Variational Model Perturbation for Source-Free Domain Adaptation

Anonymous Author(s)
Affiliation
Address
email

Abstract

We aim for source-free domain adaptation, where the task is to deploy a model pre-trained on source domains to target domains. The challenges stem from the distribution shift from the source to the target domain, coupled with the unavailability of any source data or labeled target data for optimization. Rather than fine-tuning the model by updating the parameters, we propose to perturb the source model to achieve adaptation to target domains. We introduce perturbations into the model parameters by variational Bayesian inference in a probabilistic framework. By doing so, we can effectively adapt the model to the target domain while largely preserving the discriminative ability. Importantly, we demonstrate the theoretical connection to learning Bayesian neural networks, which proves the generalizability of the perturbed model to target domains. To enable more efficient optimization, we further employ a parameter sharing strategy, which substantially reduces the learnable parameters compared to a fully Bayesian neural network. Our model perturbation provides a new probabilistic way for domain adaptation which enables efficient adaptation to target domains while maximally preserving knowledge in source models. Experiments on several source-free benchmarks under three different evaluation settings verify the effectiveness of the proposed variational model perturbation for source-free domain adaptation.

1 Introduction

A well-trained neural network can achieve satisfactory performance on in-distribution tasks. However, in real applications, a common situation is that the training set and the test set are from different distributions. In this case, the performance of the models usually degrades significantly due to domain shifts. Domain adaptation [1, 2] is proposed to solve this problem by transferring knowledge from the labeled source domain (the training set) to the unlabeled target domain (the test set). It is assumed that source data are still available when adapting to target data. In this paper, we focus on the more challenging scenario where source data are not accessible, which is known as source-free domain adaptation (SFDA) [3, 4]. SFDA promises much wider ranges of applications than the vanilla domain adaptation in that it can circumvent the problem of not disclosing source data in order to protect the privacy of individuals or the intellectual property of commercial entities.

To deal with domain shift, two popular techniques in the literature to adapt the SFDA model to the target domain are fine-tuning (updating the whole parameters) and modulating (tuning the batch normalization layers and freeze all the other parameters). The former focuses on the difference of domains, while the latter focuses on the relevance of domains. For fine-tuning methods, e.g., SHOT [5] and NRC [6], the parameters of the entire model are further updated by designing optimization objectives. For example, SHOT [5] calibrates the model predictions by information maximization and pseudo-labeled self-supervision, while NRC [6] exploits the nearest neighbor

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

relationship between samples. Though being empirically successful, both model fine-tuning and batch normalization modulation suffer from drawbacks. Fine-tuning may cause some problems. First, the model will deviate from the source data, leading to performance degradation on the source domain, similar to catastrophic forgetting in continual learning [7, 8, 9, 10]. Second, fine-tuning could result in a model being biased to a specific target domain (overfitting), thus reducing the generalization of the model [11, 12, 13]. Batch normalization modulation approaches [14, 15, 16] keep weights of all convolutional and fully-connected layers frozen, and only modulate the statistics in the batch normalization layers [17] to adapt to target data. However, they tend to be less flexible and would not continuously adapt to target data with large domain shifts. In addition, those are all deterministic approaches which suffer from lack of generalizability [18, 19, 20].

In this paper, we propose variational model perturbations, which fixes the existing weights and slightly perturbs them to encode the uncertainties according to the domain shifts. We learn to perturb model parameters to achieve domain adaptation by variational inference. By variational perturbations, we introduce uncertainties into the model. As a result, we perturb a deterministic network into a Bayesian neural network, allowing the model to generalize to relevant domains. We demonstrate that by perturbing the model in a variational manner, we are equivalent to learning a special Bayesian neural network with a fixed mean and a learnable variance. To achieve more efficient perturbation, we further adopt a parameter sharing strategy, which significantly reduces the learnable parameters. This enables more efficient adaptation in contrast to learning a fully Bayesian neural network. It is worth noting that the perturbation is naturally regularized due to the variational inference formulation, which prevents the perturbed weights from being distorted during the continuous unsupervised learning. Our approach is easy to implement, offers a plug-and-play module to adapt model parameters and works seamlessly with different optimization objectives. We conduct extensive experiments on five datasets with three different evaluation settings. The results consistently demonstrate the effectiveness of variational model perturbation for source-free domain adaptation.

