MVDG: A Unified Multi-view Framework for Domain Generalization —Appendix—

Jian Zhang[1,2], Lei Qi[3,*], Yinghuan Shi[1,2,*], and Yang Gao[1,2]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, China.
[2] National Institute of Healthcare Data Science, Nanjing University, China.
[3] School of Computer Science and Engineering, Southeast University, China.

A Additional Experiments

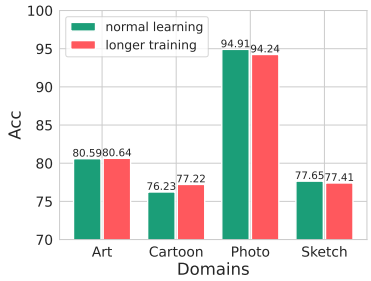


Fig. 1: The comparison of accuracy (%) of normal training and longer training in the meta-learning framework.

Longer training. In our training scheme, we need to train a model with more tasks than traditional meta-learning, resulting in a longer training time. To investigate whether it improves the performance, we train Reptile with a large epoch (i.e., 120 epochs). As shown in Fig. 1, the longer training does not bring any drastic performance gain compared to the original training epochs, which also validates the efficacy of our method.

The effectiveness of re-estimating BN. During the test stage, we note that the test accuracy is unstable, which is caused by the mismatch between BN statistics and model weights. During training, BN first normalizes data with statistics calculated in a batch and then keeps a running average of its statistics, which is used to normalize test images. However, for MVRML, the second step is not accomplished because we only update model weights, leaving BN statistics unchanged, which causes a mismatch. Thus, the performance fluctuates when we apply these mismatched weights and statistics to test images.

A straightforward solution is to replace the statistics in the updated model with temporary model statistics, or we simply forward a current batch of data

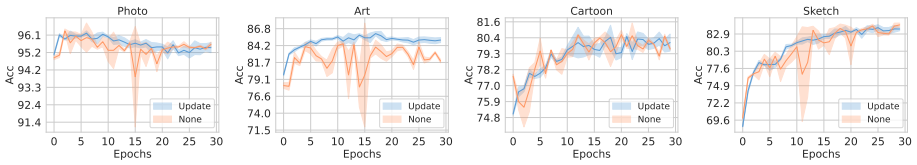


Fig. 2: The test accuracy (%) of model with or without re-estimating BN statistics at eachepoch. The mean and standard deviation are denoted with the solid line and shaded areas, respectively.

to continue accumulating the statistics in the updated model. However, we find that all these operations cannot help stabilize the training procedure. We hypothesize that the updating procedure $\theta_{j+1} = \theta_j + \beta(\theta_{tmp} - \theta_j)$ makes previous BN statistics totally unsuitable for the current weight and only a batch of data or replacement of statistics cannot remedy this effect. Therefore, at the end of the training stage, we need to re-estimate the statistics by forwarding the training set to find suitable statistics.

To visualize the performance fluctuation, we plot the mean (the solid line) and standard deviation (the shaded area) of model accuracy in the target domain. As seen on the orange line of Fig. 2, the performance of the model trained without re-estimating BN statistics fluctuates violently, making it hard to select the best model on the validation set. However, after we re-estimate BN statistics at the end of each epoch, its accuracy becomes more stable, and clear performance gains can be obtained in the "art" domain, as seen on the blue line.

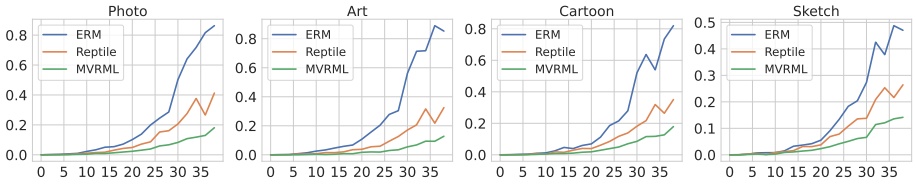


Fig. 3: Local sharpness comparison across ERM, Reptile and MVRML on the *validation* set of PACS. The X-axis indicates the distance γ to the original parameter and Y-axis indicates the sharpness of loss surface (the lower and stable, the more flat).

