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

Anonymous Author(s)
Affiliation
Address
email

Abstract

In Domain Generalization (DG) settings, models trained independently 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 this chaotic behavior exists even along the training optimization trajectory of a single model, and propose a simple model averaging protocol that 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 early stopping. Taking advantage of our observation, we show that instead of ensembling unaveraged models (that is typical in practice), ensembling moving average models (EoA) from independent runs further boosts performance. We theoretically explain the boost in performance of ensembling and model averaging by adapting the well known Bias-Variance trade-off to the domain generalization setting. On the DomainBed benchmark, when using a pre-trained ResNet-50, this ensemble of averages achieves an average of 68.0%, beating vanilla ERM (w/o averaging/ensembling) by $\sim 4\%$, and when using a pre-trained RegNetY-16GF, achieves an average of 76.6%, beating vanilla ERM by 6%. *code.zip* is provided in supplementary material.

1 Introduction

Domain generalization (DG, [5]) 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 [10], and excel primarily in the IID setting [51].

While a number of algorithms have been proposed to mitigate this problem (cf [51] for a survey), [18] demonstrate that models trained using empirical risk minimization (ERM, [43]) 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 DG within the ERM framework, and mitigates performance instability of deep models on out-domain data,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

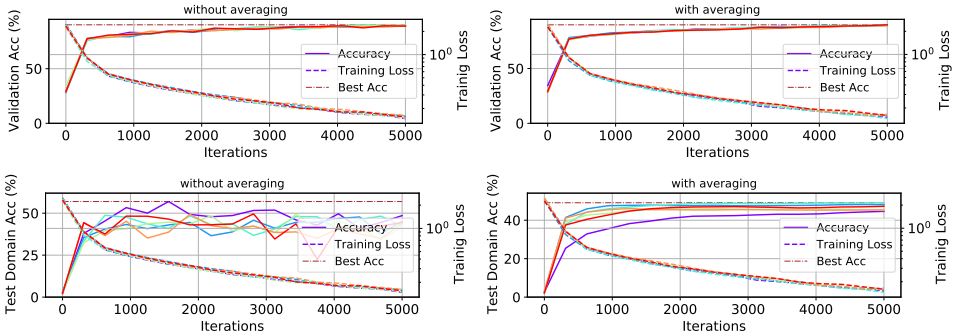

Figure 1: Model averaging improves out-domain performance *stability*. **Left**: In-domain validation accuracy and out-domain test accuracy during training of models using ERM. **Right**: Same as left, except validation and test predictions are made using a simple moving average of the model being optimized, 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 2.

specifically with respect to in-domain validation data. This makes model selection more reliable. Next, taking advantage of our observation, we show that ensembling moving average models 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 DG. The observation that model averaging can boost domain generalization performance is not new, and was exposed by SWAD [8], which inspired our work. Our contribution in this respect are as follows:

1. **Hyperparameter-free**:In contrast to SWAD, which introduces three additional hyper-parameters for its model averaging algorithm, we show that the hyperparameter-free strategy of maintaining a simple moving average (SMA) of the model parameters being trained, starting near initialization (Appendix Figure 5), works just as well (when a pre-trained model is used as initialization).

2. **Computationally efficient**:SWAD requires computing validation performance more frequently than is typically done (2x-6x on the DomainBed datasets), which is needed because it needs to find the start and end iteration between which model averaging is done. This increases compute requirements. This segment is selected based on the validation performance computed using the model being trained. Our proposal to instead use the SMA model to perform early stopping and inference, side-steps this need and does not require frequent validation performance check. We show that the root cause for this difference is that the model being trained has unstable performance on OOD data, while the SMA model has a more stable OOD performance (see Figure 1 and Table 2). Thus this observation results in our hyperparameter-free and more efficient model averaging strategy.

3. **EoA**: Taking advantage of our efficient model averaging protocol (section 2.2), we find that an ensemble of moving average models (EoA) outperforms a traditional ensemble of unaveraged models (Table 4). We also show ablation analysis that the rank correlation between in-domain validation performance and out-domain test performance is better for the ensemble of average models (Table 3).

4. **Theoretical explanation**: To explain why both model averaging and ensembling improve OOD performance under a unified theoretical framework, we adapt the well known Bias-Variance decomposition to the domain generalization setting, and argue that the expected OOD loss for individual models comprises of both the bias and the variance term, while the expected OOD loss for ensembles and averaged models comprises mainly of the bias term only, and is thus strictly lower (section 3.2). Our explanation is in contrast with SWAD, which uses flat minima to explain the improved OOD generalization, which applies to model averaging, but is less straight forward for explaining the boost by ensembles.

5. **Benchmarking**: For benchmarking, we experiment with three different pre-trained models as initializations for DG training, with increasing pre-training dataset size and model size. In these experiments we find that EoA provides a larger gain over the *corresponding* ERM baseline with increasing dataset and model size. These gains range from $4\% - 6\%$ (Table 4). Notice that this claim is different from existing work [20], which states that the baseline ERM performance improves with larger pre-training data and model size.

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 when a pre-trained model is used as initialization. At any iteration t, we denote:

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

where θ_t is the online model's state at iteration t. Note that effectively, $\hat{\theta}_t := \frac{1}{t-t_0+1} \cdot \sum_{t'=t_0}^{t} \theta_{t'}$. 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 5.1. A Pytorch code is provided in Appendix section F.

2.3 Ablation Analysis

Here we perform four ablation studies: 1) impact of the start iteration t_0 used in our SMA protocol in Eq. 1; 2) the frequency of model averaging; 3) instability reduction of SMA model compared to the online mode along the optimization trajectory on out-domain data; 4) correlation between in-domain and out-domain accuracy across independently trained models.

Due to space limitation, we show experiments for 1,2 and 4 in Appendix section C. In summary, we find that: 1) starting averaging close to initialization results in improved out-domain performance (Figure 5 in Appendix) when the parameters are initialized used a pre-trained model; 2) the frequency of SMA does not have a significant impact on performance, unless sampling is done at too large intervals (Figure 6 in Appendix); 4) the rank correlation is poor between validation and test accuracy of *independently* trained models (Figure 8 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 (echoing the findings of [10]).

Table 1: Spearman correlation (closer to 1 is better) between within-run in-domain validation accuracy and out-domain test accuracy on multiple datasets. Model averaging improves rank correlation for both individual models (left) and ensemble of averages (right).

Table 2: Individual Models

TerraIncognita	w/o avg	w/ avg
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	$\mathbf{0.52 \pm 0.14}$

Table 3: Ensembles

TerraIncognita	w/o avg	w/ avg
L100	0.48	**1**
L38	0.17	**0.95**
L43	**0.59**	0.38
L46	0.08	**0.61**

2.3.1 Instability Reduction: Rank Correlation

We study the reliability of model selection for domain generalization when using online models vs moving average models, using rank correlation (see Appendix C.4 for definition). 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 sampled at regular intervals during the training process. 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.

The rank correlations are shown in Table 2 (and Table 8 in Appendix) for the PACS, VLCS, Office-Home, 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. These experiments therefore suggest that the reliability of model selection is significantly higher within a run when using model averaging.

3 Ensemble of Averages (EoA)

[18] 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. An ensemble combines predictions from multiple models, and is a well known approach for achieving this goal [11] by exploiting function diversity [14]. 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 traditional ensembles (specifically the bagging method [6]), the class $\hat{y}$ predicted by an EoA for an input $\mathbf{x}$ is given by the formula:

$$\hat{y} = \arg\max_k Softmax(\frac{1}{E}\sum_{i=1}^{E} f(\mathbf{x}; \hat{\theta}_i))_k \tag{2}$$

where E is the total number of models in the ensemble, $\hat{\theta}_i$ denotes the parameters of the ith moving average model, and the sub-script $(.)_k$ denotes the kth element of the vector argument. Finally, the state $\hat{\theta}_i$ of the ith 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.