2 Methodology

We start from the vanilla unsupervised domain adaptation, based on which we formally introduce the setting for source-free domain adaptation. For the vanilla unsupervised domain adaptation setting, we are given a source domain $\mathcal{D}_s=\{(x_i^s, y_i^s)|x_i^s \in \mathcal{X}_s, y_i^s \in \mathcal{Y}_s\}_{i=1}^{n_s}$ with n_s labeled data and a target domain $\mathcal{D}_t=\{x_j^t|x_j^t \in \mathcal{X}_t\}_{j=1}^{n_t}$ with n_t unlabelled data under the condition that $p(\mathcal{X}_s) \neq p(\mathcal{X}_t)$ where $p(\mathcal{X}_s)$ and $p(\mathcal{X}_t)$ are the source and target data distributions. The source and target domains share the same label space, i.e., $\mathcal{Y}_s=\mathcal{Y}_t$. The task is to learn a model $f : \mathcal{X}_t \rightarrow \mathcal{Y}_t$ that could predict the target labels. In the source-free domain adaptation setting, we are only given the unlabeled target domain data $\mathcal{D}_t=\{x_j^t\}_{j=1}^{n_t}$ and the model trained on the source domain parameterized by w_s.

Model Perturbation with Uncertainty From the probabilistic perspective, we define a model $p(w_s|D_s)$ to be trained on the source domain D_s. Then, the model parameters w_s can be optimized by maximum likelihood estimation under the i.i.d. assumption:

$$w_s = \arg\max_{w_s} \log p(D_s|w_s) = \arg\max_{w_s} \sum_i \log p(y_i^s|x_i^s, w_s). \tag{1}$$

Usually we could optimize Eq. (1) through mini-batch based stochastic gradient descent [21] and the optimum of w_s is a point estimate of the model parameter. Having a model pre-trained in Eq. (1) on a source domain, we would like to deploy it on different target domains. However, usually the performance of the source model degrades significantly on the target domains due to the domain shift.

Therefore, we propose to perturb the model parameters based on the point estimate w_s in order to make it generalizable to the target domain while maximally maintain the discriminative ability of the source model. Denote the perturbed weights as w_t, which is obtained by a simple perturbation operation as follows:

$$w_t = w_s + \Delta w, \tag{2}$$

where Δw is the perturbation for w_s. It is worth noting that we only perturb the weights of the convolutional and fully-connected layers in the network and we update the running mean and running variance in the batch normalization layers through moving average.

In this paper, we formulate the perturbation as a variational inference problem in a probabilistic framework. Given the target domain D_t, we compute the posterior distribution of the perturbations by

Bayesian inference, i.e., $p(\Delta w|D_t)$. However, directly computing the expectations of the posterior distribution is intractable, since it is equivalent to computing expectations of an uncountably infinite set of neural networks [19, 22]. Therefore, as a more realistic alternative, we resort to computing the posterior distributions through variational inference.

Specifically, assume the perturbations are drawn from a zero-mean distribution, i.e., $\Delta w \sim \mathcal{N}(0, \sigma^2 I)$, where σ is the learnable standard deviation. Then, our task reduces to solve the optimization problem such that the variational posterior $q(\Delta w|0, \sigma)$ can approach the true posterior $p(\Delta w|D_t)$. We minimize the Kullback-Leibler (KL) divergence between them as follows:

$$\begin{aligned}
\sigma^* &= \arg\min_{\sigma} \mathrm{KL}[q(\Delta w|0,\sigma)||p(\Delta w|D_t)] \\
&= \arg\min_{\sigma} \int q(\Delta w|0,\sigma) \log \frac{q(\Delta w|0,\sigma)}{p(\Delta w|D_t)} d\Delta w \\
&= \arg\min_{\sigma} \int q(\Delta w|0,\sigma) \log \frac{q(\Delta w|0,\sigma)}{p(D|\Delta w)p(\Delta w)} d\Delta w \\
&= \arg\min_{\sigma} \mathrm{KL}[q(\Delta w|0,\sigma)||p(\Delta w)] - \mathbb{E}_{q(\Delta w|0,\sigma)}[p(D_t|\Delta w)], \qquad (3)
\end{aligned}$$

where the first term in Eq. (3) is the KL-divergence between the posterior and the prior, and the second term is the expectation likelihood. The prior offers a regularization on the perturbation, which prevents the model to be perturbed too much from the source model.

Connection to Learning a Bayesian Neural Network It has been proven that introducing uncertainty into weights can improve generalizability of neural networks [19, 20]. Actually, under mild conditions, our variational model perturbation is equivalently learning a Bayesian neural network. We demonstrate this as follows.