Local sharpness comparison. As mentioned in Sec. 4.4 in the main page that the sharpness is calculated by the gap between the original parameter and perturbed parameter, i.e., $\mathbb{E}_{\theta'=\theta+\epsilon}[\mathcal{L}(\theta, \mathcal{D}) - \mathcal{L}(\theta', \mathcal{D})]$. We sample the perturbation 10 times from Gaussian distribution with different γ and average the result to produce a stable sharpness value. In addition to the sharpness on the test set shown in Fig. 4 in the main page, we also plot the local sharpness on the

validation set, as shown in Fig. 3. Our method also can find a flatter minimum than DeepAll and Reptile.

Table 1: The influence on accuracy (%) of different augmentation combination of multi-view prediction on PACS dataset. The best performance is marked as **bold**.

crop	flip	jitter	RA	A	C	P	S	Avg.
				85.20	79.97	95.29	83.11	85.89
✓				85.20	79.58	95.54	84.65	86.24
✓	✓			**85.62**	79.98	**95.54**	**85.08**	**86.56**
✓	✓	✓		84.17	**80.13**	94.84	84.28	85.85
✓			✓	72.38	74.28	91.94	77.38	79.00
✓	✓		✓	72.35	74.09	91.80	77.63	78.97
✓	✓	✓	✓	70.83	73.98	91.31	77.55	78.42

Weak vs. Strong augmentation in MVP. When we apply MVP, augmentation also plays a significant role in ensemble performance. To investigate how different augmentation transformations affect performance, we select several weak and strong augmentations, including random resized crop with a scale factor of $[0.8, 1]$, random horizontal flip, color jittering with a magnitude of 0.4, and RandAugment with $N = 4$ and $M = 5$. The model is trained with MVRML. The number of augmented images is set to 32. In Tab. 1, simple weak augmentations (i.e., random resized crop and flip) can achieve the best performance. By contrast, the strong augmentations (i.e., the color jittering and RandAugment (RA)) have an adverse effect because the images augmented with strong augmentation drift off the data manifold [4], making it harder to predict.

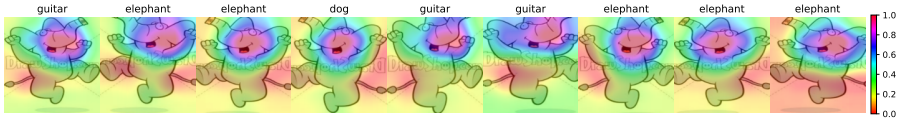


Fig. 4: Class activation map of weak augmented images. The texts above the images are model predictions (best viewed in color).

Visualization of applying multi-view prediction. We visualize the class activation maps of weak augmented images. As shown in Fig. 4, with small perturbations applied to the images, the model tends to make different predictions. However, by ensembling their predictions, the model could eliminate the situation that it makes a wrong prediction from a rare view of the testing image.

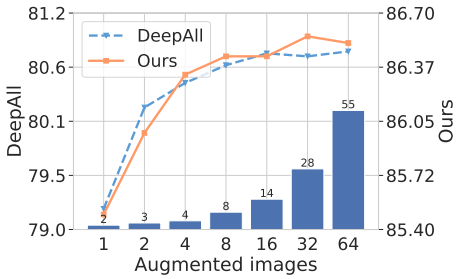


Fig. 5: Accuracy(%) (lines) and testing time(s) (histogram) with augmented images.

The optimal number of trajectories in Reptile. We compare Reptile by varying its number of trajectories. Specifically, the results of using 1, 2, 3, and 4 trajectories are 82.34%, 82.84%, 83.06%, and 82.83%, respectively. We notice that using 3 trajectories is optimal for Reptile. Since our method achieves 85.89% *without* multi-view prediction, its performance still can surpass Reptile.

Table 2: Comparison to SOTA with MVP on PACS dataset.

	MLDG†	FSDCL*	**MVDG**
w/o MVP	82.34	85.85	85.89
w/ MVP	83.41 (+1.07)	86.37 (+0.52)	86.56 (+0.67)

Comparison to SOTA with multi-view prediction. We compare our method to other two SOTA methods (i.e., MLDG and FDSCL) equipped with MVP. We directly utilize MVP to their available trained models (denoted as *) or our reproduced models (denoted as †). As shown in Tab. 2, our method still outperforms these methods. Also, it validates the efficacy of MVP again.

Training and testing time of the method. The experiments are all conducted on 2080Ti GPU. As shown in Tab. 3, the training time of our method increases with more tasks and trajectories. Although our method requires 32 minutes to train a model, it is still comparable with SOTA (e.g., FACT: 2.81h, for the same epochs). Also, we plot the testing time of MVP in Fig. 5 on the Photo domain. The testing time increases with more augmented images. However, we find that 8 augmented images are enough to produce satisfying performance, which does not bring too much computational overhead. Besides, with 8 augmented images, the model can process 208 images per second on a 2080Ti card, which is sufficient for real-time tasks.

