Supplementary Materials for Variational Model Perturbation for Source-Free Domain Adaptation

Mengmeng Jing[1,2]*, **Xiantong Zhen**[2] †, **Jingjing Li**[1], **Cees G. M. Snoek**[2]
[1]University of Electronic Science and Technology of China
[2]University of Amsterdam

1 Analysis of Sample Numbers

In Fig. 1, we analyse the impact of different sample numbers on performance during the training phase. From Fig. 1, we observe that the performance of the model does not change much as the sample number increases. Thus, our variational model perturbation is robust to different sampling numbers. In practical applications, we only sample once in the training phase for running efficiency.

2 Detailed Expression of the Objective Function

The objective function of our method consists of two terms, the KL divergence term and the expected likelihood term:

$$\mathcal{L} = \mathcal{L}_{KL} + \mathcal{L}_{exp} = \mathrm{KL}[q(\Delta w|0,\sigma)||p(\Delta w)] - \mathbb{E}_{q(\Delta w|0,\sigma)}[p(D_t|\Delta w)].$$

For the KL divergence term, it is defined as:

$$\mathcal{L}_{KL} = \mathrm{KL}[q(\Delta w|0,\sigma)||p(\Delta w)] = \int q(\Delta w|0,\sigma)\log\frac{q(\Delta w|0,\sigma)}{p(\Delta w)}d\Delta w,$$

where $q(\Delta w|0,\sigma)$ and $p(\Delta w)$ are the variational posterior and prior distributions, respectively. Assume $q(\Delta w|0,\sigma) \sim \mathcal{N}(\mu_0, \sigma_0^2 \mathrm{I})$, $p(\Delta w) \sim \mathcal{N}(\mu_1, \sigma_1^2 \mathrm{I})$, then the KL divergence term can be calculated as:

$$\mathrm{KL}[q(\Delta w|0,\sigma)||p(\Delta w)] = \frac{1}{2}[\mathrm{tr}(\sigma_1^{-1}\sigma_0) - k + (\mu_1 - \mu_0)^{\mathrm{T}}\sigma_1^{-1}(\mu_1 - \mu_0) + \log(\frac{\det \sigma_1}{\det \sigma_0})],$$

where k is the dimension of the random variable, tr and det represent trace and determinant operation, respectively.

For the second term, the specific form of it depends on the model being perturbed. For SHOT [1], maximizing this term is implemented as minimizing the information loss and self-supervised loss:

$$\mathcal{L}_{exp} = \mathcal{L}_{im} + \beta\mathcal{L}_{sup},$$

where $\mathcal{L}_{im}$ is the information maximization loss which can be calculated as:

$$\mathcal{L}_{im} = \frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{H}(p_t^i) - \mathcal{H}(\frac{1}{n_t}\sum_{i=1}^{n_t}(p_t^i)),$$

where $\mathcal{H}$ is the entropy, $p_t^i = \mathrm{softmax}(G_t(x_t^i))$ is the classification probability for sample x_t^i.

---

[*] This work was done when Mengmeng Jing was a visiting student at University of Amsterdam.

[†] Currently with United Imaging Healthcare, Co., Ltd., China.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

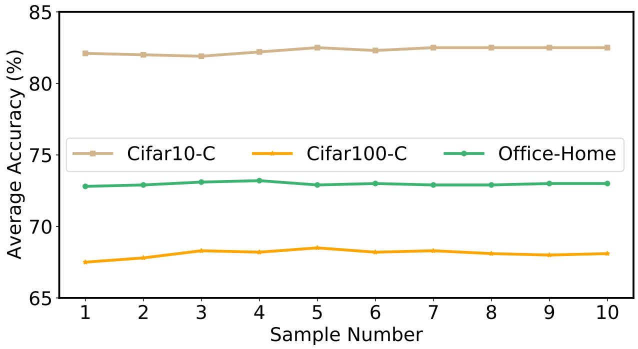

Figure 1: Impact of different sample numbers for the performance.

$\mathcal{L}_{sup}$ is the self-supervision loss which can be calculated as:

$$\mathcal{L}_{sup} = \frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{L}_C(G_t(x_t^i), \hat{y}_t^i)$$

where $\mathcal{L}_C$ is the cross-entropy loss, $\hat{y}_t^i$ is the pseudo-labels of the target image x_t^i.

