Ensemble of Averages: Improving Model Selection and Boosting Performance in Domain Generalization

Devansh Arpit [1] **Huan Wang** [1] **Yingbo Zhou** [1] **Caiming Xiong** [1]

Abstract

In Domain Generalization (DG) settings, models trained on a given set of training domains have notoriously chaotic performance on distribution shifted test domains, and stochasticity in optimization (e.g. seed) plays a big role. This makes deep learning models unreliable in real world settings. We first show that a simple protocol for averaging model parameters along the optimization path, starting early during training, both significantly boosts domain generalization and diminishes the impact of stochasticity by improving the rank correlation between the in-domain validation accuracy and out-domain test accuracy, which is crucial for reliable model selection. Next, we show that an ensemble of independently trained models also has a chaotic behavior in the DG setting. Taking advantage of our observation, we show that instead of ensembling unaveraged models, ensembling moving average models (EoA) from different runs does increase stability and further boosts performance. On the DomainBed benchmark, when using a ResNet-50 pre-trained on ImageNet, this ensemble of averages achieves 88.6% on PACS, 79.1% on VLCS, 72.5% on Office-Home, 52.3% on TerraIncognita, and 47.4% on DomainNet, an average of 68.0%, beating ERM (w/o model averaging) by $\sim 4\%$. We also evaluate a model that is pre-trained on a larger dataset, where we show EoA achieves an average accuracy of 72.7%, beating its corresponding ERM baseline by 5%.

1. Introduction

Domain generalization (DG, Blanchard et al. (2011)) aims at learning predictors that generalize well on data sampled from test distributions that are different from the training distribution. Currently, deep learning models have been shown to be poor at this form of generalization (D'Amour et al., 2020), and excel primarily in the IID setting (Zhou et al., 2021a).

While a number of algorithms have been proposed to mitigate this problem (cf Zhou et al. (2021a) for a survey), Gulrajani & Lopez-Paz (2020) demonstrate that models trained using empirical risk minimization (ERM, Vapnik & Vapnik (1998)) along with proper model selection (i.e. early stopping using validation set), using a subset of data from all the training domains, largely match or even outperform the performance of most existing domain generalization algorithms. This suggests that model selection plays an important role in domain generalization. Despite its importance, *there has not been much investigation into the reliability of model selection*. As we demonstrate in Figure 1, the out-domain performance varies greatly along the optimization trajectory of a model during training, even though the in-domain performance does not. This instability therefore hurts the reliability of model selection, and can become a problem in realistic settings where test domain data is unavailable, because it causes the rank correlation between in-domain validation accuracy and out-domain test accuracy to be weak.

In this paper, we first investigate a simple protocol for model averaging that both boosts domain generalization within the ERM framework, and mitigates performance instability of deep models on out-domain data, specifically with respect to in-domain validation data. This makes model selection more reliable. Next, we show that even an ensemble of independently trained models suffers from the said instability. Taking advantage of our observation, we show that ensembling moving average models is able to mitigate this problem, and further boosts performance, making it a better choice for practical scenarios. Note that we do not claim that model averaging or ensembling can fully solve the problem of domain generalization. Our findings are as follows:

1. Moving average models have a much more stable out-domain performance compared to unaveraged models (see Figure 1 for qualitative illustration). The rank correlation between in-domain validation accuracy and out-domain test accuracy along the optimization trajectory is also significantly better for moving average models (see Table 1). Thus

[1]Salesforce Research, USA. Correspondence to: Devansh Arpit <devansharpit@gmail.com>.

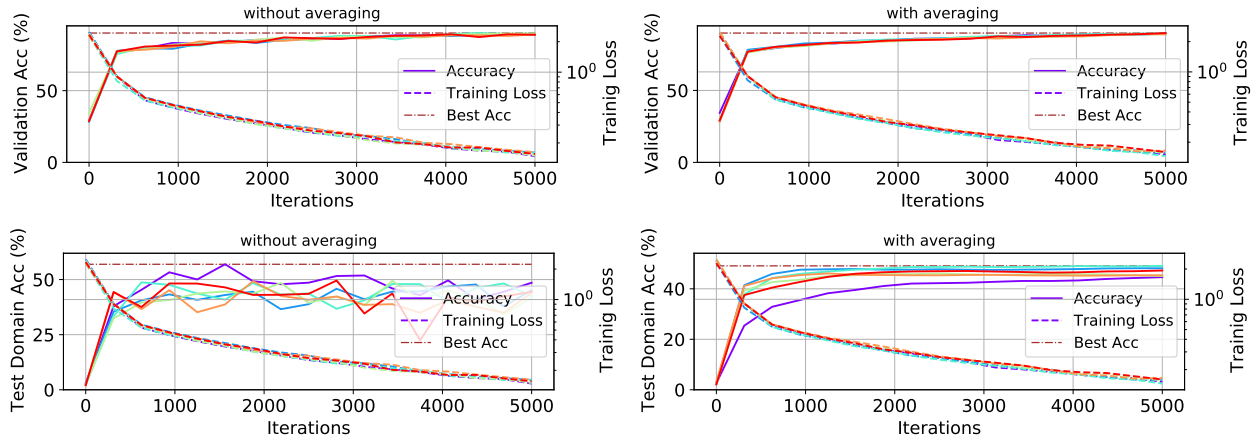

Figure 1. Model averaging improves out-domain performance *stability*. **Left**: In-domain validation accuracy and out-domain test accuracy during training of models using ERM. **Right**: Same as left, except validation and test predictions are made using a simple moving average of the model being optimized, along its optimization path. **Details**: The plots are for the TerraIncognita dataset with domain L38 used as the test domain, and others as training/validation data, and ResNet-50. Solid lines denote accuracy, dashed lines denote training loss, and dash-dot lines denote best accuracy achieved during training and all runs (for reference). Each color denotes a different run with a different random seed and training/validation split. **Gist**: Model averaging reduces out-domain performance instability, and makes the test curves correlate better with the validation curves, making model selection using in-domain validation set more reliable during optimization. We see a similar pattern when using ensemble of models, with and without model averaging, in Figure 5.

model selection is more reliable when we make predictions on validation set using the moving average models.

2. Starting averaging *early* during training, boosts domain generalization (Figure 2).

3. Unfortunately, the rank correlation is poor between validation and test accuracy of independently trained models (Figure 6 in appendix). An implication of this is that it is difficult to discover the best model (for out-domain performance) from a pool of independently trained models, based only on their in-domain validation performance.

4. The frequency of model averaging does not have a significant impact on performance, unless sampling is done at too large intervals (Figure 3).

5. Taking advantage of our simple model averaging protocol (section 2.2), we find that an ensemble of moving average (EoA) models outperforms an ensemble of unaveraged models (Table 3). We also show ablation analysis that the rank correlation between in-domain validation performance and out-domain test performance is also better for the ensemble of average models (Table 2).

6. For benchmarking, we experiment with two different pre-trained models as initializations for DG training, one pre-trained on ImageNet, and the other involving semi-weakly supervised pre-training on IG-1B targeted dataset and ImageNet. While the latter improves the baseline performance, in both cases, using EoA consistently yields around $4\%-5\%$ test accuracy gain on average (Table 3).

2. Model Averaging

2.1. Terminology

Online Model: For a given supervised learning objective function, let $f_\theta(.)$ denote the deep network being optimized using gradient based optimizer, where θ denotes the parameters of this model. We refer to f_θ as the *online model*, or *unaveraged model*. The output of $f_\theta(.)$ is a vector of K logits corresponding to the K classes in the supervised task.

Moving Average (MA) Model: While the online model is being trained, we maintain a moving average of the online model's parameters. This process is sometime referred to as *iterate averaging* in existing literature. The deep network whose parameters are set to be this moving average is referred to as the *moving average model*, or more specifically *simple moving average (SMA) model* because of its use in our work. We denote the parameters of this model by $\hat{\theta}$.

2.2. Model Averaging Protocol

We use a simple moving average (SMA) of the online model. Instead of calculating the moving average starting from initialization (as done in Polyak-Ruppert averaging), we instead start after a certain number of iterations t_0 during training (tail averaging), and maintain the moving average until the end of training. As we discuss in the next section, t_0 is chosen to be close, but not equal to the initialization.

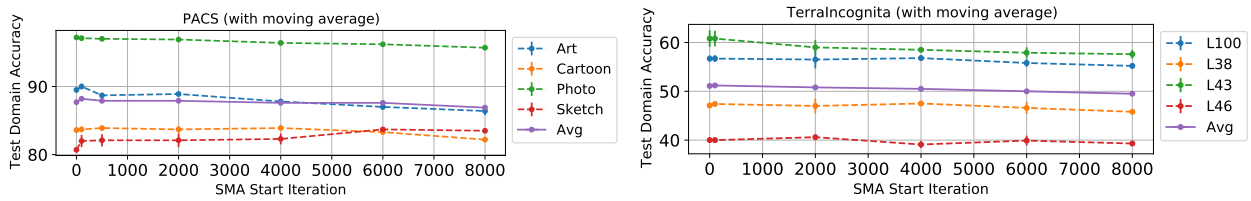

Figure 2. The impact of iteration t_0 at which we start simple moving averaging as described in Eq. 1, on the domain generalization performance for PACS and TerraIncognita datasets. The dominant pattern across all the experiments suggests that starting averaging earlier yields a stronger boost in performance.