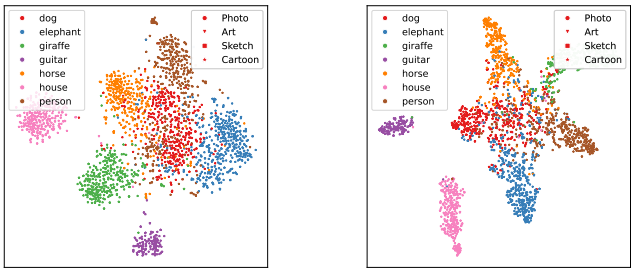
Table 3: The training time w.r.t. the number of tasks and trajectories on Photo domain.

Tasks–Traj.	1-1	1-2	1-3	2-1	2-2	2-3	3-1	3-2	3-3
Time (minute)	11	14	16	16	22	27	21	26	32

Table 4: Experiments on DomaniBed.

	CMNIST	RMNIST	VLCS	PACS	OfficeHome	TerraInc	DomainNet	Avg
ERM	51.5±0.1	98.0±0.0	77.5±0.4	85.5±0.2	66.5±0.3	46.1±1.8	40.9±0.1	66.6
ERM*	51.5±0.1	95.0±0.1	77.1±0.2	85.3±0.1	70.3±0.1	47.7±0.5	38.1±0.1	66.4
MVRML	52.1±0.1	97.6±0.0	77.9±0.2	88.3±0.3	71.3±0.1	51.0±0.6	45.7±0.1	69.1

Experiments on DomainBed. We conduct an experiment on DomianBed benchmark, including CMNIST, RMNIST, VLCS, PACS, OfficeHome, TerraInc and DomainNet. The experiments are repeated three times with a learning rate of $1e-3$. Since our method consists of re-estimate Batch Normalization layer, the Batch Normalization layers are unfreezd. As shown in Tab. 4, ERM is the reported performance of DomainBed and ERM* is our reproduced performance with the same learning rate and freezed BN. Our method can achieve better performance than ERM (a strong baseline in DomainBed) on these datasets.



(a) DeepAll (b) Ours

Fig. 6: The visualization of the feature learned on PACS dataset. DeepAll and the model trained with our method are shown in the figure. The target domain is Cartoon, and the others are all source domains. Different colors indicate different classes, and different shapes indicate different domains (best viewed in color).

Visualization of learned representation. To qualitatively visualize the representation learned by our method, we generate the t-SNE map of both

DeepAll and our model. The target domain is Cartoon, and we only utilize the validation set in the source domain and all images in the test domain to obtain the visualization result. The better the model can generalize, the more clustered the data should be. As shown in Fig. 6, DeepAll cannot cluster the unseen samples well since the plain training cannot prevent overfitting. By contrast, MVRML can yield better clustering results, demonstrating its generalizability.

B Proof of Theorem 1

B.1 Notations

We denote the source domain as $\mathcal{S} = \{\mathcal{D}_1, ...\mathcal{D}_N\}$ and the target domain as $\mathcal{T} = \mathcal{D}_{N+1}$. We denote a task as $t = (\mathcal{B}_{tr}, \mathcal{B}_{te})$, which is obtained by sampling from $\mathcal{S}$. At each iteration, a sequence of sampled tasks along a single trajectory is defined as $\mathbf{T} = \{t_0, \ldots, t_m\}$ with a size of m. Each task in $\mathbf{T}$ is sampled from a mixture distribution of source domains $\mathcal{U} = \sum_{\mathcal{D}_i \sim \mathcal{S}} \alpha_i \mathcal{D}_i$, where $\sum_i \alpha_i = 1$. Each distribution $\mathcal{U}_j$ is sampled from a meta distribution $\mathcal{V}$ with uniformly distributed mixture cooefficients $\sum_i \alpha_i^j = 1$. Thus, $\mathcal{V}$ and $\mathcal{S}$ are actually equivalent with respect to the data point (x, y) since each data point appears with the same probability in $\mathcal{V}$ and $\mathcal{S}$. The meta sequence for the meta-learning methods is defined as $\mathbb{T} = \{\mathbf{T}_0, \mathbf{T}_1, \ldots, \mathbf{T}_n\}$ with a size of n. A training algorithm $\mathbb{A}$ trained with $\mathbb{T}$ or $\mathbf{T}$ is denoted as $\theta = \mathbb{A}(\mathbb{T})$ or $\theta = \mathbb{A}(\mathbf{T})$. We define the expected risk as $\mathcal{E}_{\mathcal{P}}(\theta) = \mathbb{E}_{(x_j, y_j) \sim \mathcal{P}} \ell(f(x_j|\theta), y_j)$. With a little abuse of notation, we define the loss with respect to $\mathbf{T}$ as $\mathcal{L}(\mathbf{T}; \theta) = \frac{1}{m} \sum_{(\mathcal{B}_{tr}, \mathcal{B}_{te}) \in \mathbf{T}} \frac{1}{2} (\mathcal{L}(\mathcal{B}_{tr}; \theta) + \mathcal{L}(\mathcal{B}_{te}; \theta))$ and the loss with respect to $\mathbb{T}$ as $\mathcal{L}(\mathbb{T}; \theta) = \frac{1}{n} \sum_{\mathbf{T} \in \mathbb{T}} \mathcal{L}(\mathbf{T}; \theta)$.