3.1 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

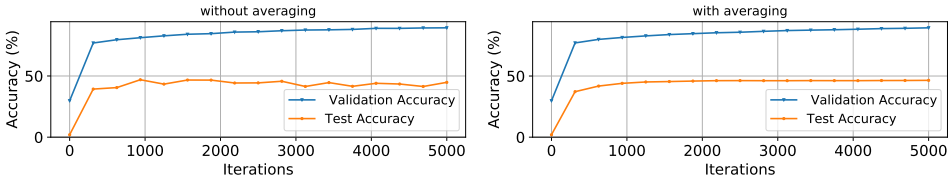

Figure 2: Ensemble of moving averages (EoA) (right) has better out-domain test performance *stability* compared with ensemble of online models (left), w.r.t. in-domain validation accuracy. **Details**: The plots are for the TerraIncognita dataset with domain L38 used as the test domain, and others as 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.

dataset, and fix one of its domains as 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 [18] 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 2. 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 10 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 2.3.1. Here we use the same experimental setup described in the qualitative analysis above. But in addition, we also conduct experiments on VLCS, OfficeHome and DomainNet datasets. The results are shown in Table 3 (and Table 9 in Appendix). 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.

3.2 Why does Ensembling and Model Averaging Improve Performance?

We explain the performance boost achieved by ensemble of averages (see next section) by adapting the Bias-Variance decomposition [17] to the domain generalization setting. For classification tasks with one-hot labels, the Bias-Variance decomposition is given as [49],

$$\mathbb{E}_{\mathbf{x},y}\mathbb{E}_{\mathcal{T}}[CE(y, f(\mathbf{x};\mathcal{T}))] = \underbrace{\mathbb{E}_{\mathbf{x},y}[CE(y, \bar{f}(\mathbf{x}))]}_{\text{Bias}^2} + \underbrace{\mathbb{E}_{\mathbf{x},\mathcal{T}}[KL(\bar{f}(\mathbf{x}), f(\mathbf{x};\mathcal{T}))]}_{\text{Variance}}$$

where CE denotes the cross entropy loss, KL denotes KL divergence, $\mathcal{T} = \{(\mathbf{x}_i^{in}, y_i^{in})\}_{i=1}^N$ are N IID samples drawn from the in-domain training distribution $\mathbb{P}^{in}$, $f(\mathbf{x};\mathcal{T})$ denotes the prediction of the model f on sample $\mathbf{x}$ such that the model is trained on the dataset $\mathcal{T}$, and $\bar{f}(\mathbf{x}) = \mathbb{E}_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})]$. Finally $(\mathbf{x}, y) \sim \mathbb{P}^{out}$ where $\mathbb{P}^{out}$ is the out-domain distribution. Notice how $\mathcal{T}$ and $(\mathbf{x}, y)$ come from different distributions. For instance, in PACS dataset, $\mathbb{P}^{in}$ could be the union of art, cartoon and photo domains, and $\mathbb{P}^{out}$ could be the sketch domain.

The L.H.S. of the above equation is the expected cross entropy loss on the out-domain distribution achieved by individual models, i.e., when we train an individual model on a particular instance of the training dataset $\mathcal{T}$, the expected out-domain test loss is denoted by L.H.S. Importantly, the Bias term on the R.H.S. denotes the expected cross entropy loss on the out-domain distribution achieved by the function $\bar{f}(.)$, which is essentially an *ensemble*. Finally, the variance term captures how much the

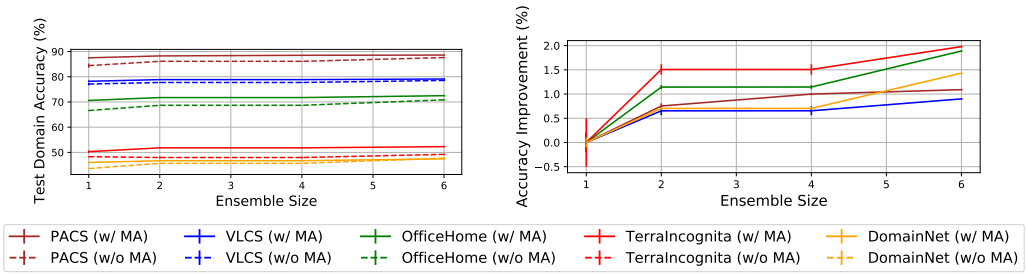

Figure 3: **Left**: Effect of ensemble size (number of models in an ensemble) on out-domain performance (mean and standard error) for models with and without moving average (MA) parameters for ResNet-50 pre-trained on ImageNet. **Right**: Using the performance of ensemble of size 1 (shown in the left plot) as reference, right plot shows the percentage point improvement for ensembles of size > 1. The plots show that i) ensemble of averages (solid lines in left plot) are consistently better than ensemble of models without averaging (dashed lines in left plot); ii) ensemble of averages consistently improves performance over averaged models (ensemble of size 1 in right plot).

prediction of individual models differs in expectation from the ensemble prediction, which makes this term strictly greater than zero.

Therefore, the above decomposition tells us that the expected test domain error of an ensemble is strictly less than that of an individual model. This interpretation directly explains why a traditional ensemble of unaveraged models can be expected to perform better than individual unaveraged models. However, it is still not clear why EoA performs better that a traditional ensemble in practice. To establish this connection, we note that in practice, we typically train a small number of independent models to form a traditional ensemble due to computational constraints. Thus such ensembles do not behave identically to the expected ensemble $\bar{f}(.)$ described above. Model averaging on the other hand has been shown to approximate an ensemble [23]. To see this, consider without any loss of generality that the ensemble contains models with parameters $\{\theta_1, \theta_2 \dots \theta_T\}$, and denote $\hat{\theta}_T := \frac{1}{T} \cdot \sum_{t=1}^{T} \theta_t$. Then note that the second order Taylor's expansion around $\hat{\theta}_T$ of each model's k^{th} dimension's prediction is given by,

$$\frac{1}{T} \cdot \sum_{t=1}^{T} f(\theta_t)_k \approx f(\hat{\theta}_T)_k + \frac{1}{T} \cdot \sum_{t=1}^{T} (\hat{\theta}_T - \theta_t)^T \frac{\partial f(\hat{\theta}_T)_k}{\partial \hat{\theta}_T} + 0.5(\hat{\theta}_T - \theta_t)^T \frac{\partial^2 f(\hat{\theta}_T)_k}{\partial \hat{\theta}_T^2} (\hat{\theta}_T - \theta_t)$$

Notice that $f(.)$ is the model output and therefore the first and second order terms are the derivatives of the model output and not the loss gradient and Hessian. The first order term is zero due to $\hat{\theta}_T := \frac{1}{T} \cdot \sum_{t=1}^{T} \theta_t$. A crucial difference of our analysis compared to [23] is that they average model states that lie near different loss minima, while we perform tail averaging. Therefore, the term $(\hat{\theta}_T - \theta_t)$ may not behave similar to that in their case. To shed light on its behavior, we plot the histogram of the second order term and the moving average model's logit $f(\hat{\theta}_T)_k$ in Eq. 3 for the first dimension ($k = 1$) for test domain data in figure 4 (details and additional experiments provided in Appendix D). The histogram shows that the second order term concentrates near zeros while the logit values span a wider range, which implies that under the second order approximation, the model averaging protocol used in our work behaves like an ensemble. Finally, to study the impact of ensemble size on out-domain performance, we plot the test domain accuracy as a function of ensemble size in figure 3. The plots show that i. EoA outperforms traditional ensembles for all ensemble sizes (left); and ii. ensembles of larger size typically have better

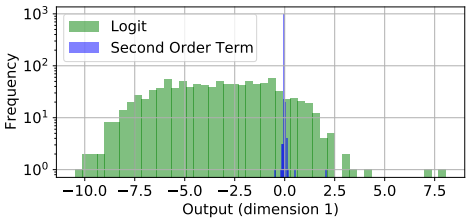

Figure 4: The scale of terms– moving average model's logit and the second order term in Eq. 3. The latter concentrates around 0, suggesting our model averaging protocol approximates ensembles.