For Tent [2], maximizing this term is implemented as minimizing the entropy loss:

$$\mathcal{L}_{exp} = \frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{H}(p_t^i),$$

3 Local Reparameterization Trick

The local reparameterization trick [3] allows us to sample pre-activations instead of weights. Specifically, for the perturbed weight w_t sampled from a factorized Gaussian $\mathcal{N}(w_s, \sigma^2 I)$, the pre-activations $a = w_t h$ are distributed according to $\mathcal{N}(w_s h, \sigma^2 h^2 I)$, where σ is the learnable perturbation parameters, w_s is the weight before perturbations, h is the output from the preceding layer.

4 Discussion

Application to other architectures. The perturbation operation is architecture-agnostic, and could be applied to various architectures though we implement convolutional and fully-connected layers in this work. Therefore, our method is not limited to computer vision settings. The exploration of applying variational model perturbations to other architectures will be a promising direction.

Application to source-needed DA. Variational model perturbation is specifically designed for SFDA. That is, we have a source-trained model and perturb the model parameter using a handful of target data. In source-needed DA, we could directly use source and target data to retrain the model if source data are already available. Therefore, it is not necessary to apply variational model perturbation to source-needed DA.

Why variational model perturbation is effective? Variational model perturbation is effective mainly because of the probabilistic modeling. In this way, we perturb a deterministic model into a Bayesian model. As a result, the perturbed model can be generalized to more domains while maintaining the performance on the source domain, i.e., we expand the effective coverage of the source model. Moreover, by adopting the parameter sharing strategy, our method is effective even though learnable parameters are very few.

Limitations. Our method is closely related to learning a Bayesian neural network. While enjoying the better generalization ability, our method could also suffer from the limitation of high training cost [4]. This would make it computationally expensive when the model size is very large.

```
Algorithm1: Variational Model Perturbation
Input: Unlabeled target dataset $\mathcal{D}_t=\{x_j^t|x_j^t \in \mathcal{X}_t\}_{j=1}^{n_t}$, pre-trained source model $G_s$ parameterized by $w_s$, the max iteration number T.
Initialization:
    1: Initialize the target model $G_t$ with $G_s$.
    2: Calculate the variance Var($w_s^{li}$) in Eq. 9 for each convolution kernel of the source model layer by layer to determine the prior distribution of the convolution kernel.
    3: Initialize the perturbation parameters $\rho$ and add $\rho$ to the optimizer.
    4: Modify the forward function of every layer so that we can compute the perturbed weights and the corresponding output layer by layer according to Eq. 8.
While t $\leq$ T
    1: Sample a batch of target sample $x_t$ and feed them to the network $G_t$.
    2: Compute the perturbed weights $w_t$ and the corresponding output $z_t$ layer by layer according to Eq. 8.
    3: Compute the loss in Eq. 3 using the final output of $G_t$.
    4: Update the perturbation parameter $\rho$ through backpropagation.
end while
Output: the perturbed model $G_t$ parameterized by $w_t$.
Prediction: Classify $x_t$ by $\tilde{y}_t = G_t(x_t)$
```

Table 1: Detailed accuracy (%) on Office-Home under Generalized SFDA, where all the methods use ResNet-50 backbone. Accuracy on source and target domain is denoted by S and T, respectively. Model perturbation achieves the best performance across all the compared methods.

	Ar→Cl	**Ar→Pr**	**Ar→Rw**	**Cl→Ar**	**Cl→Pr**	**Cl→Rw**	**Pr→Ar**	**Pr→Cl**	**Pr→Rw**	**Rw→Ar**	**Rw→Cl**	**Rw→Pr**	**Avg.**	**H**
	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	
Source-only	77.5/44.6	77.5/67.1	77.5/73.5	79.3/48.9	79.3/59.5	79.3/61.9	92.4/52.2	92.4/40.8	92.4/73.5	85.2/64.8	85.2/45.3	85.2/76.9	83.6/59.1	69.2
SHOT [1]	60.7/55.0	64.9/76.7	71.5/80.0	65.5/67.6	63.6/76.6	67.0/75.6	79.0/65.1	73.7/53.7	84.8/80.2	79.4/71.3	72.0/59.1	78.3/84.0	71.7/70.4	71.0
Ours	63.9/55.7	68.2/78.0	76.0/81.5	71.0/69.7	69.4/77.2	71.5/77.4	82.3/66.7	80.7/54.7	90.9/81.5	83.6/71.9	78.6/59.9	82.8/84.6	76.6/71.6	74.0

5 Pseudo Code

For a clear understanding, we summarize the detailed description of model training in **Algorithm 1**.

6 Elaboration of the Experimental Details

For a clearer elaboration of the experimental details, we clarify SFDA consists of three phases, namely, source domain pre-training, target domain adaptation and testing. Our paper considers three different settings, each with a different experimental procedure:

Offline SFDA: in the source domain pre-training phase, we use all labeled source domain data to train the source model by minimizing cross entropy loss. In the target domain adaptation phase, we use all unlabeled target data to adapt the model to the target domain. In the testing phase, we make predictions for all target data.