B.2 Lemma

Lemma 1. Given two distributions $\mathcal{P}$ and $\mathcal{Q}$, the following inequality holds [5]:

$$\mathcal{E}_{\mathcal{P}}(\theta) \leq \mathcal{E}_{\mathcal{Q}}(\theta) + \frac{1}{2} \mathbf{Div}(\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{Q}}).$$

where **Div** is the divergence between two distributions.

Since our training scheme is based on the meta-learning algorithm, we introduce the generalization bound of meta-learning with respect to the sample size in [1]. By reformulating our training tasks mentioned above, we introduce the following lemma:

Lemma 2. Assume that an algorithm $\mathbb{A}$ satisfies the following two conditions:

C1. For every pair of meta sequences $\mathbb{T} = \{\mathbf{T}_0, ..., \mathbf{T}_n\}$, $\mathbb{T}^{\setminus i} := \mathbb{T} \setminus \{\mathbf{T}_i\}$, and for every task sequence $\mathbf{T}$, we have $\left\| \mathcal{L}(\mathbf{T}; \mathbb{A}(\mathbb{T})) - \mathcal{L}(\mathbf{T}; \mathbb{A}(\mathbb{T}^{\setminus i})) \right\| \leq \beta_1$.

C2. For every pair of task sequences $\mathbf{T} = \{t^0, \ldots, t^m\}$, $\mathbf{T}^{\setminus j} := \mathbf{T} \setminus \{t_j\}$, and for any task t, we have $\left\| \mathcal{L}(t; \mathbb{A}(t)) - \mathcal{L}(t; \mathbb{A}(t^{\setminus j})), y) \right\| \leq \beta_2$.

Then for any meta distribution $\mathcal{P}$, the following inequality holds with probability at least $1-\delta$ [1]:

$$\mathcal{E}_{\mathcal{P}}(\theta) \leq \hat{\mathcal{E}}_{\mathcal{P}}(\theta) + 2\beta_1 + (4n\beta_1 + M)\sqrt{\frac{\ln\frac{1}{\delta}}{2n}} + 2\beta_2,$$

where M is a bound of loss function ℓ. $\hat{\mathcal{E}}_{\mathcal{P}}(\theta)$ is the empirical error on distribution $\mathcal{P}$. When $\beta_1 = o(1/n^a), a \geq 1/2$ and $\beta_2 = o(1/m^b), b \geq 0$, this bound becomes non-trivial.

C1 and *C2* are β-uniform stability conditions [2] that indicate the sensitivity of the algorithm to the removal of an arbitrary point from the training sample, and if the uniform stability condition holds, we can upper bound the expected error by the empirical error.