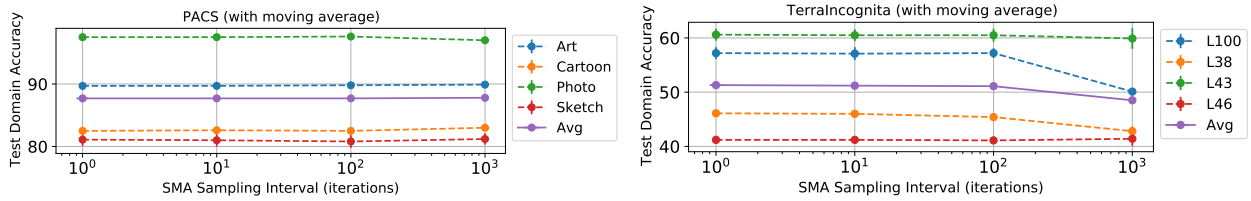

Figure 3. The impact of the frequency (number of iterations), at which model states are sampled for computing the simple moving average (SMA), on the domain generalization performance for PACS and TerraIncognita datasets. Broadly, we find that the frequency of sampling does not have a major influence on performance unless the sampling interval is too large: performance drops significantly on TerraIncognita only when the frequency is set to sampling every 1000 iterations.

At any iteration t, we denote:

$$\hat{\theta}_t = \begin{cases} \theta_t, & \text{if } t \leq t_0 \\ \frac{t-t_0}{t-t_0+1} \cdot \hat{\theta}_{t-1} + \frac{1}{t-t_0+1} \cdot \theta_t, & \text{otherwise} \end{cases} \quad (1)$$

where θ_t is the online model's state at iteration t. Further, at iteration t, if we need to calculate validation performance, we use $\hat{\theta}_t$ to do so, and not θ_t. As we show in the next section, the benefit of doing so is that the rank correlation between in-domain validation accuracy and out-domain test accuracy is significantly better when predictions are made using $\hat{\theta}_t$. This makes model selection more reliable for domain generalization. Finally, for a given run, model selection selects $\hat{\theta}_{t^*}$ for making test set predictions, such that $\hat{\theta}_{t^*}$ achieves the best validation performance. We discuss some theoretical perspectives on why model averaging can help domain generalization in section 6.1. A Pytorch code is provided in appendix section C.

3. Analysis

Experimentation Details: We use the training protocol described in Gulrajani & Lopez-Paz (2020) with minor changes: we use a smaller hyper-parameter search space for feasibility, and train on DomainNet dataset for $15,000$ iterations instead of $5,000$ similar to Cha et al. (2021), because its training loss is quite high. Unless specified otherwise, we use the said protocol in all the experiments. For model selection, we use the *training-domain validation set* protocol in Gulrajani & Lopez-Paz (2020), where the average out-domain test performance is reported across all runs. For more details, see section A in the appendix.

Dataset Details: We use a subset of the DomainBed benchmark: PACS dataset (4 domains, 7 classes, and 9,991 images), TerraIncognita dataset (4 domains, 10 classes, and 24,788 images) VLCS dataset (4 domains, 5 classes, and 10,729 images), OfficeHome dataset (4 domains, 65 classes, and 15,588 images), and DomainNet dataset (6 domains, 345 classes, and 586,575 images).

3.1. Start Iteration

We investigate how domain generalization performance is impacted by the choice of iteration when we start model averaging. In this section, we simply refer to it as start iteration, which should not be confused with the start of the training process. For experiments, we use the PACS and TerraIncognita datasets. To investigate a wide range of start and end iterations, for all experiments in this section, we train models for $10,000$ iterations.

We consider starting model averaging from iterations in $\{0, 100, 500, 2000, 4000, 6000, 8000\}$. We plot the performance in Figure 2 for each value. We find that the test performance, averaged across all the domains for both datasets, decreases if we start model averaging later during the training. It seems that starting averaging soon after the training starts yields the best performance. We believe using the initialization state in model averaging causes a slight dip in performance because loss is initially high. Based on these

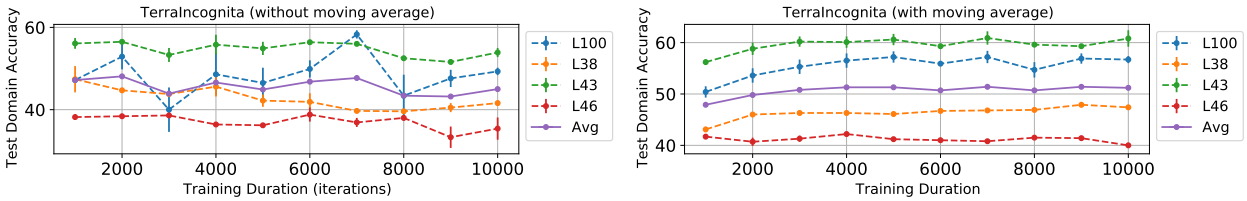

Figure 4. Qualitatively accessing the reliability of model selection while varying the training duration. For each training duration, the out-domain test accuracy is calculated using model selection over the in-domain validation data. Not using model averaging leads to unreliable model selection as evident in the instability of the out-domain performance. Model averaging is able to overcome this instability.

experiments, instead of tuning start iteration as a hyper-parameter, *we arbitrarily choose 100 as the start iteration for the remaining experiments in this paper*. This choice of starting averaging later during training is call *tail averaging*, and the theoretical motivation for this choice are discussed in more detail in section 6.1.

3.2. Averaging Frequency

When performing simple model averaging described in Eq. 1, instead of averaging iterates from every iteration, we can alternatively sample iterates at a larger interval. We study the impact of averaging frequency on out-domain test performance. Once again, we use the PACS and TerraIncognita datasets. We train models for $10,000$ iterations, and sample iterates at intervals in $\{1, 10, 100, 1000\}$ iterations. Test accuracy is once again computed using the protocol of Gulrajani & Lopez-Paz (2020) for each case. The performance as a function of the iterate sampling interval used in SMA is shown in Figure 3. Broadly, we find that the frequency of sampling does not have a major impact on performance unless the sampling interval is too large, which happens in the case of TerraIncognita, where performance drops significantly when the sampling interval is set to 1000.

3.3. Instability Reduction: Qualitative Analysis

Here we try to qualitatively study the robustness of model selection using in-domain validation set, on out-domain performance. To do so, consider the ideal scenario where the in-domain validation performance correlates well with the out-domain performance. In this case, training longer should not be a problem in general, because if the model starts overfitting beyond a certain point, model selection can take care of it. In such a situation, we would expect the out-domain performance to either improve with longer training, or remain stable.

We use TerraIncognita dataset for this experiment. We consider training duration to be $1,000$ to $10,000$ iterations, at intervals of $1,000$. We plot the performance in Figure 4 for online model (left) and moving average model (right). We find that the performance of moving average models is more stable compared to online models, suggesting that model selection is more reliable when using moving average models. Figure 9 in the appendix shows the training loss, in-domain validation accuracy and out-domain test accuracy for all the runs used in the above experiment. It shows that the out-domain test performance is unstable during optimization without model averaging, which causes problem for model selection using the in-domain validation performance, as is evident in the above experiment.

3.4. Instability Reduction: Rank Correlation

Rank correlation metrics aim to quantify the degree to which an increase in one random variable's value is consistent with an increase in the other random variable's value. Therefore, they are better suited for studying the relationship between the in-domain and out-domain performance for the purpose of model selection because we select the best model during an optimization based on ranking the validation performance. We consider Spearman correlation in our experiments. Its value vary between -1 and $+1$, where -1 implies the ranking of the two random variables are exactly the reverse of each other, and $+1$ implies the ranking of the two random variables are exactly the same as each other. A value of 0 implies there is no relationship between the two variables.

Within-run rank correlation: Compared to the previous subsection, we now perform a more direct study of the reliability of model selection for domain generalization when using online models vs moving average models, using rank correlation. To do so, we train models on a dataset, both with and without model averaging, and compute Spearman correlation between the in-domain validation accuracy and out-domain test accuracy during the training process. For clarity of the procedure, consider the PACS dataset which has 4 domains. Using the training-evaluation protocol in Gulrajani & Lopez-Paz (2020), multiple runs are performed such that for each run, one of the 4 domains is considered the test domain while the remaining are used as training/validation data, and different seeds and hyper-parameter values are used. For each such run, we track the in-domain validation accuracy and out-domain test accuracy during training

Table 1. Spearman correlation (closer to 1 is better) between within-run in-domain validation accuracy and out-domain test accuracy on the multiple datasets in the DomainBed benchmark. In most cases, using model averaging results in a significantly better rank correlation, which makes model selection more reliable.