Table 4: Performance benchmarking on 5 datasets of the DomainBed benchmark using two different pre-trained models. SWAD and MIRO are the previous SOTA. **See Table 10 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 (25M Parameters, Pre-trained on ImageNet)						
ERM (our runs)	84.4 $\pm$ 0.8	77.1 $\pm$ 0.5	66.6 $\pm$ 0.2	48.3 $\pm$ 0.2	43.6 $\pm$ 0.1	64.0
Ensemble (our runs)	87.6	78.5	70.8	49.2	**47.7**	66.8
ERM [18]	85.7 $\pm$ 0.5	77.4 $\pm$ 0.3	67.5 $\pm$ 0.5	47.2 $\pm$ 0.4	41.2 $\pm$ 0.2	63.8
SWAD [8]	88.1 $\pm$ 0.4	**79.1 $\pm$ 0.4**	70.6 $\pm$ 0.3	50.0 $\pm$ 0.4	46.5 $\pm$ 0.2	66.9
MIRO [9]	85.4 $\pm$ 0.4	79.0 $\pm$ 0.	70.5 $\pm$ 0.4	50.4 $\pm$ 1.1	44.3 $\pm$ 0.2	65.9
SMA (ours)	87.5 $\pm$ 0.2	78.2 $\pm$ 0.2	70.6 $\pm$ 0.1	50.3 $\pm$ 0.5	46 $\pm$ 0.1	66.5
EoA (ours)	**88.6**	**79.1**	**72.5**	**52.3**	47.4	**68.0**
ResNeXt-50 32x4d [48] (25M Parameters, Pre-trained 1B Images)						
ERM (our runs)	88.9 $\pm$ 0.3	79.0 $\pm$ 0.1	70.9 $\pm$ 0.5	51.4 $\pm$ 1.2	48.1 $\pm$ 0.2	67.7
Ensemble (our runs)	91.2	80.3	77.8	53.5	52.8	71.1
SMA (ours)	92.7 $\pm$ 0.3	79.7 $\pm$ 0.3	78.6 $\pm$ 0.1	53.3 $\pm$ 0.1	53.5 $\pm$ 0.1	71.6
EoA (ours)	**93.2**	**80.4**	**80.2**	**55.2**	**54.6**	**72.7**
RegNetY-16GF [40] (81M Parameters, Pre-trained on 3.6B Images)						
ERM (our runs)	92 $\pm$ 0.4	78.6 $\pm$ 0.6	73.8 $\pm$ 0.5	55.6 $\pm$ 0.9	53.1 $\pm$ 0.2	70.6
Ensemble (our runs)	95.1	80.6	80.5	59.5	57.8	74.7
ERM [9]	89.6 $\pm$ 0.4	78.6 $\pm$ 0.3	71.9 $\pm$ 0.6	51.4 $\pm$ 1.8	48.5 $\pm$ 0.6	68.0
SWAD [9]	94.7 $\pm$ 0.2	79.7 $\pm$ 0.2	80.0 $\pm$ 0.1	57.9 $\pm$ 0.7	53.6 $\pm$ 0.6	73.2
MIRO [9]	**97.4 $\pm$ 0.2**	79.9 $\pm$ 0.6	80.4 $\pm$ 0.2	58.9 $\pm$ 1.3	53.8 $\pm$ 0.1	70.1
SMA (ours)	95.5 $\pm$ 0.0	80.7 $\pm$ 0.1	82.0 $\pm$ 0.0	59.7 $\pm$ 0.0	60.0 $\pm$ 0.0	75.6
EoA (ours)	**95.8**	**81.1**	**83.9**	**61.1**	**60.9**	**76.6**

out-domain performance. See a discussion on functional diversity of ensembles vs model averaging in Appendix E.

4 Empirical Results

4.1 DomainBed Benchmarking

We now benchmark our model averaging protocol (SMA) 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 [27], VLCS [13], OfficeHome [45], TerraIncognita [3] and DomainNet [35] datasets in DomainBed. The training-evaluation protocols are the same as described in section 2.3 for moving average and online models, and in section 3 for ensembles. Full details can be found in section B in the Appendix.

Comparison with existing results using ResNet-50 pre-trained on ImageNet: Here we compare existing methods with our runs. All methods use ResNet-50 (25M parameters) [19] pre-trained on ImageNet as initialization. Comparing ERM [18] and ERM (our runs), we find that they perform similarly, especially considering we have used a smaller hyper-parameter space (further discussion in Appendix E). A comparison between SWAD and SMA 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. Interestingly, traditional ensembles and SMA achieve similar performance (66.8% and 66.5% respectively). 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 [18], EoA test accuracy is achieved by a single predictor that combines the output of multiple models.

Experiments with larger pre-training datasets and larger models: In addition to ResNet-50 pre-trained on ImageNet, we now also experiment with ResNeXt-50 32x4d (25M parameters), that is pre-trained using semi-weakly supervised objective on Instagram 1B images and ImageNet labeled data [48], and RegNetY-16GF (81M parameters) pre-trained using Instagram 3.6B images. Note that both ResNet-50 and ResNeXt-50 32x4d have similar number of parameters, while RegNetY-16GF has more than 3x the number of parameters. On the other hand, also notice that the three architectures are respectively pre-trained on an increasing size of datasets. The rationale behind this choice is that

recent trends in deep learning has shown that models pre-trained on larger datasets and architectures achieve better downstream transfer performance [12, 32, 20]. Therefore, we expect the latter models to improve the ERM baseline, and our goal is to investigate the out-domain performance gain by model averaging and EoA relative to the *corresponding* ERM baseline with increasing pre-training dataset size and model size.

The experimental results are shown in Table 4. To investigate models with the same size, but one pre-trained on a larger dataset, we compare the results of ResNet-50 and ResNeXt-50 32x4d. On average across all five datasets, the gain of SMA over ERM (our runs) is 2.5% for ResNet-50 and 3.9% for ResNeXt-50 32x4d. The gain of EoA over ERM is larger: 4% vs 5% respectively. This suggests that pre-training the model on a larger dataset increases the gain of model averaging and EoA over the *corresponding* ERM baseline, while the ERM performance itself improves.

Next, to investigate the impact of both larger model size and larger pre-training dataset, we compare the results of ResNeXt-50 32x4d and RegNetY-16GF. On average across all five datasets, the gain of SMA over ERM (our runs) is 3.9% for ResNeXt-50 32x4d and 5% for RegNetY-16GF. The gain of EoA over ERM is again larger: 5% vs 6% respectively. This suggests that increasing both model size and pre-training dataset size allow model averaging and EoA to provide larger out-domain gains over the corresponding ERM baseline. Notice that these claims are different from existing work [20], which states that the baseline ERM performance improves with larger pre-training data and model size.

4.2 In-domain Performance Improvement using Model Averaging

We study the in-domain test accuracy on PACS and OfficeHome datasets using ImageNet pre-trained ResNet-50 with and without our SMA protocol. In this experiment, we combine all the domains of PACS and split it into training/-validation/test splits (0.8/0.1/0.1). We run 10 different runs with different seeds and randomly chosen splits for each dataset. The best model for each run is chosen using the validation set. The remaining optimization details are identical to those used in the previous section. The test accuracy mean and standard error using these best models are shown in Table 5. As expected, SMA outperforms models without averaging.

Table 5: SMA outperforms ERM without model averaging in the IID setting.

Algorithm	PACS	OfficeHome
ERM (no averaging)	94.39 ± 0.46	77.09 ± 0.57
SMA (ours)	$\mathbf{96.77} \pm \mathbf{0.20}$	$\mathbf{83.56} \pm \mathbf{0.21}$

5 Related Work

5.1 Model Averaging

A theoretical perspective: In our model averaging protocol, we compute a simple moving average of the model parameters starting early during training. This is known as *tail-averaging* [24], which is slightly different from Polyak-Ruppert averaging [36] in that the latter starts averaging from the very beginning of training. In the context of least square regression in the IID setting, [24] 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 [34] similar to that of Tikhonov regularization [42]. 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 [10], 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 [8]: SWAD propose flat minima as a means for improving domain generalization. Following the intuition of stochastic weight averaging (SWA, [23]), 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. This is done to prevent including an under-performing 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 instead of finding the start and end period for model averaging meticulously, 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 SWAD.