Assume that we would like to learn a neural network by variational Bayesian inference, where we place a prior $p(w_t)$ over the weight. We design the variational posterior as $q(w_t|w_s, \sigma)$. Here we assume the mean w_s to be a constant which is the point estimate of the model parameters, and the variance σ^2 to be a learnable variable. We minimize the KL divergence between the variational posterior $q(w_t|w_s, \sigma)$ and the true posterior $p(w_t|D_t)$:

$$\sigma^* = \arg\min_{\sigma} \mathrm{KL}[q(w_t|w_s,\sigma)||p(w_t|D_t)] \qquad (4)$$

$$= \arg\min_{\sigma} \int q(w_t|w_s,\sigma) \log \frac{q(w_t|w_s,\sigma)}{p(w_t)p(D_t|w_t)} dw_t. \qquad (5)$$

As we define $w_t = \Delta w + w_s$, the posterior $w_t \sim q(w_t|w_s, \sigma^2)$ is transformed from $\Delta w \sim q(\Delta w|0, \sigma^2)$. By adopting the technique of change of variables to Eq. (5), we arrive at:

$$\sigma^* = \arg\min_{\sigma} \int q(\Delta w|0,\sigma) \log \frac{q(\Delta w|0,\sigma)}{p(\Delta w)p(D_t|\Delta w)} d\Delta w \qquad (6)$$

$$= \arg\min_{\sigma} \mathrm{KL}[q(\Delta w|0,\sigma)||p(\Delta w|D_t)]. \qquad (7)$$

Therefore, minimizing the KL-Divergence between $q(w|w_s, \sigma)$ and $p(w|D_t)$ is equivalent to minimizing the KL-Divergence between $q(\Delta w|0, \sigma)$ and $p(\Delta w|D_t)$. Here we assume that we adopt the same prior, i.e., $p(w_t) = p(\Delta w)$.

Variational Posterior of Perturbations Following [19], we assume that the variational posterior distribution is a diagonal Gaussian distribution. Then we can obtain a sample of the perturbations Δw by sampling the unit Gaussian and scaling it by the standard deviation σ. To ensure that σ is non-negative, in practical implementations σ is calculated as $\sigma = \sqrt{\exp(\rho)}$, where ρ is the *de facto* variational posterior parameter which is the logarithm of the variance. Thus, a sample from the perturbed w_t can be represented as:

$$w = w_s + \epsilon \circ \sqrt{\exp(\rho)}, \qquad (8)$$

where $\epsilon \sim N(0, I)$ and $\circ$ denotes element-wise multiplication.

Prior of Perturbations Previous works have shown that the selection of different priors affects the generalization performance of the model [23, 18, 24]. Considering that weights of each layer have different variations, it may not be optimal to use a uniform prior for the whole network. Therefore, we propose to specify different priors for perturbations of each layer according to the statistical properties within the pre-trained model. Specifically, each convolutional layer contains multiple convolutional kernels. Previous research [23] has found that weights in the same convolutional kernel have strong correlations, while different convolutional kernels do not, even if they are in the same layer. In view of this, we propose an *Adaptive prior*, which computes the standard deviation of each convolutional kernel in the pre-trained model as the prior of the perturbations of the whole convolutional kernel. Then, the prior distribution shared by the whole convolutional kernel is as follows:

$$p(\Delta w^{li}) = \mathcal{N}(0, \lambda \text{Var}(w_s^{li})I), \tag{9}$$

where $p(\Delta w^{li})$ represents the prior distribution for perturbations of the i-th convolution kernel in l-th convolutional layer, and Var is the variance of the convolutional kernel in the pre-trained model w_s. λ is a scale coefficient. We experimentally observed that λ=1.0 can achieve satisfactory performance.

As for the fully connected layer, its weights are a matrix. Every column may have some semantic meaning. For example, if this layer is the classifier layer, then every column corresponds to the prototype features of one of the classes. So we compute the variance for each column of the fully connected layers in the pre-trained model as the prior of that column.

Likelihood Function In general, our variational model perturbation is independent of the likelihood function. In source-free domain adaptation, since data from the target domain are unlabeled, the likelihood function can be implemented by optimization with respect to unsupervised losses. For example, we can impose Shannon entropy minimization to encourage the model predictions to be close to one-hot encodings, which is also used in Tent [14]. In addition, Liang et al. [5] highlight that ideal predictions should have enough category diversity, and they achieve this by maximizing the mean of the entropy of predictions in a mini-batch. All these objective functions can be well integrated within our model perturbation framework. We report the results of model perturbations under different implementations of the likelihood in the experimental section.

Parameter Sharing Strategy We adopt a parameter sharing strategy for the convolutional layers to further reduce the learnable parameters, i.e., all the weights in the convolutional kernel share the same perturbations. Thus, the number of learnable parameters for each convolutional layer is equal to the number of output channels. By doing so, we can largely reduce learnable parameters without significantly degrading performance. The feasibility of the parameter sharing strategy lies in the fact that Bayesian neural networks have heavy redundancy. For example, Blundell et al. [19] have shown that BNNs still perform well even with 95% of the parameters pruned.

Through the parameter sharing strategy, we deconstruct the pre-training and adaptation of the model into a static *invariant part* and a dynamic *varying part*: we can train a model with millions of parameters (*static part*) in the source domain, and then we adapt it to different target domains by learning a small number of perturbation parameters (*varying part*), which avoids training the parameters of the whole model every time, reduces the storage space and makes the whole process of domain adaptation more flexible.