PACS	Without averaging	With averaging
Art	0.31 ± 0.04	$\mathbf{0.62 \pm 0.04}$
Cartoon	0.25 ± 0.10	$\mathbf{0.52 \pm 0.03}$
Photo	$\mathbf{0.09 \pm 0.07}$	-0.38 ± 0.15
Sketch	0.24 ± 0.06	$\mathbf{0.53 \pm 0.06}$
TerraIncognita	**Without averaging**	**With averaging**
L100	0.21 ± 0.07	$\mathbf{0.90 \pm 0.05}$
L38	0.12 ± 0.13	$\mathbf{0.83 \pm 0.05}$
L43	0.30 ± 0.06	$\mathbf{0.67 \pm 0.18}$
L46	0.03 ± 0.11	$\mathbf{0.52 \pm 0.14}$
VLCS	**Without averaging**	**With averaging**
Caltech101	$\mathbf{0.21 \pm 0.10}$	0.16 ± 0.15
LabelMe	$\mathbf{0.30 \pm 0.08}$	0.02 ± 0.14
Sun09	0.27 ± 0.12	$\mathbf{0.32 \pm 0.11}$
VOC2007	0.17 ± 0.11	$\mathbf{0.38 \pm 0.05}$
OfficeHome	**Without averaging**	**With averaging**
Art	0.05 ± 0.11	$\mathbf{0.80 \pm 0.04}$
Clipart	0.33 ± 0.04	$\mathbf{0.84 \pm 0.04}$
Product	0.61 ± 0.04	$\mathbf{0.80 \pm 0.04}$
RealWorld	0.41 ± 0.06	$\mathbf{0.74 \pm 0.04}$
DomainNet	**Without averaging**	**With averaging**
Clip	0.96 ± 0.01	$\mathbf{1 \pm 0}$
Info	0.80 ± 0.05	$\mathbf{1 \pm 0}$
Paint	0.87 ± 0.02	$\mathbf{1 \pm 0}$
Quick	0.65 ± 0.04	$\mathbf{1 \pm 0}$
Real	0.91 ± 0.01	$\mathbf{1 \pm 0}$
Sketch	0.82 ± 0.04	$\mathbf{1 \pm 0}$

at regular intervals. Each time, we get a tuple of these two values, which over the entire training duration yields a list of such tuples. We calculate the Spearman correlation for this list of (validation, test) accuracy. Finally, since there are multiple runs where a given domain acts as the test domain, we calculate the mean and standard error of these values over these runs.

Using the above procedure, the rank correlations are shown in Table 1 for the PACS, VLCS, OfficeHome, TerraIncognita and DomainNet datasets. We find that in majority of the cases, using model averaging results in a significantly better rank correlation compared to using the online model[1]. These experiments therefore strongly suggest that the reliability of model selection is significantly higher within a run when using model averaging.

Cross-run rank correlation: There is another way in which it makes sense to study the rank correlation between validation and test performance. Suppose we set one of the domains of PACS as our test domain, and the remaining as training/validation data, and perform multiple independent runs with different seeds/hyper-parameter values. At each iteration during training, we can gather the tuple (validation, test) accuracy for each of these runs, and then study the rank correlation between them. The utility of this perspective is to assess the reliability of model selection in terms of selecting a single model across multiple independently trained models, based on their validation performance. We study this rank correlation for PACS and TerraIncognita datasets. The results are shown in Figure 6 in the appendix. We find that the cross-run rank correlations are poor (not consistently close to 1) for both online model (without averaging) and moving average model. This implies that in-domain validation performance based model selection is not a reliable approach for selecting a model from a pool of multiple independently trained models.

4. Ensemble of Averages (EoA)

Gulrajani & Lopez-Paz (2020) propose a rigorous framework for evaluation in the domain generalization setting which accounts for randomness due to seed and hyper-parameter values, and recommend reporting the average test accuracy over all the runs computed using a model selection criteria. However, in practice, it is desirable to have a single predictor that has a high accuracy. Ensembles combine predictions from multiple models, and is a well known approach for achieving this goal (Dietterich, 2000) by exploiting function diversity (Fort et al., 2019). However, as we show, even ensembles suffer from instability in the domain generalization setting. Building on the observations of the previous section, we investigate the behavior of ensemble of moving average models and find that it mitigates this issue. We begin by describing the EoA protocol below.

EoA Protocol: We perform experiments with ensemble of multiple independently trained models (i.e., with different hyper-parameters and seeds). When each of these models are moving average models from their corresponding runs, we refer to this ensemble in short as the *ensemble of averages (EoA)*. Identical to how we make predictions for ensembles (specifically the bagging method Breiman (1996)), the class $\hat{y}$ predicted by an EoA for an input $\mathbf{x}$ is given by the formula:

$$\hat{y} = \arg\max_k Softmax(\frac{1}{E}\sum_{i=1}^{E} f_{\hat{\theta}_i}(\mathbf{x}))^{(k)} \qquad (2)$$

[1]To explain the negative correlation on PACS, we note that test performance on the Photo domain converges to $\sim 99\%$ soon after training begins (because the photo domain is close to the ImageNet dataset, which is the dataset on which the model is pre-trained on). Therefore the correlation between validation and test accuracy is largely noise. See Fig. 8 in appendix.

where E is the total number of models in the ensemble, $\hat{\theta}_i$ denotes the parameters of the i^{th} moving average model, and the super-script $(.)^{(k)}$ denotes the k^{th} element of the vector argument. Finally, the state $\hat{\theta}_i$ of the i^{th} moving average model used in the ensemble is selected from its corresponding run using its in-domain validation set performance (described in section 2.2). We now investigate the behavior of EoA compared with ensembles of online models on domain generalization tasks.

4.1. Analysis

Qualitative visualization: For the purpose of contrasting the behavior of traditional ensembles vs ensemble of averages, we begin by qualitatively studying the stability of out-domain performance of these two ensembling techniques during the training process. To do so, we use the TerraIncognita dataset, and fix one of its domains has the test domain while using the others as training/validation data. We then train 6 different models independently for $5,000$ iterations with different seeds, hyper-parameters and training-validation splits identical to the Gulrajani & Lopez-Paz (2020) protocol. We also maintain moving average models corresponding to each of these 6 models. At every 300 iterations, we form an ensemble of the 6 online models from their corresponding runs and compute the out-domain test accuracy. Since, each run has a different training-validation split, we calculate the mean validation accuracy of each of these online models at that iteration. We follow an identical procedure for the moving average models and plot these performances in Figure 5. We find that the ensemble of averages has a better stability on out-domain test set compared to the ensemble of online models.

For clarity, note that this procedure for calculating test accuracy at regular intervals is different from what we proposed earlier for EoA for practical purposes. This experiment is only meant to highlight the fact that making predictions on out-domain data using an ensemble of online models suffers from instability along the optimization trajectory, while an ensemble of averages mitigates this issue. For plots on other domains of TerraIncognita, see Figure 7 in the appendix.

Rank correlation: We now measure the rank correlation between in-domain validation accuracy and out-domain test accuracy for a quantitative evaluation. The details of the metric and motivations behind this experiment are same as those described in section 3.4. Here we use the same experimental setup described above in the qualitative visualization experiment for ensembles. But in addition, we also conduct experiments on VLCS, OfficeHome and DomainNet datasets. The results are shown in Table 2. We find that in majority of the cases, using EoA results in a significantly better rank correlation compared to using the online model ensemble. These results show more concretely the fact that predictions by an ensemble of online models on out-domain data suffers from instability along the optimization trajectory, and EoA mitigates this problem.

Table 2. Spearman correlation (closer to 1 is better) between within-run in-domain validation accuracy and out-domain test accuracy on the multiple datasets in the DomainBed benchmark for *ensemble*. In most cases, ensemble of averages (right) has a significantly better rank correlation compared with ensemble of online models (left).

PACS	Without averaging	With averaging
Art	0.06	**0.78**
Cartoon	0.33	**0.81**
Photo	**-0.12**	-0.52
Sketch	0.43	**0.70**
TerraIncognita	**Without averaging**	**With averaging**
L100	0.48	**1**
L38	0.17	**0.95**
L43	**0.59**	0.38
L46	0.08	**0.61**
VLCS	**Without averaging**	**With averaging**
Caltech101	0.52	**0.81**
LabelMe	0.05	**0.38**
Sun09	0.63	**0.82**
VOC2007	0.55	**0.65**
OfficeHome	**Without averaging**	**With averaging**
Art	0.27	**0.92**
Clipart	0.66	**0.95**
Product	0.20	**0.95**
RealWorld	0.09	**0.78**
DomainNet	**Without averaging**	**With averaging**
Clip	**1**	**1**
Info	0.88	**1**
Paint	0.98	**1**
Quick	0.95	**1**
Real	0.97	**1**
Sketch	0.97	**1**

5. DomainBed Benchmarking

We now benchmark model averaging (ERM w/ MA) and ensemble of averages against online models (ERM, without MA) and ensemble of online models (ensembles). Note that all these models are trained using the ERM objective as before. We evaluate on PACS (Li et al., 2017), VLCS (Fang et al., 2013), OfficeHome (Venkateswara et al., 2017), TerraIncognita (Beery et al., 2018) and DomainNet (Peng et al., 2019) datasets in DomainBed. The training-evaluation protocols are the same as described in section 3 for moving average and online models, and in section 4 for ensembles. Full details can be found in section A in the appendix.