Generalized SFDA: we first split the source domain data into 80% and 20% parts. In the source domain pre-training phase, the data of the 80% part are labeled and we use them to pre-train the source model. In the target domain adaptation phase, we use all the unlabeled target data to adapt the model to the target domain. In the testing phase, we predict the remaining 20% source data and all target data.

Continual Online SFDA: the source domain consists of the clean images from CIFAR10, CIFAR100 and ImageNet, while the target domain consists of the corrupted images from CIFAR10-C, CIFAR100-C and ImageNet-C. In the source domain pre-training phase, we directly adopt the publicly available model pre-trained on the clean training set as the source model for SFDA. Specifically, CIFAR10-C uses WideResNet-28 from RobustBench [7]. CIFAR100-C uses ResNeXt-29 from [8]. ImageNet-C uses ResNet-50 from RobustBench [7]. Different from the previous settings, the target domain adaptation phase and the testing phase of Continual Online SFDA are carried out

Table 2: Detailed test errors (%) on CIFAR10-C, where all the results use the WideResNet-28 backbone. Model perturbation achieves the best performance across all the compared methods.

	Gaussian	**shot**	**impulse**	**defocus**	**glass**	**motion**	**zoom**	**snow**	**frost**	**fog**	**bright.**	**contra.**	**elast.**	**pixel.**	**jpeg**	**Avg.**
Source-only	72.3	65.7	72.9	46.9	54.3	34.8	42.0	25.1	41.3	26.0	9.3	46.7	26.6	58.5	30.3	43.5
AdaBN [5]	28.1	26.1	36.3	12.8	35.3	14.2	12.1	17.3	17.4	15.3	8.4	12.6	23.8	19.7	27.3	20.4
Tent [2]	**24.8**	**20.6**	**28.6**	14.4	31.1	16.5	14.1	19.1	18.6	18.6	12.2	20.3	25.7	20.8	24.9	20.7
Tent-FT [2]	27.8	25.5	35.4	12.6	34.3	13.8	11.8	17.0	16.9	14.7	**8.1**	12.7	22.5	18.6	25.6	19.8
BACS (MAP) [6]	26.5	24.8	31.5	12.9	33.5	14.2	12.1	17.4	17.5	15.1	8.6	13.3	22.9	19.5	25.0	19.6
Ours	25.9±0.1	21.4±0.1	29.8±0.2	**12.2±0.1**	**30.3±0.1**	**13.4±0.1**	**11.5±0.1**	**15.9±0.1**	**15.4±0.1**	**14.4±0.1**	8.6±0.1	**12.2±0.2**	**21.1±0.2**	**15.5±0.3**	**21.2±0.3**	**17.9**

Table 3: Detailed test errors (%) on CIFAR100-C, where all the results use the ResNeXt-29 backbone. Model perturbation achieves the best performance across all the compared methods.

	Gaussian	**shot**	**impulse**	**defocus**	**glass**	**motion**	**zoom**	**snow**	**frost**	**fog**	**bright.**	**contra.**	**elast.**	**pixel.**	**jpeg**	**Avg.**
Source-only	73.0	68.0	39.4	29.3	54.1	30.8	28.8	39.5	45.8	50.3	29.5	55.1	37.2	74.7	41.2	46.4
AdaBN [5]	42.1	40.7	42.7	27.6	41.9	29.7	27.9	34.9	35.0	41.5	26.5	30.3	35.7	32.9	41.2	35.4
BACS (MAP) [6]	38.1	33.7	37.2	28.5	39.1	31.5	29.3	35.8	35.7	40.2	31.6	38.7	40.6	37.7	47.3	36.3
Tent [2]	**37.2**	**35.8**	41.7	37.9	51.2	48.3	48.5	58.4	63.7	71.1	70.4	82.3	88.0	88.5	90.4	60.9
Tent-FT [2]	42.1	40.6	42.7	27.6	41.8	29.7	27.8	34.8	34.9	41.5	26.5	30.2	35.5	32.8	40.9	35.3
Ours	40.8±0.4	36.7±0.3	**38.0±0.3**	**26.0±0.2**	**37.7±0.3**	**27.9±0.1**	**25.7±0.2**	**32.4±0.3**	**31.2±0.3**	**36.8±0.2**	**25.1±0.4**	**28.9±0.2**	**32.8±0.2**	**29.2±0.1**	**38.1±0.3**	**32.5**

simultaneously. Specifically, the corrupted images are fed into the network in an online fashion. At each iteration, we first predict the corrupted images and then update the model for one step. We minimize the entropy loss just as Tent [2]. Since there are 15 different corruption types, we adapt the source pre-trained model to each corruption type sequentially.