5.2 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 [41, 30, 39, 37]. There are different variants of this idea, such as minimizing some divergence metric between the latent representation of different domains (E.g. DANN [16]), or less strictly, minimizing the difference between the latent statistics of different domains (E.g. DICA [33], CORAL [41]). In the meta learning category, source domains are typically split into 2 subsets to be used as the training and test domains in episodes to simulate the domain generalization setting [28, 29]. 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 [50, 52]) to self-supervised learning involving matching the representations of an image with different augmentations (E.g. [1, 7]). Finally, different ways of regularizing models (implicit and explicit) have also been developed with the goal of encouraging domain-invariant feature learning [38, 47, 46]. For instance, invariant risk minimization [2] propose a regularization such that the classifier is optimal in all the environments. Representation Self-Challenging [21] 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 [26] 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 [51] 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.

6 Conclusion

We investigated a hyperparameter-free and efficient protocol for model averaging 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 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. We experimented with three pre-trained models with increasing pre-training dataset and model size, and found that EoA provides a proportionally larger gain compared to the *corresponding* ERM baseline, and lies in the range of $4\% - 6\%$. Finally, we explain the boost in performance by EoA using the Bias-Variance trade-off perspective. Further discussions along with limitations of our work are provided in Appendix E.

References

[1] Isabela Albuquerque, Nikhil Naik, Junnan Li, Nitish Keskar, and Richard Socher. Improving out-of-distribution generalization via multi-task self-supervised pretraining. *arXiv preprint arXiv:2003.13525*, 2020.

[2] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[3] 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.

[4] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

[5] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 24:2178–2186, 2011.

[6] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[7] Silvia Bucci, Antonio D'Innocente, Yujun Liao, Fabio Maria Carlucci, Barbara Caputo, and Tatiana Tommasi. Self-supervised learning across domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[8] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. *arXiv preprint arXiv:2102.08604*, 2021.

[9] Junbum Cha, Kyungjae Lee, Sungrae Park, and Sanghyuk Chun. Domain generalization by mutual-information regularization with pre-trained models. *arXiv preprint arXiv:2203.10789*, 2022.

[10] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

[11] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[13] 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.

[14] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[15] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[16] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[17] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[18] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021.

[21] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 124–140. Springer, 2020.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[23] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[24] Prateek Jain, Sham Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18, 2018.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

[27] 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.

[28] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1446–1455, 2019.

[30] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[31] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 624–639, 2018.

[32] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196, 2018.

[33] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013.

[34] Gergely Neu and Lorenzo Rosasco. Iterate averaging as regularization for stochastic gradient descent. In *Conference On Learning Theory*, pages 3222–3242. PMLR, 2018.

[35] 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.

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

[37] Alexandre Rame, Corentin Dancette, and Matthieu Cord. Fishr: Invariant gradient variances for out-of-distribution generalization. *arXiv preprint arXiv:2109.02934*, 2021.

[38] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[39] Yuge Shi, Jeffrey Seely, Philip HS Torr, N Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937*, 2021.

[40] Mannat Singh, Laura Gustafson, Aaron Adcock, Vinicius de Freitas Reis, Bugra Gedik, Raj Prateek Kosaraju, Dhruv Mahajan, Ross Girshick, Piotr Dollár, and Laurens van der Maaten. Revisiting weakly supervised pre-training of visual perception models. *arXiv preprint arXiv:2201.08371*, 2022.

[41] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[42] Andrey Nikolayevich Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198, 1943.

[43] Vladimir Vapnik and Vlamimir Vapnik. Statistical learning theory wiley. *New York*, 1(624):2, 1998.

[44] Ramakrishna Vedantam, David Lopez-Paz, and David J Schwab. An empirical investigation of domain generalization with empirical risk minimizers. *Advances in Neural Information Processing Systems*, 34, 2021.

[45] 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.

[46] Yufei Wang, Haoliang Li, and Alex C Kot. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3622–3626. IEEE, 2020.

[47] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6502–6509, 2020.

[48] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

[49] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. In *International Conference on Machine Learning*, pages 10767–10777. PMLR, 2020.

[50] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. 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*, pages 2100–2110, 2019.

[51] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *arXiv preprint arXiv:2103.02503*, 2021.

[52] Kaiyang Zhou, Chen Change Loy, and Ziwei Liu. Semi-supervised domain generalization with stochastic stylematch. *arXiv preprint arXiv:2106.00592*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Appendix E, we have discussed various aspects including the limitations of our and existing methods in addressing domain generalization, functional diversity of ensembles vs model averaging strategy, and more.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix A.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See section 3.2.
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Table 6: Hyper-parameter search space for all experiments.

Hyper-parameter	**Default value**	**Random distribution**	
		[18]	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	$10^{\text{Uniform}(-6,-2)}$	$10^{\text{Uniform}(-6,-4)}$

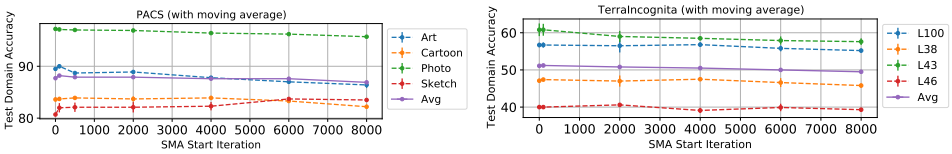

Figure 5: The impact of iteration t_0 at which we start simple moving averaging as described in Eq. 1, on the domain generalization performance for PACS and TerraIncognita datasets. The dominant pattern across all the experiments suggests that starting averaging earlier yields a stronger boost in performance.

Appendix

A Broader Impact

Our work aims at improving out-domain performance of models. Thus it is naturally geared towards mitigating the effects of training dataset biases on the hypothesis learned by the model, which we believe has a positive societal impact.

B Training and Evaluation Protocols for DomainBed Benchmarking

We use the training protocol described in [18] with minor changes: we use a smaller hyper-parameter search space (shown in Table 6) and smaller number of random trials for computational reasons, and train on DomainNet dataset for $15,000$ iterations instead of $5,000$ similar to [8], 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 3.

All models are trained using the ERM objective and optimized using the Adam optimizer [25]. We use ResNet-50 [19] 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 [48]. This model was downloaded from Pytorch hub. For all models, the batch normalization [22] 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 [18] with $80\% - 20\%$ training-validation split, and the average out-domain test performance is reported across all runs for each domain.

All experiments were performed on Google Could Platform (GCP) using 24 NVIDIA A100 GPUs.

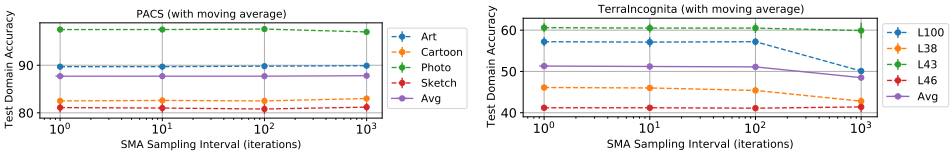

Figure 6: The impact of the frequency (number of iterations), at which model states are sampled for computing the simple moving average (SMA), on the domain generalization performance for PACS and TerraIncognita datasets. Broadly, we find that the frequency of sampling does not have a major influence 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.

C Additional Ablation Analysis of Our Model Averaging Protocol (Eq. 1)

C.1 Dataset and Experiment Details for SMA Ablation Analysis in Section 2.3, C.2, and C.3

Experimentation Details: We use the training protocol described in [18] 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 [8], 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 [18], where the average out-domain test performance is reported across all runs. For more details, see section B 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).