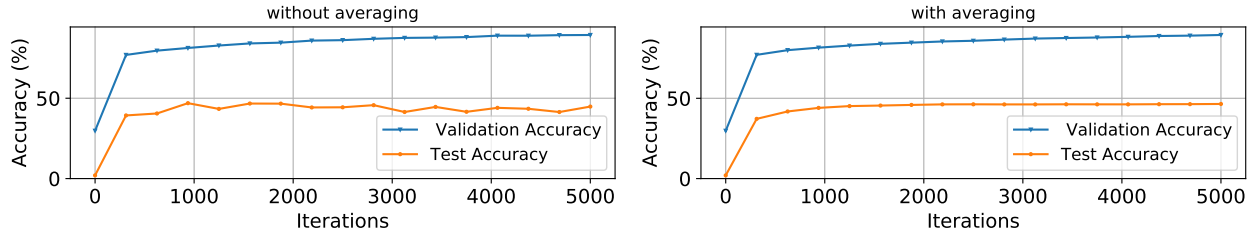

Figure 5. Ensemble of moving averages (EoA) (right) has better out-domain test performance *stability* compared with ensemble of online models (left), w.r.t. in-domain validation accuracy. **Details**: The plots are for the TerraIncognita dataset with domain L38 used as the test domain, and others as training/validation domain, and ResNet-50. Each ensemble has 6 different models from independent runs with different random seeds, hyper-parameters, and training/validation split.

Table 3. Performance benchmarking on 5 datasets of the DomainBed benchmark using two different pre-trained models. SWAD is the previous SOTA. **See Table 5 in appendix for comparison with more methods.** Note that ensembles do not have confidence interval because an ensemble uses all the models to make a prediction. Gray background shows our proposal. *Our runs* implies we ran experiments, but we did not propose it.

Algorithm	PACS	VLCS	OfficeHome	TerraIncognita	DomainNet	Avg.
ResNet-50 (pre-trained on ImageNet)						
ERM (Gulrajani & Lopez-Paz, 2020)	85.7 ± 0.5	77.4 ± 0.3	67.5 ± 0.5	47.2 ± 0.4	41.2 ± 0.2	63.8
ERM (our runs)	84.4 ± 0.8	77.1 ± 0.5	66.6 ± 0.2	48.3 ± 0.2	43.6 ± 0.1	64.0
Ensemble (our runs)	87.6	78.5	70.8	49.2	**47.7**	66.8
SWAD (Cha et al., 2021)	88.1 ± 0.4	**79.1 ± 0.4**	70.6 ± 0.3	50.0 ± 0.4	46.5 ± 0.2	66.9
ERM w/ MA (ours)	87.5 ± 0.2	78.2 ± 0.2	70.6 ± 0.1	50.3 ± 0.5	46 ± 0.1	66.5
Ensemble of Averages (ours)	**88.6**	**79.1**	**72.5**	**52.3**	47.4	**68.0**
ResNeXt-50 32x4d (semi-supervised pre-training on IG-1B targeted and ImageNet data, Yalniz et al. (2019))						
ERM (our runs)	88.9 ± 0.3	79.0 ± 0.1	70.9 ± 0.5	51.4 ± 1.2	48.1 ± 0.2	67.7
Ensemble (our runs)	91.2	80.3	77.8	53.5	52.8	71.1
ERM w/ MA (ours)	92.7 ± 0.3	79.7 ± 0.3	78.6 ± 0.1	53.3 ± 0.1	53.5 ± 0.1	71.6
Ensemble of Averages (ours)	**93.2**	**80.4**	**80.2**	**55.2**	**54.6**	**72.7**

Comparison with existing results using ResNet-50 pre-trained on ImageNet: Here we compare existing methods with our runs. All methods use ResNet-50 (He et al., 2016) pre-trained on ImageNet as initialization.

Comparing ERM (Gulrajani & Lopez-Paz, 2020) and ERM (our runs), we find that they perform similarly, especially considering we have used a smaller hyper-parameter space (further discussion in section 7). A comparison between SWAD and ERM w/ MA shows that SWAD is slightly better (by 0.4% on average). However, recall that our protocol retains the advantage of not tuning any hyper-parameters while SWAD has 3 additional ones that they tune separately in addition to the optimization hyper-parameters. Finally, EoA outperforms all the existing results: ERM by 4% and SWAD (previous SOTA) by 1.1%. Importantly, note that while all non-ensemble models report the average test accuracy of multiple models following the protocol of Gulrajani & Lopez-Paz (2020), EoA test accuracy is achieved by a single predictor that combines the output of multiple models.

Comparison between different components in our experiments: In addition to ResNet-50 pre-trained on ImageNet, we now also experiment with ResNeXt-50 32x4d, that is pre-trained using semi-weakly supervised objective on IG-1B targeted (containing 1 billion weakly labeled images) and ImageNet labeled data (Yalniz et al., 2019). Note that both ResNet-50 and ResNeXt-50 32x4d have similar number of parameters (~ 25**M**). The reason for this choice is that recent trends in deep learning has shown that models pre-trained on larger datasets achieve better downstream transfer performance (Dosovitskiy et al., 2020; Mahajan et al., 2018). Therefore, we expect this model to improve the ERM baseline, and our goal is to show that model averaging and EoA provide a consistent performance boost in both baselines.

Table 3 shows that the baseline (ERM) performance of ResNeXt-50 32x4d is 67.7% on average over all the datasets, which is 3.3% better than the performance of ResNet-50. And we find that in both cases, the moving average model (ERM w/ MA) achieves a performance boost (2.5% and 3.9%). EoA provides a larger boost in both cases: $\sim 4\%$

for ResNet-50 and 5% for ResNeXt-50 32x4d. For both models, we find that the performance of ensembles without moving average roughly matches the performance of moving average models, which are both worse compared with EoA. Finally, we also note that the performance boost of EoA over baseline, for each dataset, either matches or is larger for ResNeXt-50 32x4d, i.e., model pre-trained on larger data.

6. Related Work

6.1. Model Averaging

A theoretical perspective: In our model averaging protocol, we compute a simple moving average of the model parameters starting early during training, but not at initialization. This is known as *tail-averaging* (Jain et al., 2018), which is slightly different from Polyak-Ruppert averaging (Polyak & Juditsky, 1992) in that the latter starts averaging from the very beginning of training. In the context of least square regression in the IID setting, Jain et al. (2018) theoretically study the behavior of tail averaging and show that the excess risk of the moving average model is upper bounded by a bias and a variance term. This bias term depends on the initialization state of the parameter, but interestingly, it decays exponentially with t_0, where t_0 is the iteration at which model averaging is started. The variance term on the other hand depends on the covariance of the noise inherent in the data w.r.t. the optimal parameter, and is shown to decay at a faster rate when using model averaging, as opposed to a slower rate without averaging. This motivated them to propose *tail-averaging*.

Model averaging has also been shown to have a regularization effect (Neu & Rosasco, 2018) similar to that of Tikhonov regularization (Tikhonov, 1943). This regularization has been classically used in ill-posed optimization problems (typically least squared regression), which are *under-specified*. This property provides an interesting connection between model averaging and the *under-specification* problem discussed in D'Amour et al. (2020), where the authors perform large scale experiments showing that the performance of multiple over-parameterized deep models, trained independently with different hyper-parameters and seeds, have a high variance on out-domain data, even though their in-domain performances are very close together. Based on this connection, a simple intuition why one can expect model averaging to help in domain generalization is its Tikhonov regularization effect. However, this intuition requires a more thorough investigation.

SWAD (Cha et al., 2021): SWAD propose flat minima as a means for improving domain generalization. Following the intuition of stochastic weight averaging (SWA, Izmailov et al. (2018)), they use model averaging to find flat minima. However, their proposal is different from sampling model states at regular intervals and towards the end of training (as done in SWA). SWAD selects contiguous model states along the optimization path for averaging, based on their validation loss. The use of validation loss to select model states is proposed in order to prevent including an underperforming state (determined using the in-domain validation set) in the moving average model. SWAD however adds additional hyper-parameters of its own: the validation loss threshold below which the the model states are selected, and patience parameters (number of iterations that determine the start and end of the averaging process). Note that this also requires computing validation loss more frequently during training. In this context, we show that finding the start and end period for model averaging does not need to be so meticulous. Instead, we can simply start model averaging early during training and continue till the end. This difference arises from the fact that SWAD uses the online network to calculate validation performance while we use the SMA model in our protocol. This is explained further in section 2.2. The benefit our observations provide over SWAD is that they allow us to take advantage of model averaging without the additional hyper-parameters and compute required by the SWAD algorithm.

