steps):
    # Your training code here

    # Validation
    if (step % eval_num_steps == 0)
        model.eval()
        validation_loss = compute_validation_loss(model, validation_data, criterion)

        # Check if validation loss is not decreasing
        if validation_loss >= previous_validation_loss:
            decay_count += 1
            if decay_count >= max_decay_steps:
                break
            else:
                # Decay learning rate
                lr *= lr_decay_factor
                for param_group in optimizer.param_groups:
                    param_group['lr'] = lr

        # Update previous validation loss for the next iteration
            previous_validation_loss = validation_loss

```

Figure 15. **Learning Rate Decay with Auto-lr**. The learning rate is decreased by a factor of 0.1 when the validation loss does not decrease. Training is stopped after the third consecutive non-decreasing validation loss.

References

[1] Simone Angarano, Mauro Martini, Francesco Salvetti, Vittorio Mazzia, and Marcello Chiaberge. Back-to-bones: Rediscovering the role of backbones in domain generalization. *arXiv preprint arXiv:2209.01121*, 2022. 1

[2] Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. Ensemble of averages: Improving model selection and boosting performance in domain generalization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 1

[3] Peter Bandi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermsen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, et al. From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge. *IEEE Transactions on Medical Imaging*, 2018. 2, 11

[4] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018. 2, 8

[5] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. In *Advances in Neural Information Processing Systems*, volume 34, pages 22405–22418, 2021. 1, 3, 5, 6, 7

[6] Junbum Cha, Kyungjae Lee, Sungrae Park, and Sanghyuk Chun. 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*, pages 440–457. Springer, 2022. 1, 2, 3, 5, 6, 7

[7] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 10

[8] 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*, pages 1657–1664, 2013. 2, 7

[9] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 5, 6, 7

[10] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 2

[11] Diederik P Kingma and Jimmy Ba. dam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 3

[12] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021. 2, 10, 11

[13] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017. 2, 9

[14] Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*, 2022. 1

[15] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019. 1, 6

[16] Alexandre Rame, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, patrick gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. In *Advances in Neural Information Processing Systems*, 2022. 1, 2, 3, 5, 6, 7

[17] Zhiqiang Shen and Marios Savvides. Meal v2: Boosting vanilla resnet-50 to 80%+ top-1 accuracy on imagenet without tricks. *arXiv preprint arXiv:2009.08453*, 2020. 2

[18] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017. 1, 4

[19] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 4