7 Detailed Experimental Results

In Table 1-Table 4, we report the detailed accuracies of variational model perturbation. From the results, variational model perturbation outperforms the compared methods in all the SFDA tasks.

8 Other Implementations

Apart from SHOT and Tent, we have also applied our variational model perturbation to other methods. In Table 5-Table 8, we report results of variational model perturbation based on NRC [9], GSFDA [10] and CoTTA [11], respectively.

It is worth noting that in Table 8, CoTTA [11] adopts stochastic restoration for the weights to avoid catastrophic forgetting. To verify the effectiveness of variational model perturbation in avoiding knowledge forgetting, we implement model perturbation based on CoTTA without stochastic restoration. In Table 8, we observe that variational model perturbation can achieve better performance compared with stochastic restoration.

9 Learnable Parameters

In Table 9, we report the number of learnable parameters for the case where the model perturbation uses a shared strategy and a non-shared strategy. We observe that model perturbation can achieve excellent performance with only a few learnable parameters.

Table 4: Detailed test errors (%) on ImageNet-C, where all the results use the ResNet-50 backbone. Model perturbation achieves the best average accuracies across all the compared methods.

	Gaussian	**shot**	**impulse**	**defocus**	**glass**	**motion**	**zoom**	**snow**	**frost**	**fog**	**bright.**	**contra.**	**elast.**	**pixel.**	**jpeg**	**Avg.**
Source-only	95.3	94.6	95.3	84.9	91.1	86.9	77.2	84.4	79.7	77.3	44.4	95.6	85.2	76.9	66.7	82.4
AdaBN [5]	87.8	87.4	87.6	88.6	88.0	78.5	64.6	68.0	70.3	55.1	**37.2**	89.6	58.5	57.0	67.8	72.4
Tent [2]	**83.1**	**75.9**	**75.0**	**81.5**	**80.2**	76.0	66.4	72.8	75.0	67.9	53.1	89.0	72.8	70.4	74.3	74.2
Tent-FT [2]	86.7	83.1	82.0	85.3	83.6	74.0	61.2	64.8	66.6	54.7	39.8	84.8	58.6	57.0	63.6	69.7
BACS (MAP) [6]	84.3	77.7	76.1	81.4	79.6	73.4	62.7	68.9	70.6	61.4	45.9	83.0	61.6	58.0	64.6	69.9
Ours	86.6±0.2	83.0±0.1	81.7±0.2	83.8±0.3	81.5±0.1	**71.6±0.2**	**59.1±0.2**	**63.9±0.1**	**65.1±0.1**	**54.0±0.2**	38.4±0.1	**79.9±0.2**	**54.3±0.3**	**52.2±0.1**	**59.8±0.3**	**67.7**

Table 5: Detailed accuracy (%) of other implementations on Office, where all methods use a ResNet-50 backbone. VMP represents variational model perturbation.

	A→W	**A→D**	**W→A**	**W→D**	**D→A**	**D→W**	**Avg.**
NRC [9]	90.8	96.0	75.0	100.0	75.3	99.0	89.4
NRC+VMP	94.3±0.2	95.6±0.3	76.2±0.1	100.0±0.0	75.8±0.2	99.0±0.3	90.2

Table 6: Detailed accuracy (%) of other implementations on Office-Home, where all methods use a ResNet-50 backbone. VMP represents variational model perturbation.

	Ar→ Cl	**Ar→ Pr**	**Ar→ Rw**	**Cl→ Ar**	**Cl→ Pr**	**Cl→ Rw**	**Pr→ Ar**	**Pr→ Cl**	**Pr→ Rw**	**Rw→ Ar**	**Rw→ Cl**	**Rw→ Pr**	**Avg.**
NRC [9]	57.7	80.3	82.0	68.1	79.8	78.6	65.3	56.4	83.0	71.0	58.6	85.6	72.2
NRC+VMP	57.9±0.1	78.3±0.2	83.2±0.2	69.7±0.3	79.9±0.1	81.7±0.1	68.4±0.2	54.5±0.2	83.6±0.1	75.3±0.1	60.1±0.2	84.7±0.3	73.1

Table 7: Detailed accuracy (%) of other implementation on Office-Home under Generalized SFDA, where all the methods use ResNet-50 backbone. Accuracy on source and target domain is denoted by S and T, respectively. VMP represents variational model perturbation.