C.2 Start Iteration

We investigate how domain generalization performance is impacted by the choice of iteration t_0 when we start model averaging in Eq. 1. 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 5 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, and choosing t_0 close to initialization 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 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 called *tail averaging*, and the theoretical motivation for this choice are discussed in more detail in section 5.1.

C.3 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 $\{10^0, 10^1, 10^2, 10^3\}$ iterations. Test accuracy is once again computed using the protocol of [18] for each case. The performance as a function of the iterate sampling interval used in SMA is shown in Figure 6. 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.

C.4 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, instead of Pearson's correlation, 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 (early stopping). 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.

C.5 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 7 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 12 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.

C.6 Cross-run rank correlation

In addition to the experiments in 2.3.1, 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 8 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.

D Why Does Ensembling Improve Performance?

We describe the histogram experiment presented in section 3.2. For this experiment, we use the TerraIncognita dataset with L46 as the test domain. We use one of the runs with model averaging from the experiments done in section 4.1. We begin by re-writing the Taylor's expansion more precisely for our model averaging protocol,

$$\frac{1}{T-t_0+1}\cdot\sum_{t=t_0}^{T} f(\mathbf{x};\theta_t)_k \approx f(\mathbf{x};\hat{\theta}_T)_k + 0.5\cdot\frac{1}{T-t_0+1}\cdot\sum_{t=t_0}^{T}(\hat{\theta}_T-\theta_t)^T\frac{\partial^2 f(\mathbf{x};\hat{\theta}_T)_k}{\partial\hat{\theta}_T^2}(\hat{\theta}_T-\theta_t) \tag{3}$$

where $\hat{\theta}_T := \frac{1}{T-t_0+1}\cdot\sum_{t=t_0}^{T}\theta_t$. Notice the first order term has been omitted since it is zero. As an important detail, instead of computing the second order term in the above equation for all

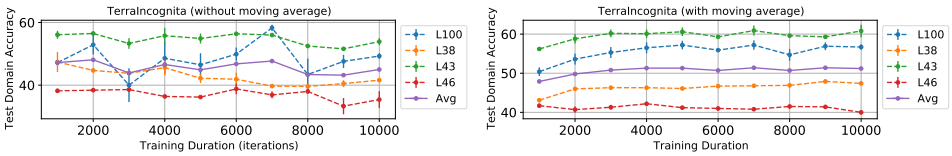

Figure 7: Qualitatively accessing the reliability of model selection while varying the training duration. For each training duration, the out-domain test accuracy is calculated using model selection over the in-domain validation data. Not using model averaging leads to unreliable model selection as evident in the instability of the out-domain performance. Model averaging is able to reduce this instability.

Table 7: 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 individual models (left) and ensembles (right). In most cases, using model averaging results in a significantly better rank correlation, which makes model selection more reliable.

Table 8: Individual Models

PACS	w/o avg	with avg
Art	0.31 ± 0.04	**0.62 ± 0.04**
Cartoon	0.25 ± 0.10	**0.52 ± 0.03**
Photo	**0.09 ± 0.07**	-0.38 ± 0.15
Sketch	0.24 ± 0.06	**0.53 ± 0.06**
TerraIncognita	w/o avg	with avg
L100	0.21 ± 0.07	**0.90 ± 0.05**
L38	0.12 ± 0.13	**0.83 ± 0.05**
L43	0.30 ± 0.06	**0.67 ± 0.18**
L46	0.03 ± 0.11	**0.52 ± 0.14**
VLCS	w/o avg	with avg
Caltech101	**0.21 ± 0.10**	0.16 ± 0.15
LabelMe	**0.30 ± 0.08**	0.02 ± 0.14
Sun09	0.27 ± 0.12	**0.32 ± 0.11**
VOC2007	0.17 ± 0.11	**0.38 ± 0.05**
OfficeHome	w/o avg	with avg
Art	0.05 ± 0.11	**0.80 ± 0.04**
Clipart	0.33 ± 0.04	**0.84 ± 0.04**
Product	0.61 ± 0.04	**0.80 ± 0.04**
RealWorld	0.41 ± 0.06	**0.74 ± 0.04**
DomainNet	w/o avg	with avg
Clip	0.96 ± 0.01	**1 ± 0**
Info	0.80 ± 0.05	**1 ± 0**
Paint	0.87 ± 0.02	**1 ± 0**
Quick	0.65 ± 0.04	**1 ± 0**
Real	0.91 ± 0.01	**1 ± 0**
Sketch	0.82 ± 0.04	**1 ± 0**

Table 9: Ensembles

PACS	w/o avg	with avg
Art	0.06	**0.78**
Cartoon	0.33	**0.81**
Photo	**-0.12**	-0.52
Sketch	0.43	**0.70**
TerraIncognita	w/o avg	with avg
L100	0.48	**1**
L38	0.17	**0.95**
L43	**0.59**	0.38
L46	0.08	**0.61**
VLCS	w/o avg	with avg
Caltech101	0.52	**0.81**
LabelMe	0.05	**0.38**
Sun09	0.63	**0.82**
VOC2007	0.55	**0.65**
OfficeHome	w/o avg	with avg
Art	0.27	**0.92**
Clipart	0.66	**0.95**
Product	0.20	**0.95**
RealWorld	0.09	**0.78**
DomainNet	w/o avg	with avg
Clip	**1**	**1**
Info	0.88	**1**
Paint	0.98	**1**
Quick	0.95	**1**
Real	0.97	**1**
Sketch	0.97	**1**

integer valued t between t_0 and T, we only use θ_t for $t = 300 * i$ for $i \in \{1, 2, \ldots, 16\}$ due to computational constraints. This should not affect our conclusion because as shown in section C.3, averaging frequency does not have a significant impact on performance. Finally, we record the logit values and the second order term for 1000 randomly selected samples $\mathbf{x}$ from the test domain data. Note that TerraIncognita has 10 classes and so we have the above equation for $k \in \{1, 2, \ldots, 10\}$. We plot the histogram of the logit values and the second order term corresponding to each class's output separately. All the plots are shown in figure 9. The conclusion is the same as described in the main text.

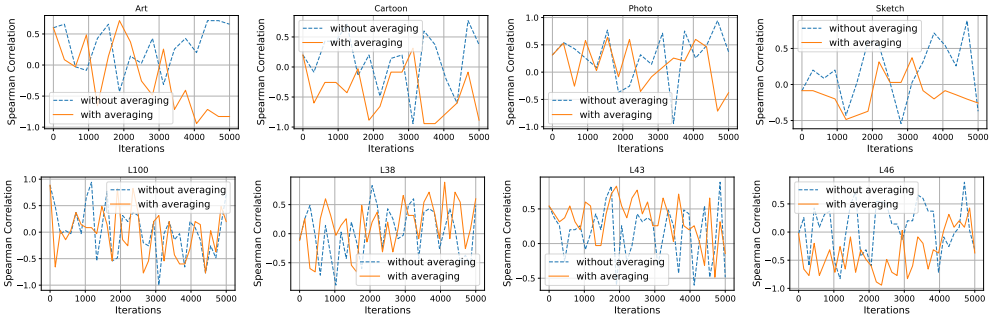

Figure 8: Spearman correlation between *cross-run* in-domain validation accuracy and out-domain test accuracy for PACS dataset (top) and TerraIncognita dataset (bottom). The cross-run rank correlations are poor (not consistently close to 1) for both online model (without avg) and moving average model. 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 2.3.1 for details.

E Discussions and Limitations

Domain Generalization Limitations: The Bias-Variance trade-off analysis of EoA in section 3.2 shows that the expected loss of individual models constitutes both the bias and variance term, while that of ensembles is dominated by the bias term alone, and is thus strictly lower. Thus ensembles (either explicitly or implicitly by model averaging) is guaranteed to improve performance on OOD data compared to the corresponding individual unaveraged models. However, this strategy cannot go beyond getting rid of the variance term, and other strategies need to be used to reduce the bias term, which will further improve the OOD performance.