3 Related Works

Unsupervised Domain Adaptation Unsupervised domain adaptation (UDA) aims at transferring knowledge from the source domain to the unlabeled target domain in order to perform as well on the target domain. According to Ben-David's domain adaptation theory [25, 26], the expected error of the model on the target domain consists of three components:

$$r_T \leq r_S + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) + \lambda. \tag{10}$$

Here r_S represents the expected error in the source domain, which can be minimized with a cross-entropy loss. $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T})$ is a measure of domain shifts. λ is considered to be a sufficiently small constant to represent the complexity of the hypothesis space [27, 28]. Therefore, most UDA methods focus on the second term, i.e., the domain shift, to minimize the expected error on the

Table 1: Difference between our model perturbation and related methods. Model perturbation is the only source-free online domain adaptation method that introduce uncertainty to the model without any extra requirement.

	Source-Free	Online	Uncertainty	Adaptation Scheme	Extra Requirement
Tent [14]	✓	✓	×	-	×
SHOT [5]	✓	×	×	Conv + FC	×
TTT [33]	×	✓	×	Conv + FC	auxiliary task
BACS [35]	✓	✓	✓	Conv + FC	multiple models
This paper	✓	✓	✓	Perturbations	×

target domain. For example, methods based on adversarial generative networks [27, 1] learn domain-indistinguishable features in an adversarial manner. Statistical matching methods [29, 28, 30] try to mitigate domain shifts by aligning the first- or second-order statistics between domains.

Source-Free Domain Adaptation Unsupervised domain adaption assumes that source data is available when adapting to target domains, which would not always hold in real-world applications due to issues of intellectual property, individual privacy, decentralisation, etc. Therefore, Source-Free Domain Adaptation (SFDA) is proposed to solve this problem [3, 4, 5, 6, 31, 32]. In the setting of SFDA, only the pre-trained model in the source domain and the unlabeled target data are provided. Kundu et al. [4] attempt to align the source and target domains using generative models. Ishii and Sugiyama [32] reduce domain shifts by aligning the statistics stored in the batch normalization layers. From an information-theoretic perspective, Liang et al. [5] proposes SHOT to fine-tune the source model using target data by imposing entropy minimisation and diversity maximisation for predictions. In addition, SHOT employs self-supervised learning using pseudo-labels. Yang et al. [6] exploit the intrinsic neighborhood structure to encourage label consistency among target data. Huang et al. [31] maintain the consistency of features between the historical model and adapted model through contrastive learning.

Online Source-Free Domain Adaptation SFDA methods [5, 4, 6, 31] assume all target domain data are available during training, and they also optimize dedicated objective functions through multiple iterations. To be applicable to scenarios such as autonomous driving and real time object tracking online SFDA, also known as test-time adaptation, has been explored. For example, AdaBN [15] and Test-time BN [16] improve the performance of the model in the target domain by recalculating the Batch Normalization (BN) statistics in the test period instead of using fixed BN statistics from the pre-trained model. Tent [14] modulates the parameters in the BN layers through entropy minimization to adapt the test data in real time. For stable and effective adaptation, Tent freezes all the parameters except the BN layers. Test-Time-Training [33] introduces additional auxiliary tasks to improve test-time performance. Cotta [34] adopts weight-averaging and augmentation-averaging to calibrate pseudo-labels to avoid error accumulation during continual test-time adaptation, then they use these reliable pseudo-labels for self-supervised regularization. By contrast, our model perturbation freezes all weights except the batch normalization parameters in the source model, which avoids error accumulation and knowledge forgetting.

We provide a comparison between the proposed model perturbation and related methods in Table 1. Most existing approaches are developed in a deterministic manner, without exploring the uncertainty. Differently, in this work we address source-free domain adaption in a probabilistic framework by modelling weight uncertainty.

Bayesian Neural Networks Our work is also closely related to Bayesian neural networks. They combine neural networks with Bayesian inference, which explain the uncertainties from the model and provides the distributions over the weights and outputs. Various techniques have been explored to learn Bayesian neural networks including variational inference [36, 22], probabilistic backpropagation [37], Hamiltonian Monte Carlo [38] and the Laplace approximation [39]. Our variational model perturbation provides an alternative way of implementing Bayesian inference for neural networks. Compared to previous implementations, our method is plug-and-play and more flexible. It provides an efficient way to adapt a deterministic source model to a target domain by transforming into a probabilistic one while avoiding re-training a full Bayesian neural network.