6.2. Domain Generalization

Existing methods aimed at domain generalization can be broadly categorized into techniques that perform domain alignment, regularization, data augmentation, and meta-learning. Domain alignment is perhaps the most intuitive direction, in which methods aim to learn latent representations which have similar distributions across different domains (Sun & Saenko, 2016; Li et al., 2018b; Shi et al., 2021; Rame et al., 2021). There are different variants of this idea, such as minimizing some divergence metric between the latent representation of different domains (E.g. DANN (Ganin et al., 2016)), or less strictly, minimizing the difference between the latent statistics of different domains (E.g. DICA (Muandet et al., 2013), CORAL (Sun & Saenko, 2016)). In the meta learning category, source domains are typically split into 2 subsets to to be used as the training and test domains in episodes to simulate the domain generalization setting (Li et al., 2018a; 2019). Data augmentation is also a popular tool used for improving domain generalization. It ranges from introducing various types of augmentations to simulate unseen test domain conditions (E.g. style transfer (Yue et al., 2019; Zhou et al., 2021b)) to self-supervised learning involving matching the representations of an image with different augmentations (E.g. Albuquerque et al. (2020); Bucci et al. (2021)). Finally, different ways of regularizing models (implicit and explicit) have also been developed with the goal of encouraging domain-invariant feature learning (Sagawa et al., 2019; Xu et al., 2020; Wang et al., 2020). For

instance, invariant risk minimization (Arjovsky et al., 2019) propose a regularization such that the classifier is optimal in all the environments. Representation Self-Challenging (Huang et al., 2020) propose to suppress the dominant features that get activated on the training data, which forces the network to use other features that correlate with labels. Risk extrapolation (Krueger et al., 2021) propose a regularization that minimizes the variance between domain-wise loss, in the hope that it is representative of the variance including unseen test domains. See Zhou et al. (2021a) for a survey on DG methods.

Our investigation in this work is complementary to all these domain generalization methods. Additionally, one of our main focus is to also study and improve performance instability on out-domain data during training, which results in more reliable model selection. This aspect has not received much attention.

7. Discussion

Information leaking considerations: We present many experiments where validation and test performances are studied. It is therefore natural to wonder if there was any information leak from the test set while performing this analysis. We note that in the model averaging protocol we investigated, there were two moving parts: iteration t_0 at which model averaging is started, and averaging frequency. We studied them in section 3.1 and 3.2 respectively. For both of them, we proposed to fix their values universally instead of tuning them on each dataset. Specifically, Jain et al. (2018) propose tail averaging in which iterates from every iteration are used for computing the simple moving average. We found this proposal to work well empirically in our analysis, and therefore set averaging frequency to 1. For start iteration t_0 on the other hand, Jain et al. (2018) theoretically show that the initial bias term in the excess error upper bound decays exponentially with t_0. In line with this theory, our analysis showed that an iteration close to but not at initialization worked well. So we arbitrarily set $t_0 = 100$. Note that these are not their optimal values, but are rather arbitrary choices guided by our investigation and existing theory. Aside from these two objects (which we fix in all experiments except the aforementioned ablation), there are no hyper-parameters introduced by the averaging protocol or the ensemble of averages studied in this paper, and all other experiments are purely observational. Finally, we followed the protocol of Gulrajani & Lopez-Paz (2020) for training and evaluation.

Smaller HP search space: We use a smaller hyper-parameter search space compared with that in Gulrajani & Lopez-Paz (2020). Nonetheless, we find that on average, our runs of the ERM baseline performance (without model averaging) yield 64% test accuracy on average compared with 63.8% reported in Gulrajani & Lopez-Paz (2020) on the 5 datasets we used. We also note that model averaging and ensemble of averages, that we study in our work, are not competing with ERM baseline, in the sense that these techniques essentially rely on the quality of the baseline model to further boost performance. Therefore any boost in the ERM baseline performance is likely to improve the model averaging and EoA performance. This is also evident in our benchmarking experiments in Table 3, where using ResNeXt-50 32x4d pre-trained on a larger dataset (Yalniz et al., 2019) has a better ERM baseline performance compared to ResNet-50 pre-trained on ImageNet. This results in a further boost of 3.9% and 5% test accuracy on average when using model averaging and EoA respectively.

Diversity: Model averaging mitigates instability within a run, which makes model selection more reliable. However, we note that the gap in performance between different runs still exists, although it is smaller on average compared with online models (see training evolution plots in appendix for reference). This implies that there is still diversity among the different runs. Perhaps this is one of the reasons why EoA provides a further boost in performance over individual moving average models. This is also inline with Fort et al. (2019) which shows that ensembling methods such as model averaging and Monte Carlo dropout (Gal & Ghahramani, 2016) do not provide diversity in function space as much as ensembles of independently trained models.

Scalability: Following the protocol of Gulrajani & Lopez-Paz (2020), we used samples from all the training domains in each mini-batch update. However, in settings where the number of domains is very large, this approach can be prohibitive. As an alternative, we also performed preliminary experiments in which we stochastically picked one of the training domains at every iteration, and sampled a mini-batch from that domain to update parameters. We found that this protocol resulted in a similar performance as that achieved by the protocol used in our work.

8. Conclusion

We investigated a simple protocol for model averaging (without hyper-parameters) in the ERM framework, and showed that it provides a significant boost to out-domain performance compared to un-averaged models. Building on this observation, we showed that an ensemble of averaged models also performs better compared to an ensemble of un-averaged models. Importantly, we showed that in both cases, model averaging significantly improves the rank correlation between in-domain validation accuracy and out-domain test accuracy, which is crucial for reliable model selection using in-domain validation data. Finally, we experimented with two pre-trained models, one on ImageNet, and the other on a much larger dataset (Yalniz et al., 2019). We found that

the latter significantly improves the ERM baseline, and in both cases, ensemble of averages boosts performance by $4\% - 5\%$.

References

Albuquerque, I., Naik, N., Li, J., Keskar, N., and Socher, R. Improving out-of-distribution generalization via multi-task self-supervised pretraining. *arXiv preprint arXiv:2003.13525*, 2020.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

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: 2178–2186, 2011.

Breiman, L. Bagging predictors. *Machine learning*, 24(2): 123–140, 1996.

Bucci, S., D'Innocente, A., Liao, Y., Carlucci, F. M., Caputo, B., and Tommasi, T. Self-supervised learning across domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Cha, J., Chun, S., Lee, K., Cho, H.-C., Park, S., Lee, Y., and Park, S. Swad: Domain generalization by seeking flat minima. *arXiv preprint arXiv:2102.08604*, 2021.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

Dietterich, T. G. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

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.

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.

Huang, Z., Wang, H., Xing, E. P., and Huang, D. Self-challenging improves cross-domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 124–140. Springer, 2020.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

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.

Jain, P., Kakade, S., Kidambi, R., Netrapalli, P., and Sidford, A. Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18, 2018.

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

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pp. 5815–5826. PMLR, 2021.

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, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018a.

Li, D., Zhang, J., Yang, Y., Liu, C., Song, Y.-Z., and Hospedales, T. M. Episodic training for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1446–1455, 2019.

Li, H., Pan, S. J., Wang, S., and Kot, A. C. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5400–5409, 2018b.

Li, Y., Tian, X., Gong, M., Liu, Y., Liu, T., Zhang, K., and Tao, D. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 624–639, 2018c.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and Van Der Maaten, L. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 181–196, 2018.

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

Neu, G. and Rosasco, L. Iterate averaging as regularization for stochastic gradient descent. In *Conference On Learning Theory*, pp. 3222–3242. PMLR, 2018.

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.

Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Rame, A., Dancette, C., and Cord, M. Fishr: Invariant gradient variances for out-of-distribution generalization. *arXiv preprint arXiv:2109.02934*, 2021.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Shi, Y., Seely, J., Torr, P. H., Siddharth, N., Hannun, A., Usunier, N., and Synnaeve, G. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937*, 2021.

Sun, B. and Saenko, K. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pp. 443–450. Springer, 2016.

Tikhonov, A. N. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pp. 195–198, 1943.

Vapnik, V. and Vapnik, V. Statistical learning theory wiley. *New York*, 1(624):2, 1998.

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.

Wang, Y., Li, H., and Kot, A. C. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3622–3626. IEEE, 2020.

Xu, M., Zhang, J., Ni, B., Li, T., Wang, C., Tian, Q., and Zhang, W. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6502–6509, 2020.

Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., and Mahajan, D. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

Yue, X., Zhang, Y., Zhao, S., Sangiovanni-Vincentelli, A., Keutzer, K., and Gong, B. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2100–2110, 2019.

Zhou, K., Liu, Z., Qiao, Y., Xiang, T., and Loy, C. C. Domain generalization: A survey. *arXiv preprint arXiv:2103.02503*, 2021a.

Zhou, K., Loy, C. C., and Liu, Z. Semi-supervised domain generalization with stochastic stylematch. *arXiv preprint arXiv:2106.00592*, 2021b.

Table 4. Hyper-parameter search space for all experiments.