Additionally, our proposal does not make use of the environment ID of samples. A popular strategy to utilize this information is to increase some form of alignment between the latent representations of different domains, which has been shown to be one of the terms in the upper bound of the test domain generalization error [4]. While a number of prior work has proposed variants of this strategy (eg. CORAL, DANN, see section 5.2 for a discussion), [18] has showed that ERM with appropriate training-validation protocol performs at least as well as these methods. Further, [44] has recently argued supported by empirical evidence that domain alignment is neither necessary nor sufficient for domain generalization. Thus it remains an open question what other strategies can be used to utilize the environment ID to boost domain generalization

Functional 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, though it is smaller on average compared with online models (see training evolution plots in appendix for reference). On another note, while our analysis in section 3.2 suggests that model averaging behaves as an ensemble, we believe that it does not offer as much of functional diversity as independently trained models. This is because if it did, model averaging should have had a much better performance compared with a traditional ensembles (since there are $T - t_0 + 1$ models in model averaging while only 6 models in the traditional ensemble in our experiments), but this is not the case (see their performance in table 10 for a comparison). This implies that there is still diversity among the independently trained models with model averaging. This is also inline with [14] which shows that ensembling methods such as model averaging and Monte Carlo dropout [15] do not provide diversity in function space as much as ensembles of independently trained models. Perhaps this is why EoA performs better compared to both individual moving average models and traditional ensembles, by better approximating the expected ensemble behavior.

Scalability: Following the protocol of [18], 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.

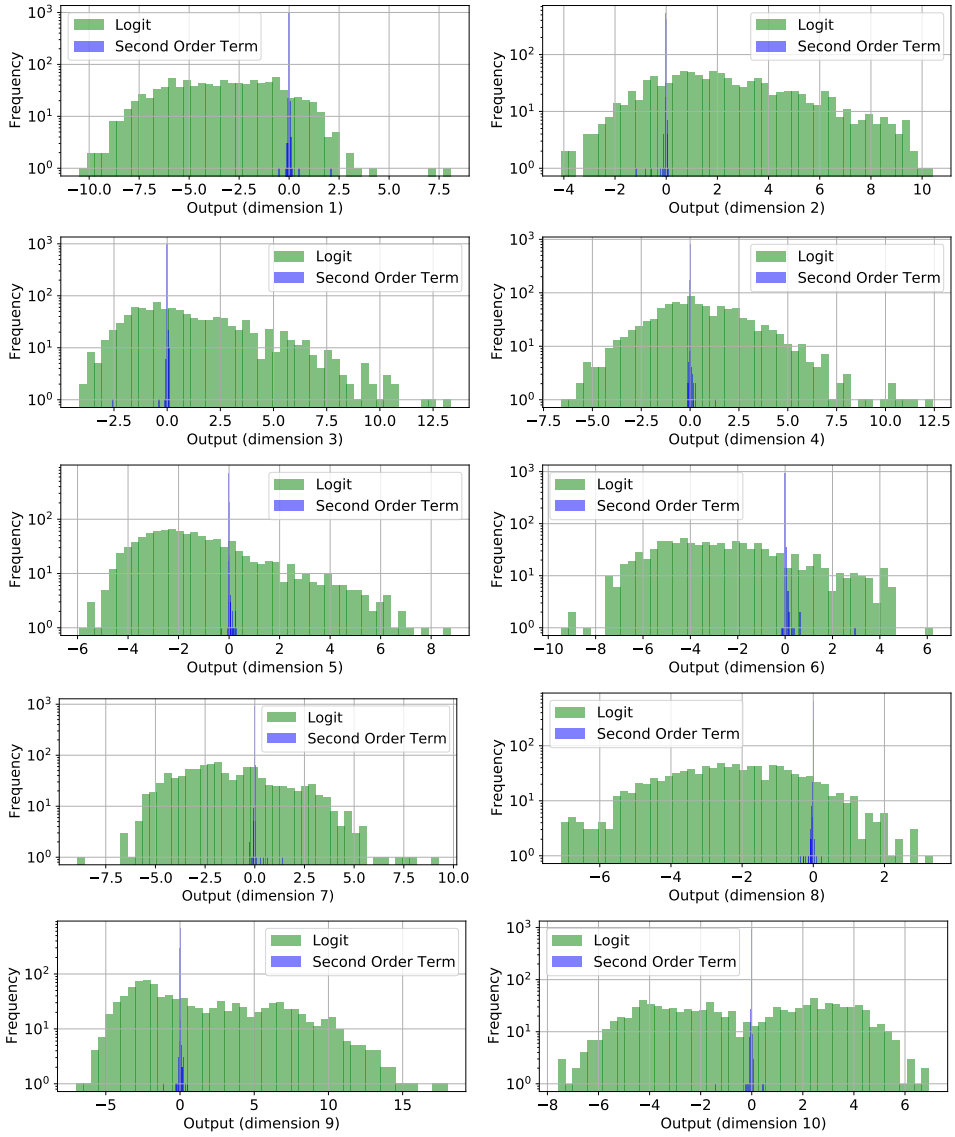

Figure 9: The scale of terms– moving average model's logit and the second order term in Eq. 3. The latter concentrates around 0, suggesting our model averaging protocol approximates ensembles.

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 C.2 and C.3 respectively. For both of them, we proposed to fix their values universally instead of tuning them on each dataset. Specifically, [24] 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, [24] 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 [18] for training and evaluation.