4 Experiments

4.1 Datasets and Settings

Datasets. We report our experiments on five datasets. The **Office** [40] dataset includes 3 domains. All the images are categorized into 31 classes. **Office-Home** [41] consists of 4 domains. Each domain contains 65 different classes. **Cifar10-C**, **Cifar100-C** [42] and **ImageNet-C** [42] include 15 different types of corruptions imposed on the original Cifar10, Cifar100 and ImageNet datasets, respectively. Each corruption type contains 5 different corruption severities. Each severity has 10,000 images in Cifar10 and Cifar100, 5,000 images in ImageNet-C, respectively.

Evaluation Settings. We test our method on three source-free domain adaptation settings. *Offline SFDA* [4, 3]: In the training phase, we can access data of the entire target domain and optimize the objective functions with multiple iterations. *Generalized SFDA* [43]: The model is first trained using 80% source data and then tested using the rest of the source data and all the target data. In this setting, we compute the harmonic mean of the source and target accuracies: $H = \frac{2 * \text{Acc}_S * \text{Acc}_T}{\text{Acc}_S + \text{Acc}_T}$. *Continual Online SFDA* [34]: At each time step, we can only access data in one batch and data in the previous time step cannot be accessed again. We carry out this experiment in a more challenging setting where data of different domains are encountered continually.

4.2 Implementation Details

For Offline SFDA and Generalized SFDA, we employ the same losses as in SHOT [5] to optimize the expected likelihood in Eq. 3. For a fair comparison, we use the same network architectures just as the previous methods [5, 1, 43, 31]. Specifically, for Office and Office-Home, we use a ResNet-50 pre-trained on ImageNet as the backbone. As for the optimizer, following [5], we employ Stochastic Gradient Descent (SGD) with weight decay 1e-3 and momentum 0.9. As for the learning rate, we set 2e-3 for the backbone model and 2e-2 for the bottleneck layer newly added in SHOT. In addition, β is a hyper-parameter which balances the importance of information loss and self-supervision loss in SHOT. Since the parameters learned by model perturbations are different from those of SHOT, the value of β used in SHOT may not be optimal. Therefore, we search the optimal value β^* in range of [0.1,2.0] for all the three datasets. Then, β^* is set to 0.3 in both Office and Office-Home. The batch size is 64.

For Continual Online SFDA, we use entropy loss as the likelihood function in Eq. 3. In this way, the implementation is equivalent to a perturbation of Tent [14]. We use the same network architectures as in [34] to make the results comparable. In Cifar10-C, we employ WideResNet-28 [44] from Robust Bench [45] as the backbone network. We used the Adam [46] optimizer with learning rate 5e-4. In Cifar100-C tasks, we used ResNeXt-29 [47] from [48]. The optimizer settings are the same as that in Cifar10-C except the learning rate is 7e-5. In the ImageNet-C experiments, a pre-trained ResNet-50 is employed. We adopt the SGD optimizer with learning rate 1e-4, momentum 0.9 and weight decay 0. We set the scale parameter λ of the prior to be 1.0. We discuss the impact of different choices of priors on performance in 4.3. In the training phase, we sample the perturbations once for running efficiency. In the test phase, we sample the perturbations 10 times. To reduce the sampling variance, we employ the local reparameterization trick [49]. We also discuss the effect of different numbers of Monte Carlo samples on performance in the training phase in the supplementary material.

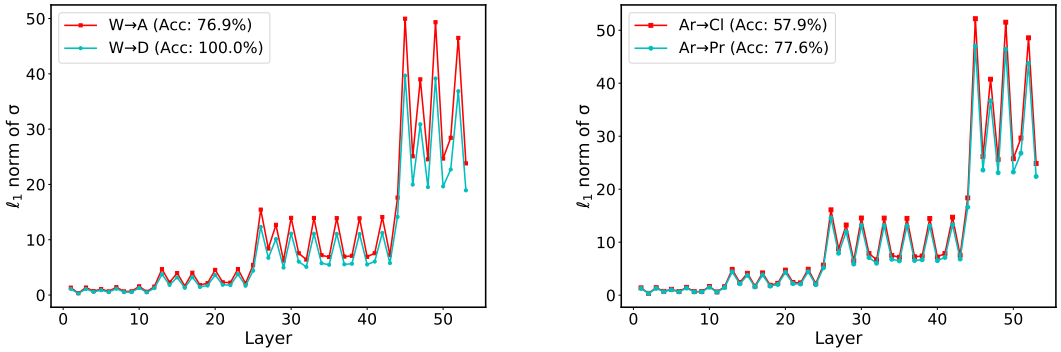
In the test phase, given a batch of target samples, we iterate the proposed method for only 1 epoch and make predictions immediately. As for the compared methods, we either report the results in the original papers or the best results we can reproduce. For the non-Bayesian methods, we run them 10 times with different random seeds and take the average of them to reduce randomness.

Apart from SHOT and Tent, we have also applied our variational model perturbation to other methods.