Hyper-parameter	Default value	Random distribution	
		Gulrajani & Lopez-Paz (2020)	Ours
Learning rate	$5e-5$	$10^{\text{Uniform}(-5,-3.5)}$	$5e-5$
Batch size	32	$2^{\text{Uniform}(3,5.5)}$	32
ResNet dropout	0	RandomChoice([0, 0.1, 0.5])	RandomChoice([0, 0.1, 0.5])
Weight decay	0	$10^{\text{Uniform}(-6,-2)}$	$10^{\text{Uniform}(-6,-4)}$

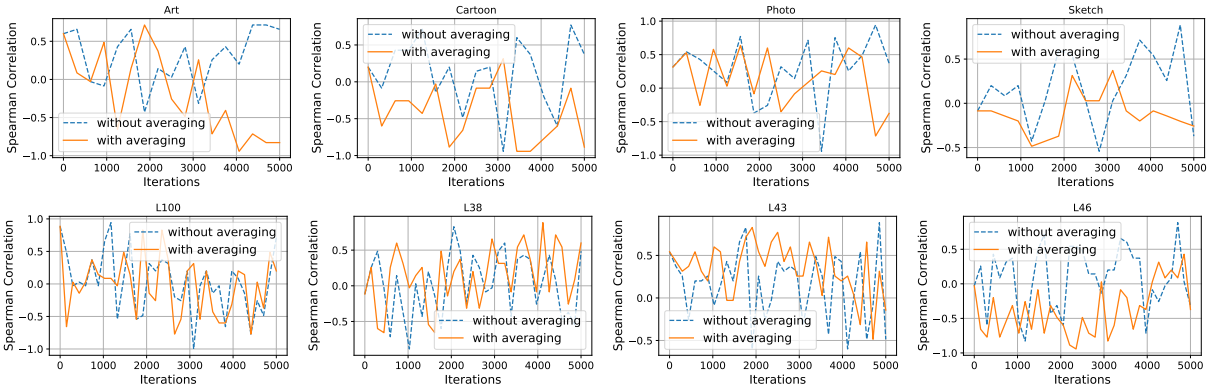

Figure 6. Spearman correlation between *cross-run* in-domain validation accuracy and out-domain test accuracy for PACS dataset (top) and TerraIncognita dataset (bottom). The cross-run rank correlations are poor (not consistently close to 1) for both online model (without averaging) and moving average model. This implies that in-domain validation performance based model selection is not a reliable approach for selecting a single model from a pool of multiple independently trained models. See section 3.4 for details.

Appendix

A. Training and. Evaluation Protocols

We use the training protocol described in Gulrajani & Lopez-Paz (2020) with minor changes: we use a smaller hyper-parameter search space (shown in Table 4) and smaller number of random trials for computational reasons, and train on DomainNet dataset for $15,000$ iterations instead of $5,000$ similar to Cha et al. (2021), because its training loss is quite high. For a dataset with D domains, we run a total of $6D$ random trials. This results in 6 experiments per domain, in which this domain is used as the test set, while the remaining domains are used as training/validation set (randomly split). This is also the reason why we use a smaller hyper-parameter search space, because otherwise the search space would be under-sampled. For moving average models, the iteration t_0 at which averaging is started (Eq. 1) is set to be 100 in all experiments unless specified otherwise. For ensembles (both with and without averaged models), the 6 models corresponding to the 6 experiments per domain, in which this domain is used as the test set (as described above), are used for ensembling as described in section 4.

All models are trained using the ERM objective and optimized using the Adam optimizer (Kingma & Ba, 2014). We use ResNet-50 (He et al., 2016) pre-trained on Imagenet as our initialization for training in all the experiments. In the final benchmarking experiment, we also use ResNeXt-50 32x4d, that is trained using semi-weakly supervised objective on IG-1B targeted (containing 1 billion weakly labeled images) and ImageNet labeled data (Yalniz et al., 2019). This model was downloaded from Pytorch hub. For all models, the batch normalization (Ioffe & Szegedy, 2015) statistics are kept frozen throughout training and inference. Validation accuracy is calculated every 300 iterations for all datasets except DomainNet where it is calculated every 1000 iterations. Unless specified otherwise, we use the said protocol in all the experiments. For model selection, we use the *training-domain validation set* protocol in Gulrajani & Lopez-Paz (2020) with $80\% - 20\%$ training-validation split, and the average out-domain test performance is reported across all runs for each domain.

B. Additional Tables and Plots

Table 5. Performance benchmarking on 5 datasets of the DomainBed benchmark using two different pre-trained models. SWAD is the previous SOTA. Note that ensembles do not have confidence interval because an ensemble uses all the models to make a prediction. Gray background shows our proposal. *Our runs* implies we ran experiments, but we did not propose it.

Algorithm	**PACS**	**VLCS**	**OfficeHome**	**TerraIncognita**	**DomainNet**	**Avg.**
ResNet-50 (pre-trained on ImageNet)						
ERM (Gulrajani & Lopez-Paz, 2020)	85.7 ± 0.5	77.4 ± 0.3	67.5 ± 0.5	47.2 ± 0.4	41.2 ± 0.2	63.8
ERM (our runs)	84.4 ± 0.8	77.1 ± 0.5	66.6 ± 0.2	48.3 ± 0.2	43.6 ± 0.1	64.0
Ensemble (our runs)	87.6	78.5	70.8	49.2	**47.7**	66.8
IRM (Arjovsky et al., 2019)	84.4 ± 1.1	78.1 ± 0.0	66.6 ± 1.0	47.9 ± 0.7	35.7 ± 1.9	62.5
Group DRO (Sagawa et al., 2019)	84.1 ± 0.4	77.2 ± 0.6	66.9 ± 0.3	47.0 ± 0.3	33.7 ± 0.2	61.8
Mixup (Xu et al., 2020; Wang et al., 2020)	84.3 ± 0.5	77.7 ± 0.4	69.0 ± 0.1	48.9 ± 0.8	39.6 ± 0.1	63.9
MLDG (Li et al., 2018a)	84.8 ± 0.6	77.1 ± 0.4	68.2 ± 0.1	46.1 ± 0.8	41.8 ± 0.4	63.6
CORAL (Sun & Saenko, 2016)	86.0 ± 0.2	77.7 ± 0.5	68.6 ± 0.4	46.4 ± 0.8	41.8 ± 0.2	64.1
MMD (Li et al., 2018b)	85.0 ± 0.2	76.7 ± 0.9	67.7 ± 0.1	49.3 ± 1.4	39.4 ± 0.8	63.6
DANN (Ganin et al., 2016)	84.6 ± 1.1	78.7 ± 0.3	65.4 ± 0.6	48.4 ± 0.5	38.4 ± 0.0	63.1
C-DANN (Li et al., 2018c)	82.8 ± 1.5	78.2 ± 0.4	65.6 ± 0.5	47.6 ± 0.8	38.9 ± 0.1	62.6
Fish (Shi et al., 2021)	85.5 ± 0.3	77.8 ± 0.3	68.6 ± 0.4	45.1 ± 1.3	42.7 ± 0.2	63.9
Fishr (Rame et al., 2021)	85.5 ± 0.4	77.8 ± 0.1	67.8 ± 0.1	47.4 ± 1.6	41.7 ± 0.0	65.7
SWAD (Cha et al., 2021)	88.1 ± 0.4	**79.1 ± 0.4**	70.6 ± 0.3	50.0 ± 0.4	46.5 ± 0.2	66.9
ERM w/ MA (ours)	87.5 ± 0.2	78.2 ± 0.2	70.6 ± 0.1	50.3 ± 0.5	46 ± 0.1	66.5
Ensemble of Averages (ours)	**88.6**	**79.1**	**72.5**	**52.3**	47.4	**68.0**
ResNeXt-50 32x4d (semi-supervised pre-training on IG-1B targeted and ImageNet data, Yalniz et al. (2019))						
ERM (our runs)	88.9 ± 0.3	79.0 ± 0.1	70.9 ± 0.5	51.4 ± 1.2	48.1 ± 0.2	67.7
Ensemble (our runs)	91.2	80.3	77.8	53.5	52.8	71.1
ERM w/ MA (ours)	92.7 ± 0.3	79.7 ± 0.3	78.6 ± 0.1	53.3 ± 0.1	53.5 ± 0.1	71.6
Ensemble of Averages (ours)	**93.2**	**80.4**	**80.2**	**55.2**	**54.6**	**72.7**

Table 6. Out-domain accuracy for PACS dataset.

Algorithm	**A**	**C**	**P**	**S**	**Avg.**
ResNet-50 (pre-trained on ImageNet)					
ERM	86.4 ± 1.0	80.4 ± 0.6	94.8 ± 0.1	76.2 ± 1.7	84.4
Ensemble	88.3	**83.6**	96.5	81.9	87.6
ERM w/ MA	89.1 ± 0.1	82.6 ± 0.2	97.6 ± 0.0	80.5 ± 0.9	87.5
Ensemble of Averages (EoA)	**90.5**	83.4	**98.0**	**82.5**	**88.6**
ResNeXt-50 32x4d (semi-supervised pre-training on IG-1B targeted and ImageNet data, Yalniz et al. (2019))					
ERM	84.7 ± 1.6	87.6 ± 0.1	97.6 ± 0.4	85.7 ± 0.1	88.9
Ensemble	90.2	89.2	98.1	87.2	91.2
ERM w/ MA	92.6 ± 0.3	90.9 ± 0.8	99.1 ± 0.3	88.3 ± 0.5	92.7
Ensemble of Averages (EoA)	**93.1**	**91.8**	**99.2**	**88.9**	**93.2**

Table 7. Out-domain accuracy for VLCS dataset.