Table 10: 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 (our runs)	84.4 $\pm$ 0.8	77.1 $\pm$ 0.5	66.6 $\pm$ 0.2	48.3 $\pm$ 0.2	43.6 $\pm$ 0.1	64.0
Ensemble (our runs)	87.6	78.5	70.8	49.2	**47.7**	66.8
ERM [18]	85.7 $\pm$ 0.5	77.4 $\pm$ 0.3	67.5 $\pm$ 0.5	47.2 $\pm$ 0.4	41.2 $\pm$ 0.2	63.8
IRM [2]	84.4 $\pm$ 1.1	78.1 $\pm$ 0.0	66.6 $\pm$ 1.0	47.9 $\pm$ 0.7	35.7 $\pm$ 1.9	62.5
Group DRO [38]	84.1 $\pm$ 0.4	77.2 $\pm$ 0.6	66.9 $\pm$ 0.3	47.0 $\pm$ 0.3	33.7 $\pm$ 0.2	61.8
Mixup [47, 46]	84.3 $\pm$ 0.5	77.7 $\pm$ 0.4	69.0 $\pm$ 0.1	48.9 $\pm$ 0.8	39.6 $\pm$ 0.1	63.9
MLDG [28]	84.8 $\pm$ 0.6	77.1 $\pm$ 0.4	68.2 $\pm$ 0.1	46.1 $\pm$ 0.8	41.8 $\pm$ 0.4	63.6
CORAL [41]	86.0 $\pm$ 0.2	77.7 $\pm$ 0.5	68.6 $\pm$ 0.4	46.4 $\pm$ 0.8	41.8 $\pm$ 0.2	64.1
MMD [30]	85.0 $\pm$ 0.2	76.7 $\pm$ 0.9	67.7 $\pm$ 0.1	49.3 $\pm$ 1.4	39.4 $\pm$ 0.8	63.6
DANN [16]	84.6 $\pm$ 1.1	78.7 $\pm$ 0.3	65.4 $\pm$ 0.6	48.4 $\pm$ 0.5	38.4 $\pm$ 0.0	63.1
C-DANN [31]	82.8 $\pm$ 1.5	78.2 $\pm$ 0.4	65.6 $\pm$ 0.5	47.6 $\pm$ 0.8	38.9 $\pm$ 0.1	62.6
Fish [39]	85.5 $\pm$ 0.3	77.8 $\pm$ 0.3	68.6 $\pm$ 0.4	45.1 $\pm$ 1.3	42.7 $\pm$ 0.2	63.9
Fishr [37]	85.5 $\pm$ 0.4	77.8 $\pm$ 0.1	67.8 $\pm$ 0.1	47.4 $\pm$ 1.6	41.7 $\pm$ 0.0	65.7
SWAD [8]	88.1 $\pm$ 0.4	**79.1 $\pm$ 0.4**	70.6 $\pm$ 0.3	50.0 $\pm$ 0.4	46.5 $\pm$ 0.2	66.9
MIRO [9]	85.4 $\pm$ 0.4	79.0 $\pm$ 0.	70.5 $\pm$ 0.4	50.4 $\pm$ 1.1	44.3 $\pm$ 0.2	65.9
SMA (ours)	87.5 $\pm$ 0.2	78.2 $\pm$ 0.2	70.6 $\pm$ 0.1	50.3 $\pm$ 0.5	46 $\pm$ 0.1	66.5
EoA (ours)	**88.6**	**79.1**	**72.5**	**52.3**	47.4	**68.0**
ResNeXt-50 32x4d [48] (25M Parameters, Pre-trained 1B Images)						
ERM (our runs)	88.9 $\pm$ 0.3	79.0 $\pm$ 0.1	70.9 $\pm$ 0.5	51.4 $\pm$ 1.2	48.1 $\pm$ 0.2	67.7
Ensemble (our runs)	91.2	80.3	77.8	53.5	52.8	71.1
SMA (ours)	92.7 $\pm$ 0.3	79.7 $\pm$ 0.3	78.6 $\pm$ 0.1	53.3 $\pm$ 0.1	53.5 $\pm$ 0.1	71.6
EoA (ours)	**93.2**	**80.4**	**80.2**	**55.2**	**54.6**	**72.7**
RegNetY-16GF [40] (81M Parameters, Pre-trained on 3.6B Images)						
ERM (our runs)	92.0 $\pm$ 0.4	78.6 $\pm$ 0.6	73.8$\pm$ 0.5	55.6 $\pm$ 0.9	53.2 $\pm$ 0.2	70.6
Ensemble (our runs)	95.1	80.6	80.5	59.5	57.8	74.7
ERM [9]	89.6 $\pm$ 0.4	78.6 $\pm$ 0.3	71.9 $\pm$ 0.6	51.4 $\pm$ 1.8	48.5 $\pm$ 0.6	68.0
SWAD [9]	94.7 $\pm$ 0.2	79.7 $\pm$ 0.2	80.0 $\pm$ 0.1	57.9 $\pm$ 0.7	53.6 $\pm$ 0.6	73.2
MIRO [9]	**97.4 $\pm$ 0.2**	79.9 $\pm$ 0.6	80.4 $\pm$ 0.2	58.9 $\pm$ 1.3	53.8 $\pm$ 0.1	70.1
SMA (ours)	95.5$\pm$ 0.0	80.7 $\pm$ 0.1	82.0 $\pm$ 0.0	59.7 $\pm$ 0.0	60.0 $\pm$ 0.0	75.6
EoA (ours)	**95.8**	**81.1**	**83.9**	**61.1**	**60.9**	**76.6**

Smaller HP search space: We use a smaller hyper-parameter search space compared with that in [18]. 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 [18] 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 4, where using ResNeXt-50 32x4d pre-trained on a larger dataset [48] 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.

Table 11: Out-domain accuracy for PACS dataset.

Algorithm	**A**	**C**	**P**	**S**	**Avg.**
ResNet-50					
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 [48]					
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**
RegNetY-16GF [40]					
ERM	90.2 ± 0.6	92.6 ± 0.8	97.6 ± 0.1	87.8 ± 2	92
Ensemble	93.75	95.35	98.02	93.38	95.1
ERM w/ MA	93.8 ± 0.3	95.8 ± 0.2	99.2 ± 0.2	**93.4 ± 0.2**	95.5
Ensemble of Averages (EoA)	**94.09**	**96.33**	**99.52**	93.31	**95.8**

Table 12: Out-domain accuracy for VLCS dataset.

Algorithm	**C**	**L**	**S**	**V**	**Avg.**
ResNet-50					
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 [48]					
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**
RegNetY-16GF [40]					
ERM	96.0 ± 0.4	66.0 ± 0.8	76.1 ± 0.7	76.2 ± 2.1	78.6
Ensemble	97.88	**67.28**	78.46	78.55	80.6
ERM w/ MA	98.1 ± 0.2	65.7 ± 1.1	79.2 ± 1.1	79.6 ± 0.2	80.7
Ensemble of Averages (EoA)	**98.23**	66.00	**79.49**	**80.63**	**81.1**

Table 13: Out-domain accuracy for OfficeHome dataset.

Algorithm	**A**	**C**	**P**	**R**	**Avg.**
ResNet-50					
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 [48]					
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**
RegNetY-16GF [40]					
ERM	70.7± 1.3	60.0 ± 0.5	82.4 ± 0.5	82.1 ± 0.4	73.8
Ensemble	79.44	68.68	86.28	87.63	80.5
ERM w/ MA	81.1 ± 0.4	72.3 ± 0.6	86.6 ± 0.1	88.2 ± 0.1	82.0
Ensemble of Averages (EoA)	**83.89**	**73.95**	**88.22**	**89.37**	**83.9**

Table 14: Out-domain accuracy for TerraIncognita dataset.

Algorithm	**L100**	**L38**	**L43**	**L46**	**Avg.**
ResNet-50					
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 [48]					
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**
RegNetY-16GF [40]					
ERM	67.1 ± 2.8	46.3 ± 2.9	61.4 ± 0	47.5 ± 1.9	55.6
Ensemble	71.67	50.76	64.11	**51.45**	59.5
ERM w/ MA	72.4 ± 0.0	52.0 ± 0.5	66.8 ± 0.4	47.4 ± 0.2	59.7
Ensemble of Averages (EoA)	**73.80**	**52.60**	**68.19**	49.75	**61.1**

Table 15: Out-domain accuracy for Domainet dataset.

Algorithm	**clip**	**info**	**paint**	**quick**	**real**	**sketch**	**Avg.**
ResNet-50							
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 [48]							
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**
RegNetY-16GF [40]							
ERM	74.7 ± 0.1	34.8± 0.9	60.3 ± 0	15.2 ± 0.6	71.1 ± 0.3	62.1 ± 0.1	53.1
Ensemble	79.12	38.71	65.89	18.13	76.43	68.40	57.8
ERM w/ MA	78.8 ± 0.1	43.0 ± 0.0	68.6 ± 0.0	21.2 ± 0.0	78.5 ± 0.1	69.8 ± 0.1	60
Ensemble of Averages (EoA)	**79.63**	**44.02**	**69.57**	**22.46**	**77.95**	**71.69**	**60.9**

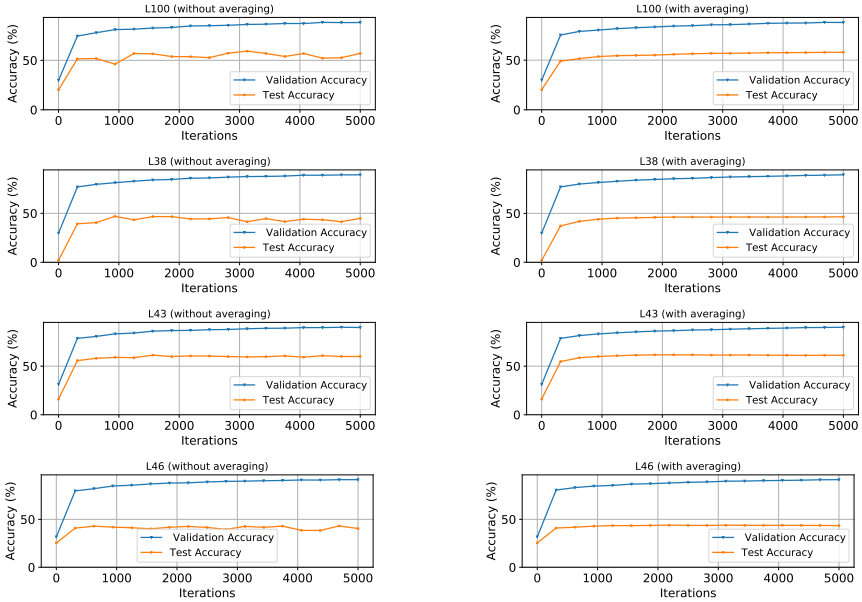

Figure 10: Ensemble of moving average (EOA) models (right) has better out-domain test performance *stability* compared with ensemble of online models (left), w.r.t. in-domain validation accuracy. **Details**: The plots are for the TerraIncognita dataset with the domain name in title used as the test 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.