4.3 Effectiveness Analysis

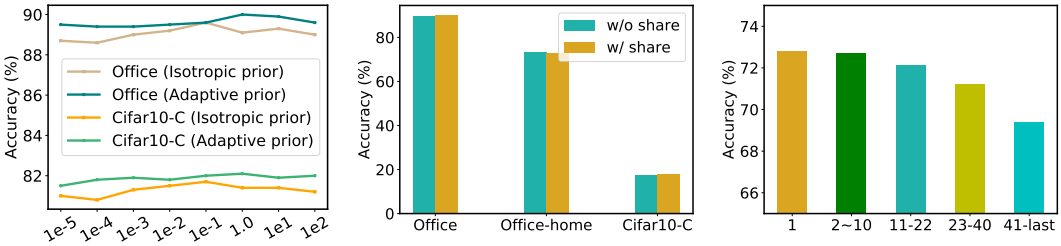
We make effectiveness analysis of our method in Fig. 1, Fig. 2 and Fig. 3. Results of offline setting and online setting are from implementations of model perturbation based on SHOT and Tent, respectively.

Quantification of weight uncertainty. In Fig. 1, we report the ℓ_1-norm of the learned σ in each layer. We use it as a proxy of weight uncertainty. We could make the following observations:



(a) Office-31 (b) Office-Home

Figure 1: We measure the weight uncertainty by ℓ_1 norm of σ in each layer of ResNet-50. σ of the difficult task (e.g., W→A) has a larger ℓ_1 norm than the easy task (e.g., W→D), and correspondingly the perturbations vary over a larger range, i.e., a larger uncertainty. Therefore, the perturbations we learned can be regarded as a measure of uncertainty, which reflects the distribution shifts between domains. Here we use the accuracy produced by the adapted model as an indicator of domain gap, where lower accuracy means a larger domain gap.



(a) Scale factor of prior (b) Parameter sharing (c) Layers w/o perturbation

Figure 2: (a) Impact of different variance scale factors; (b) Comparison of performance with and without parameter sharing strategy; (c) Impact of layers without perturbations where the results are based on Office-Home dataset and the backbone is ResNet-50.

(1) In the first few layers of the network, the perturbations are small. As the layers go deep, the perturbations become larger. This shows that the shallower layers of the neural networks are more generalized, while the deep layers are task-specific, which is consistent with the findings in the previous work [50, 28]; (2) The extent of the perturbation is task-dependent. A larger domain shift of the task needs a larger perturbation, which demonstrates that model perturbation can effectively encode uncertainty into the network, thus enhancing the generalization.

Adaptive prior. In Fig. 2 (a), we report the impact of the variance scale factor of the prior on performance. Compared to the isotropic Gaussian prior, our adaptive prior has better performance and smaller fluctuations for different scale factors. This indicates that it is reasonable to use an adaptive prior as different weights may have different variation ranges when adaptation. Our adaptive prior can specify different prior distributions for different weights according to the statistical properties within the convolutional kernels, leading to a better performance.

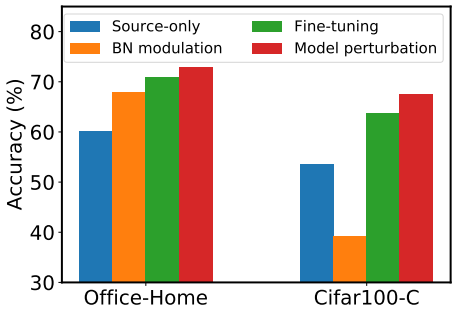


Figure 3: Comparison of different adaptation schemes. Model perturbation outperforms BN modulation and fine-tuning on both offline tasks (Office-Home) and online tasks (Cifar100-C).

Parameter sharing strategy. In Fig. 2 (b) we observe that the parameter sharing strategy achieves almost exactly the same performance compared with the non-sharing strategy. For ResNet-50 based tasks, e.g., Office and Office-

Table 2: Offline SFDA accuracy (%) on Office and Office-Home, where all methods use a ResNet-50 backbone. Our variational model perturbation performs best on the majority of target domains and results in best average accuracy. All per domain results are provided in the Appendix.

		Office		**Office-Home**	
	Source-free	Winning (6)	Average	Winning (12)	Average
DAN [28]	×	0	80.4	0	56.3
DANN [51]	×	0	82.7	0	57.6
BDG [52]	×	2	88.5	1	68.7
MDD [53]	×	0	88.0	1	68.1
BNM [54]	×	1	87.1	0	67.9
MCD [27]	×	1	86.5	0	64.1
SAFN [55]	×	0	87.1	0	67.3
CDAN+E [1]	×	2	87.7	0	65.8
Source-only	✓	0	79.3	0	60.2
SHOT [5]	✓	0	88.6	1	71.8
Ours	✓	**4**	**90.0**	**9**	**72.8**

Table 3: Generalized SFDA accuracy (%) on Office-Home, where all the methods use a ResNet-50 backbone. Our variational model perturbation effectively adapts to the target domain while still maintaining the discriminative ability on the source domain.