Algorithm	C	L	S	V	Avg.
ResNet-50 (pre-trained on ImageNet)					
ERM	98.5 ± 0.5	62.4 ± 1.4	72.1 ± 0.0	75.4 ± 0.1	77.1
Ensemble	98.7	**64.5**	72.1	**78.9**	78.5
ERM w/ MA	99.0 ± 0.2	63.0 ± 0.2	74.5 ± 0.3	76.4 ± 1.1	78.2
Ensemble of Averages (EoA)	**99.1**	63.1	**75.9**	78.3	**79.1**
ResNeXt-50 32x4d (semi-supervised pre-training on IG-1B targeted and ImageNet data, Yalniz et al. (2019))					
ERM	97.0 ± 0.4	67.8 ± 0.7	75.7 ± 0.2	75.5 ± 0.6	79.0
Ensemble	98.4	**66.1**	76.4	80.5	80.3
ERM w/ MA	**98.8 ± 0.2**	63.3 ± 0.6	77.7 ± 0.2	79.2 ± 0.8	79.7
Ensemble of Averages (EoA)	98.7	64.1	**78.2**	**80.6**	**80.4**

Table 8. Out-domain accuracy for OfficeHome dataset.

Algorithm	A	C	P	R	Avg.
ResNet-50 (pre-trained on ImageNet)					
ERM	60.5 ± 0.7	54.5 ± 0.8	74.7 ± 0.8	76.6 ± 0.2	66.6
Ensemble	65.6	58.5	78.7	80.5	70.8
ERM w/ MA	66.7 ± 0.5	57.1 ± 0.1	78.6 ± 0.1	80.0 ± 0	70.6
Ensemble of Averages (EoA)	**69.1**	**59.8**	**79.5**	**81.5**	**72.5**
ResNeXt-50 32x4d (semi-supervised pre-training on IG-1B targeted and ImageNet data, Yalniz et al. (2019))					
ERM	64.7 ± 1.0	60.6 ± 0.3	77.1 ± 0.4	81.3 ± 0.2	70.9
Ensemble	74.1	67.3	83.9	86.0	77.8
ERM w/ MA	76.7 ± 0.4	67.8 ± 0.0	84.0 ± 0.1	85.8 ± 0.1	78.6
Ensemble of Averages (EoA)	**79.0**	**70.0**	**85.2**	**86.5**	**80.2**

Table 9. Out-domain accuracy for TerraIncognita dataset.

Algorithm	L100	L38	L43	L46	Avg.
ResNet-50 (pre-trained on ImageNet)					
ERM	52.9 ± 3.3	43.3 ± 1.7	56.9 ± 0.4	40.2 ± 2.1	48.3
Ensemble	53.0	42.6	60.5	40.8	49.2
ERM w/ MA	54.9 ± 0.4	45.5 ± 0.6	60.1 ± 1.5	40.5 ± 0.4	50.3
Ensemble of Averages (EoA)	**57.8**	**46.5**	**61.3**	**43.5**	**52.3**
ResNeXt-50 32x4d (semi-supervised pre-training on IG-1B targeted and ImageNet data, Yalniz et al. (2019))					
ERM	64.0 ± 0.0	44.7 ± 3.2	56.1 ± 3.0	40.9 ± 1.4	51.4
Ensemble	**65.7**	43.1	62.6	42.6	53.5
ERM w/ MA	60.1 ± 1.0	**47.3 ± 1.4**	61.0 ± 1.7	44.9 ± 0.8	53.3
Ensemble of Averages (EoA)	63.5	46.0	**64.3**	**46.9**	**55.2**

Table 10. Out-domain accuracy for Domainet dataset.

Algorithm	clip	info	paint	quick	real	sketch	Avg.
ResNet-50 (pre-trained on ImageNet)							
ERM	63.4 ± 0.2	20.6 ± 0.1	50.0 ± 0.1	13.8 ± 0.4	62.1 ± 0.2	51.9 ± 0.3	43.6
Ensemble	**68.3**	23.1	54.5	16.3	**66.9**	57.0	**47.7**
ERM w/ MA	64.4 ± 0.3	22.4 ± 0.2	53.4 ± 0.3	15.4 ± 0.1	64.7 ± 0.2	55.5 ± 0.1	46.0
Ensemble of Averages (EoA)	65.9	**23.4**	**55.3**	**16.5**	66.4	**57.1**	47.4
ResNeXt-50 32x4d (semi-supervised pre-training on IG-1B targeted and ImageNet data, Yalniz et al. (2019))							
ERM	68.8 ± 0.1	25.5 ± 0.1	55.9 ± 0.3	14.7 ± 0.7	65.8 ± 0.4	58.0 ± 0.4	48.1
Ensemble	74.3	28.7	61.1	17.0	71.9	63.5	52.8
ERM w/ MA	73.7 ± 0.1	29.9 ± 0.0	62.8 ± 0.1	18.1 ± 0.1	73.0 ± 0.2	63.6 ± 0.4	53.5
Ensemble of Averages (EoA)	**74.6**	**31.3**	**63.7**	**19.3**	**73.6**	**65.1**	**54.6**

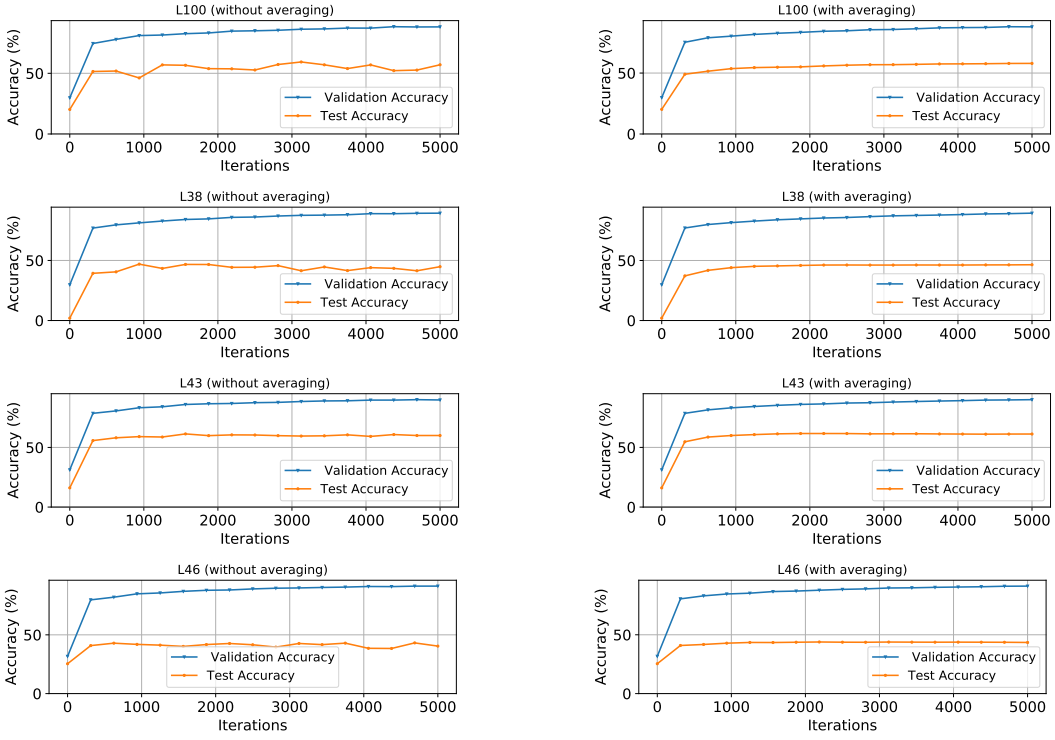

Figure 7. Ensemble of moving average (EOA) models (right) has better out-domain test performance *stability* compared with ensemble of online models (left), w.r.t. in-domain validation accuracy. **Details**: The plots are for the TerraIncognita dataset with the domain name in title used as the test domain, and others as training/validation domain, and ResNet-50. Each ensemble has 6 different models from independent runs with a different random seed and training/validation split.