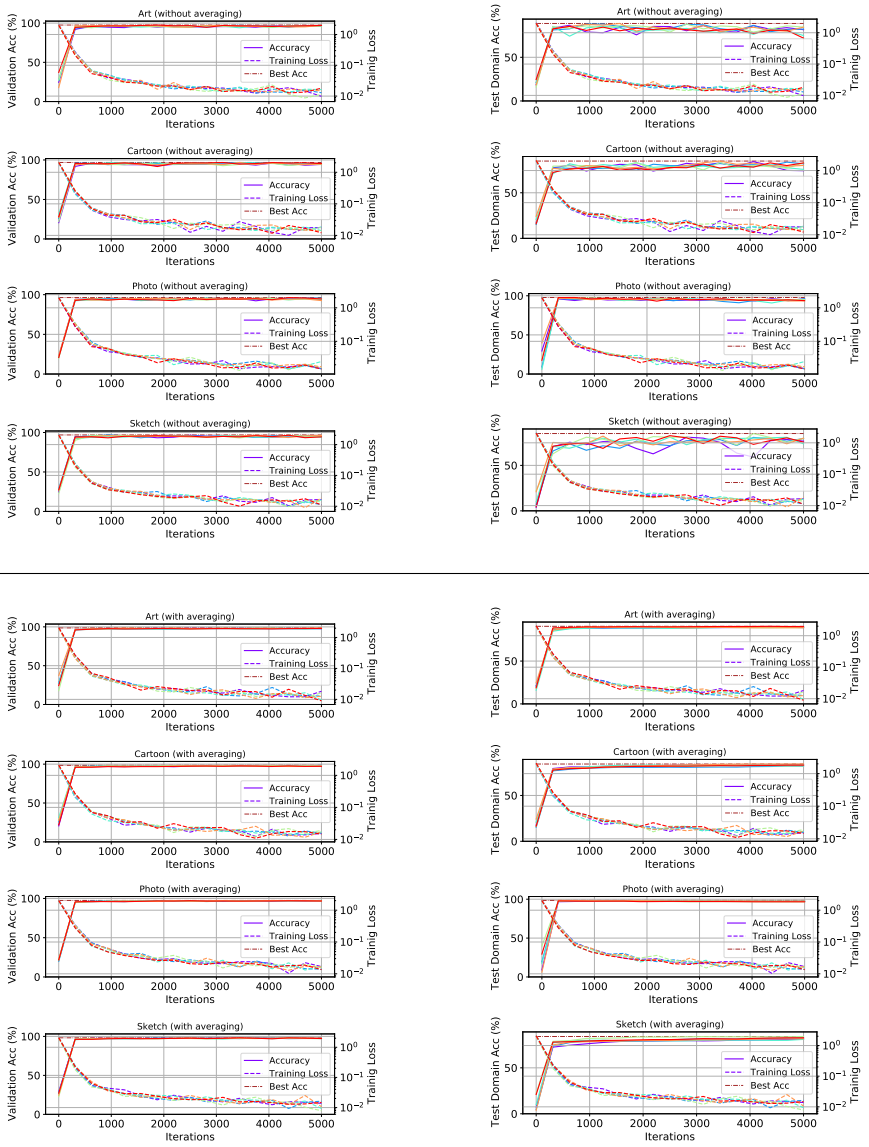

Figure 11: Evolution of training loss, in-domain validation accuracy and out-domain test accuracy for ResNet-50 (pre-trained on ImageNet) trained on PACS without model averaging (top 4 rows) and with model averaging (bottom 4 rows) for $5,000$ iterations with the domain mentioned in the title used as 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 [18].

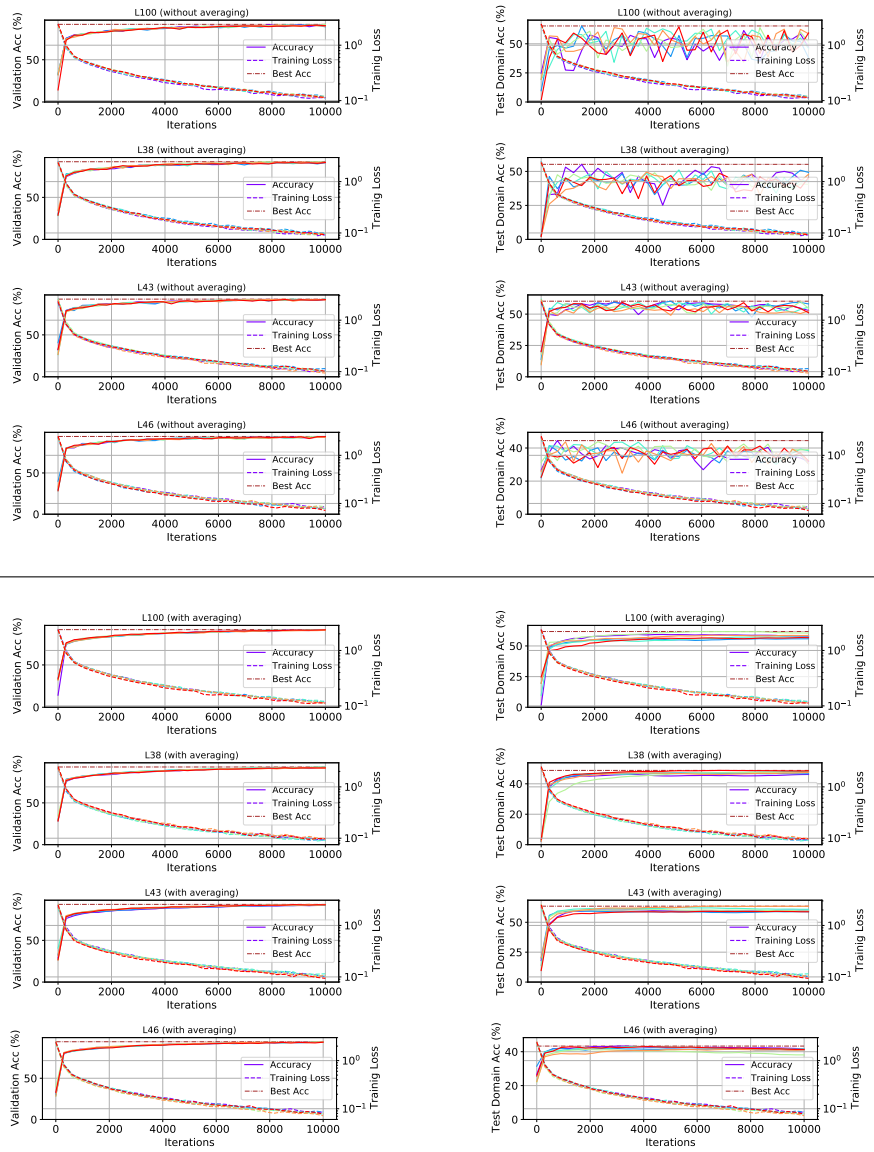

Figure 12: Evolution of training loss, in-domain validation accuracy and out-domain test accuracy for ResNet-50 (pre-trained on ImageNet) trained on TerraIncognita without model averaging (top 4 rows) and with model averaging (bottom 4 rows) for $10,000$ iterations with the domain mentioned in the title 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 [18]. **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.

F 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)
```