	Office-Home		
	Source	Target	Harmonic
Source-only	83.6	59.1	69.2
SHOT [5]	71.7	70.4	71.0
Ours	**76.6**	**71.6**	**74.0**

Home, the learnable parameters reduce from 24.03 million to 600 thousand compared with the non-sharing strategy. As for the WideResNet-28 based tasks, e.g., Cifar10-C, the learnable parameters reduce from 36.47 million to 30 thousand. These results verify the effectiveness of the parameter sharing strategy.

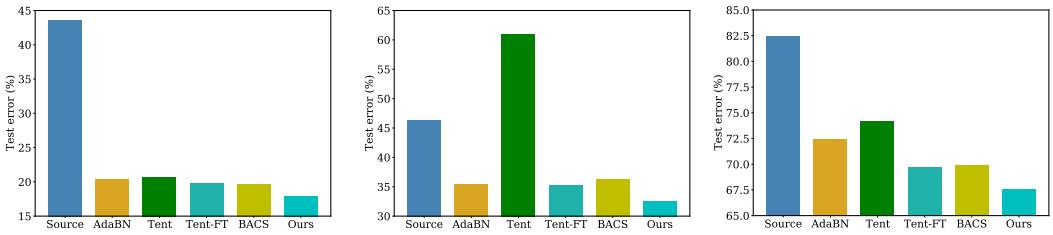
Which layers are crucial for perturbation? We test the performance by keeping some of the network layers not perturbed. As shown in Fig. 2 (c), we observe that the performance of the model degrades significantly when removing perturbations in the last few layers of the network, revealing that perturbations for the last few layers are crucial, since these layers are often task-relevant.

Adaptation schemes. In Fig. 3, we compare results of different updating schemes on Office-Home and Cifar100-C. BN modulation means only updating the statistics in batch normalization layers. Finetuning represents updating all the weights in the model. We observe that model perturbations outperforms BN modulation and fine-tuning in both the offline and online SFDA tasks. Compared with BN modulation, model perturbation is more flexible. It can generalize the model to more target domains through perturbations. Compared with Fine-tuning, model perturbation avoids the distortion of weight in source model, which prevents the knowledge in source domain from being forgotten in continuous adaptation. Therefore, model perturbation is the better choice in SFDA tasks.

4.4 Results

In this section, we provide the experimental results on five datasets under three different evaluation settings. We also put more detailed results in the supplemental material.

Offline SFDA We report the comparison results on Office and Office-home in Table 2. All per domain results are provided in the Appendix. Our model perturbation achieves the best average results on Office, showing the stronger generalization performance of the perturbed model. Moreover, our model perturbation achieves the best results on 4 out of 6 tasks. In particular, the implementation based on SHOT has improved by 1.4% compared with SHOT, which is impressive. A similar trend can be observed in the Office-Home dataset, our model perturbation wins 9 taskes across all the 12



(a) Cifar10-C (b) Cifar100-C (c) ImageNet-C

Figure 4: Test errors (%) under the continual online setting. The backbone of Cifar10-C, Cifar100-C and ImageNet-C are WideResNet-28, ResNeXt-29 and ResNet-50. The corruption severity is 5. Model perturbations achieves the best performance across all the tasks.

tasks. In addition, our method outperforms SHOT by 1.0%, which proves that model perturbation can consistently and clearly improve existing methods.

Generalized SFDA In Table 3, we report the results of model perturbation on Generalized SFDA tasks. In this setting, we first train the source model with 80% of the source data, and then adapt the model to the target domain. Finally, we evaluate the model on both the target domain and the remaining 20% of the source data. We observe that the performance of SHOT on the source domain decreases by 11.9% compared to the source model, indicating that SHOT forgets the knowledge learned in the source domain. Compared to SHOT, model perturbation maintains the performance on the target domain while significantly improving the performance on the source domain, revealing that our model perturbation can maintain a good balance between adaptation and preserving knowledge.

Continual Online SFDA In Fig. 4, we report the test errors of different methods under the continual online SFDA, where all the methods continuously encounter 15 different types of corruption. This experiment can not only test the real-time prediction performance of the compared methods, but also show the degree of knowledge forgetting. Note that Tent [14] only updates the parameters in BN layers while Tent-FT implemented by us updates all the weights. Similar to Tent, BACS (MAP) [35] also employ entropy minimization, with the difference that BACS (MAP) optimizes the model through maximum-a-posteriori (MAP).