B.3 Proof

Theorem 1. Assume that algorithm $\mathbb{A}$ satisfies β_1-uniform stability [2] with respect to $\mathcal{L}(\mathbb{T}; \mathbf{A}(\mathbb{T}))$ and β_2-uniform stability with respect to $\mathcal{L}(\mathbf{T}; \mathbf{A}(\mathbf{T}))$. The following domain generalization error bound holds with probability at least $1-\delta$:

$$\mathcal{E}_{\mathcal{T}}(\theta) \leq \hat{\mathcal{E}}_{\mathcal{S}}(\theta) + \frac{1}{2}\sup_{\mathcal{D}_i \in \mathcal{S}} \mathbf{Div}(\mathcal{D}_i, \mathcal{T}) + (2\beta_1 + (4n\beta_1 + M)\sqrt{\frac{\ln\frac{1}{\delta}}{2n}} + 2\beta_2).$$

Proof. Consider a mixture distribution of N source domains where the mixture weight is geven by γ and $\sum_{i=1}^{N}\gamma_i = 1$.

$$\mathcal{E}_{\mathcal{T}}(\theta) \leq \mathcal{E}_S(\theta) + \frac{1}{2}\mathbf{Div}(\mathcal{S}, \mathcal{T}) \leq \mathcal{E}_S(\theta) + \frac{1}{2}\sum_{i}^{N}\gamma_i \mathbf{Div}(\mathcal{D}_i, \mathcal{T}) \tag{1}$$

$$\leq \mathcal{E}_S(\theta) + \frac{1}{2}\sup_{\mathcal{D}_i \in \mathcal{S}} \mathbf{Div}(\mathcal{D}_i, \mathcal{T}) \tag{2}$$

$$\leq \hat{\mathcal{E}}_{\mathcal{S}}(\theta) + \frac{1}{2}\sup_{\mathcal{D}_i \in \mathcal{S}} \mathbf{Div}(\mathcal{D}_i, \mathcal{T}) + 2\beta_1 + (4n\beta_1 + M)\sqrt{\frac{\ln\frac{1}{\delta}}{2n}} + 2\beta_2 \tag{3}$$

We first bound the expected error between the source and target domains from Eq. (1) to Eq. (2) with **Lemma 1**. Eq. (1) is obtained according to [5] that $\mathbf{Div}(\mathcal{S}, \mathcal{T}) \leq \sum_{i}^{N}\gamma_i \mathbf{Div}(\mathcal{D}_i, \mathcal{T})$, where $\sum_{i=1}^{N}\gamma_i = 1$. Then we bound Eq. (1) with the maximum divergence $\sup_{\mathcal{D}_i \in \mathcal{S}} \mathbf{Div}(\mathcal{D}_i, \mathcal{T})$ between $\mathcal{D}_i$ and $\mathcal{T}$.

According to C.1 and C.2 and the fact that $\mathcal{E}_{\mathcal{V}}(\theta) = \mathcal{E}_{\mathcal{S}}(\theta)$ since the distribution of $\mathcal{V}$ and $\mathcal{S}$ are equivalent with respect to the data points, we have:

$$\mathcal{E}_{\mathcal{V}}(\theta) \leq \hat{\mathcal{E}}_{\mathcal{V}}(\theta) + 2\beta_1 + (4n\beta_1 + M)\sqrt{\frac{\ln\frac{1}{\delta}}{2n}} + 2\beta_2. \tag{4}$$

Thus, by replacing the expected error of source domains in Eq. (3) with empirical error in Eq. (4), we arrive at Theorem 1.

C Other Details

Visualization of loss surface. To visualize the loss surface of a model, we follow the visualization technique in [3]. Suppose we have weight vectors of three models w_1, w_2, w_3. We first find two basis $\hat{u}, \hat{v}$ of the plane across these weights: $u = (w_2 - w_1)$, $v = ((\theta_3 - \theta_1) - \langle \theta_3 - \theta_1, \theta_2 - \theta_1 \rangle) / \|\theta_2 - \theta_1\|^2 \cdot (\theta_2 - \theta_1)$, $\hat{u} = u/\|u\|, \hat{v} = v/\|v\|$. Then, we define a Cartesian grid in the basis $\hat{u}, \hat{v}$ and calculate the training/test loss corresponding to each of the points in the grid.

References

1. Al-Shedivat, M., Li, L., Xing, E., Talwalkar, A.: On data efficiency of meta-learning. In: AISTATS (2021) 6, 7
2. Bousquet, O., Elisseeff, A.: Stability and generalization. JMLR (2002) 7
3. Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D.P., Wilson, A.G.: Loss surfaces, mode connectivity, and fast ensembling of dnns. In: NeurIPS (2018) 8
4. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. arXiv (2019) 3
5. Zhao, H., Zhang, S., Wu, G., Moura, J.M., Costeira, J.P., Gordon, G.J.: Adversarial multiple source domain adaptation. In: NeurIPS (2018) 6, 7