	Ar→Cl	**Ar→Pr**	**Ar→Rw**	**Cl→Ar**	**Cl→Pr**	**Cl→Rw**	**Pr→Ar**	**Pr→Cl**	**Pr→Rw**	**Rw→Ar**	**Rw→Cl**	**Rw→Pr**	**Avg.**	**H**
	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	S/T	
Source-only	77.5/44.6	77.5/67.1	77.5/73.5	79.3/48.9	79.3/59.5	79.3/61.9	92.4/52.2	92.4/40.8	92.4/73.5	85.2/64.8	85.2/45.3	85.2/76.9	83.6/59.1	69.2
GSFDA [10]	68.2/53.8	71.1/75.0	74.1/77.6	76.4/66.1	78.6/73.4	78.0/77.3	87.2/64.0	85.1/52.5	89.9/80.9	82.6/70.9	77.8/59.3	82.6/82.3	79.3/69.4	74.0
GSFDA + VMP	69.5/54.7	73.0/77.4	73.8/79.7	77.6/67.0	79.7/76.4	79.9/78.8	89.3/65.5	88.0/54.0	90.9/82.1	84.7/71.3	80.7/60.5	83.5/83.8	80.9/70.9	75.6

Table 8: Detailed test errors (%) of other implementation in the continual online setting, where the backbone network for CIFAR10-C, CIFAR100-C and ImageNet-C are WideResNet-28, ResNeXt-29 and ResNet-50, respectively. CoTTA* means CoTTA method without stochastic restore. RST and VMP represent weight restore and variational model perturbation, respectively.

dataset	Variant	**Gaussian**	**shot**	**impulse**	**defocus**	**glass**	**motion**	**zoom**	**snow**	**frost**	**fog**	**bright.**	**contra.**	**elast.**	**pixel.**	**jpeg**	**Avg.**
CIFAR10-C	CoTTA* [11]	24.0	22.2	27.2	12.7	28.9	13.2	11.8	17.0	15.8	13.6	8.5	12.0	20.4	15.3	19.6	17.5
	CoTTA*+RST [11]	24.0	21.4	26.6	12.6	29.1	13.0	12.5	17.1	15.4	13.2	8.1	11.3	20.5	15.5	19.5	17.3
	CoTTA*+VMP	23.9	21.7	25.7	11.9	28.4	13.2	10.9	15.8	15.3	12.9	8.1	11.0	19.2	14.7	18.4	16.7
CIFAR100-C	CoTTA* [11]	40.4	37.8	39.8	27.7	37.9	28.7	27.5	34.2	32.6	41.0	26.7	28.2	33.0	29.2	33.5	33.2
	CoTTA*+RST [11]	40.4	37.9	39.9	27.3	37.7	28.4	26.5	33.1	32.2	39.9	25.3	26.8	32.5	28.4	33.2	32.6
	CoTTA*+VMP	38.4	36.2	37.2	25.3	36.8	26.7	25.2	32.3	31.2	37.9	24.2	27.2	31.5	28.0	33.7	31.4
ImageNet-C	CoTTA* [11]	86.9	82.3	78.1	81.8	78.5	73.1	66.8	69.5	69.1	65.5	58.6	72.7	63.0	60.9	62.6	71.3
	CoTTA*+RST [11]	86.9	82.3	77.8	81.1	77.2	69.8	62.8	65.3	64.6	59.1	50.4	68.3	58.3	55.6	56.7	67.7
	CoTTA*+VMP	86.6	83.1	79.8	83.2	79.9	70.3	59.4	62.6	62.7	52.8	42.2	69.2	54.7	50.9	55.5	66.2

Table 9: Comparison of learnable parameters (million) between using non-sharing strategy and parameter sharing strategy.

Architecture Task	ResNet-50 Office/Office-Home	WideResNet-28 CIFAR10-C	ResNeXt-29 CIFAR100-C	ResNet-50 ImageNet-C
non-sharing	24.03	36.50	7.02	27.63
sharing	0.06	0.03	0.14	2.13
ratio	0.25%	0.08%	1.99%	7.71%

References

[1] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020.

[2] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2020.

[3] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

[5] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. In *International Conference on Learning Representations Workshop*, 2017.

[6] Aurick Zhou and Sergey Levine. Bayesian adaptation for covariate shift. In *Advances in Neural Information Processing Systems*, 2021.

[7] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS Datasets and Benchmarks Track, 2021*, 2021.

[8] Dan Hendrycks*, Norman Mu*, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple method to improve robustness and uncertainty under data shift. In *International Conference on Learning Representations*, 2020.

[9] Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[10] Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Generalized source-free domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8978–8987, 2021.

[11] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2022.