In Fig. 4 (a), both AdaBN and Tent have test errors larger than 20.0%, revealing that the inflexibility of freezing the weights is the bottleneck for adapting to the continuously changing target domains. By unfreezing the parameter update of the convolutional and FC layers, Tent-FT improves Tent by 0.9%. Compared to Tent and Tent-FT, the implementation of model perturbation based on Tent achieves performance advantages of 2.8% and 1.9%, respectively, demonstrating that model perturbation is the better choice for SFDA tasks. The performance advantages stem from two aspects. On one hand, model perturbation fixes all the weights of convolutional and FC layers to avoid large distortion of the weights and forgetting knowledge on the source domain. On the other hand, model perturbation introduces uncertainty into the model by slightly disturbing the weights, allowing it to adapt to continuous distribution shifts. Similar conclusions can be drawn on Cifar100-C and ImageNet-C.

5 Conclusion

In this paper, we address source-free domain adaptation in a probabilistic framework by variational model perturbation. Instead of updating the network parameters by further optimization, we introduce perturbations to the weights of a pre-trained model from source domains. We learn these perturbations through variational Bayesian inference. Importantly, we prove the connection between variational model perturbation and Bayesian learning of a neural networks, which provides a theoretical guarantee for the generalization performance of model perturbation. Moreover, with the parameter sharing strategy, we substantially reduce the learnable parameters compared to a fully Bayesian neural network, making the model perturbation a flexible and lightweight framework. Our variational model perturbation offers a new, probabilistic framework for source-free domain adaptation, which can achieve efficient adaptation while largely preserving the discriminative ability of the source model. The experimental results on five benchmark datasets under three different evaluation settings reveal that model perturbation offers an effective way for source-free domain adaptation and can boost the performance considerably.

References

[1] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, pages 1640–1650, 2018.

[2] Jaemin Na, Heechul Jung, Hyung Jin Chang, and Wonjun Hwang. Fixbi: Bridging domain spaces for unsupervised domain adaptation. In *CVPR*, pages 1094–1103, 2021.

[3] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9641–9650, 2020.

[4] Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. Universal source-free domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4544–4553, 2020.

[5] 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.

[6] 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.

[7] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, volume 30, 2017.

[8] Ju Xu and Zhanxing Zhu. Reinforced continual learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[9] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[10] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[11] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2021.

[12] T Zhang and FJ Oles. A probability analysis on the value of unlabeled data for classification problems. In *International Conference on Machine Learning*. PMLR, 2000.

[13] Matthias Seeger. Input-dependent regularization of conditional density models. Technical report, 2000.

[14] 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.

[15] 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.

[16] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020.

[17] 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.

[18] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems*, volume 33, pages 4697–4708, 2020.

[19] 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.

[20] Zehao Xiao, Jiayi Shen, Xiantong Zhen, Ling Shao, and Cees G M Snoek. A bit more bayesian: Domain-invariant learning with uncertainty. In *International Conference on Machine Learning*. PMLR, 2021.

[21] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, pages 177–186. Springer, 2010.

[22] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

[23] Vincent Fortuin, Adrià Garriga-Alonso, Sebastian W. Ober, Florian Wenzel, Gunnar Ratsch, Richard E Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian neural network priors revisited. In *International Conference on Learning Representations*, 2022.

[24] Daniele Silvestro and Tobias Andermann. Prior choice affects ability of bayesian neural networks to identify unknowns. *arXiv preprint arXiv:2005.04987*, 2020.

[25] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *NeurIPS*, 19:137, 2007.

[26] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[27] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, pages 3723–3732, 2018.

[28] Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Transferable representation learning with deep adaptation networks. *TPAMI*, 41(12):3071–3085, 2018.

[29] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *ICCV*, pages 2200–2207, 2013.

[30] 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.

[31] Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. Model adaptation: Historical contrastive learning for unsupervised domain adaptation without source data. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[32] Masato Ishii and Masashi Sugiyama. Source-free domain adaptation via distributional alignment by matching batch normalization statistics. *arXiv preprint arXiv:2101.10842*, 2021.

[33] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, pages 9229–9248. PMLR, 2020.

[34] 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.

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

[36] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

[37] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pages 1861–1869. PMLR, 2015.

[38] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[39] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[40] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, pages 213–226. Springer, 2010.

[41] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, pages 5018–5027, 2017.

[42] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[43] 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.

[44] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.

[45] 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.

[46] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[47] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[48] 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.

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

[50] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328, 2014.

[51] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[52] Guanglei Yang, Haifeng Xia, Mingli Ding, and Zhengming Ding. Bi-directional generation for unsupervised domain adaptation. In *AAAI*, volume 34, pages 6615–6622, 2020.

[53] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pages 7404–7413. PMLR, 2019.

[54] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3941–3950, 2020.

[55] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *ICCV*, pages 1426–1435, 2019.

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]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (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? [N/A]
   (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? [No]
   (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]