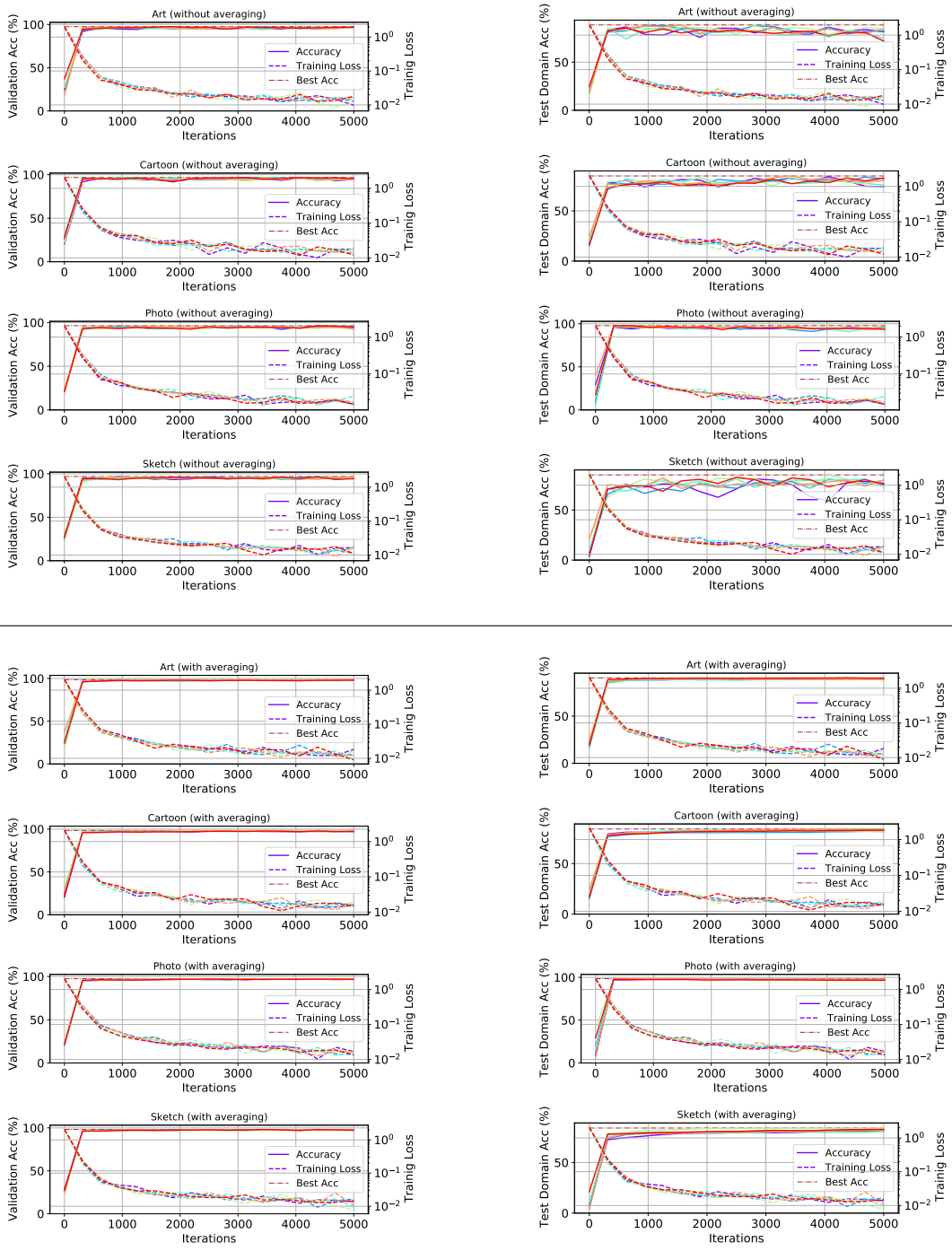

Figure 8. Evolution of training loss, in-domain validation accuracy and out-domain test accuracy for ResNet-50 (pre-trained on ImageNet) trained on PACS without model averaging (top 4 rows) and with model averaging (bottom 4 rows) for $5,000$ iterations with the domain mentioned in the title used as test domain and remain domains as training/validation data. Each color represents a different run with randomly chosen seed, hyper-parameters and training-validation split following Gulrajani & Lopez-Paz (2020).

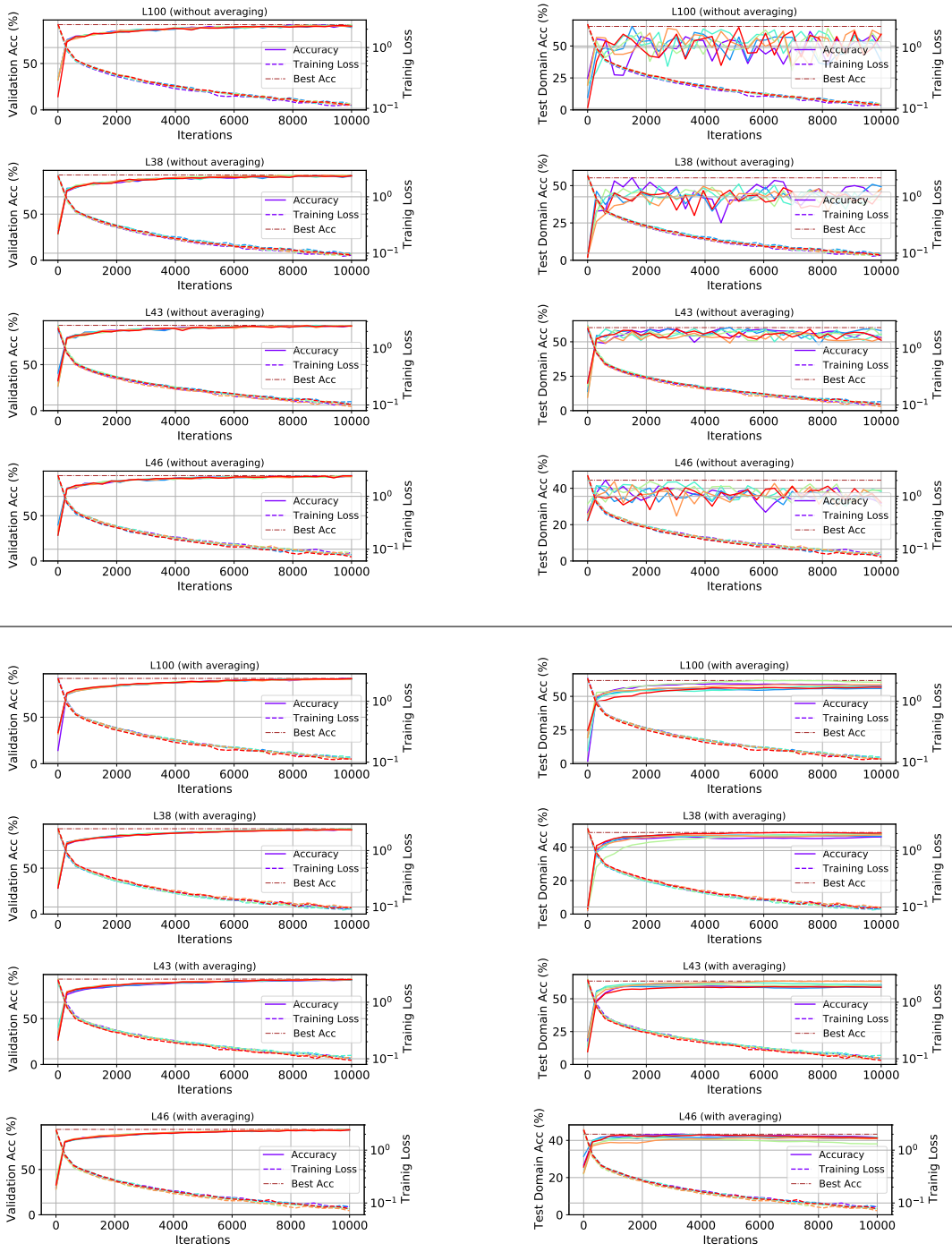

Figure 9. Evolution of training loss, in-domain validation accuracy and out-domain test accuracy for ResNet-50 (pre-trained on ImageNet) trained on TerraIncognita without model averaging (top 4 rows) and with model averaging (bottom 4 rows) for $10,000$ iterations with the domain mentioned in the title used as test domain and remain domains as training/validation data. Each color represents a different run with randomly chosen seed, hyper-parameters and training-validation split following Gulrajani & Lopez-Paz (2020). **Gist**: Out-domain test performance is unstable without model averaging, which causes problem for model selection using in-domain validation performance. Model averaging is able to mitigate this instability.

C. Code

The below Pytorch code shows how to augment the ERM function with the moving average protocol discussed in our work following the codebase provided in https://github.com/facebookresearch/DomainBed.

```
class MovingAvg:
    def __init__(self, network):
        self.network = network
        self.network_sma = copy.deepcopy(network)
        self.network_sma.eval()
        self.sma_start_iter = 100
        self.global_iter = 0
        self.sma_count = 0

    def update_sma(self):
        self.global_iter += 1
        if self.global_iter>=self.sma_start_iter:
            self.sma_count += 1
            for param_q, param_k in zip(self.network.parameters(), self.network_sma.parameters()):
                param_k.data = (param_k.data* self.sma_count + param_q.data)/(1.+self.sma_count)
        else:
            for param_q, param_k in zip(self.network.parameters(), self.network_sma.parameters()):
                param_k.data = param_q.data


class ERM_SMA(Algorithm, MovingAvg):
    """
    Empirical Risk Minimization (ERM) with Simple Moving Average (SMA) prediction model
    """

    def __init__(self, input_shape, num_classes, num_domains, hparams):
        Algorithm.__init__(self, input_shape, num_classes, num_domains, hparams)

        self.featurizer = networks.Featurizer(input_shape, self.hparams)
        self.classifier = networks.Classifier(
            self.featurizer.n_outputs,
            num_classes,
            self.hparams['nonlinear_classifier'])

        self.network = nn.Sequential(self.featurizer, self.classifier)
        self.optimizer = torch.optim.Adam(
            self.network.parameters(),
            lr=self.hparams["lr"],
            weight_decay=self.hparams['weight_decay']
        )

        MovingAvg.__init__(self, self.network)

    def update(self, minibatches, unlabeled=None):
        all_x = torch.cat([x for x,y in minibatches])
        all_y = torch.cat([y for x,y in minibatches])
        loss = F.cross_entropy(self.network(all_x), all_y)

        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()

        self.update_sma()

        return {'loss': loss.item()}

    def predict(self, x):
        self.network_sma.eval()
        return self.network_sma(x)
```