Continual Domain Adaptation through Pruning-aided Domain-specific Weight Modulation

Prasanna B
bprasanna@iisc.ac.in

Sunandini Sanyal
sunandinis@iisc.ac.in

R. Venkatesh Babu
venky@iisc.ac.in

Vision and AI Lab, Indian Institute of Science, Bengaluru

Abstract

In this paper, we propose to develop a method to address unsupervised domain adaptation (UDA) in a practical setting of continual learning (CL). The goal is to update the model on continually changing domains while preserving domain-specific knowledge to prevent catastrophic forgetting of past-seen domains. To this end, we build a framework for preserving domain-specific features utilizing the inherent model capacity via pruning. We also perform effective inference using a novel batch-norm based metric to predict the final model parameters to be used accurately. Our approach achieves not only state-of-the-art performance but also prevents catastrophic forgetting of past domains significantly. Our code is made publicly available.[1].

1. Introduction

Deep learning models often fail to perform well on unseen data (*target domain*) that is different from their training data (*source domain*) distribution (referred to as *domain-shift*). Domain adaptation techniques [5, 23, 25] seek to address this problem of domain shift by adapting the source model to the new target domain. However, they fail to generalize well in the case of multiple sequentially changing domains. In our work, we explore a challenging setting of continual learning (CL) in domain adaptation, where the goal is to keep adapting the model over several sequential domain shifts. For instance, a model trained on clear weather data must adapt to other weather conditions such as snowy, foggy, and rainy to achieve optimal performance [27]. Therefore, developing a robust approach toward continual domain adaptation is essential for real-world deployment scenarios. Also, we assume a practical setting of source-free DA [10] where data sharing across domains is restricted due to privacy concerns, as seen in most real-life scenarios.

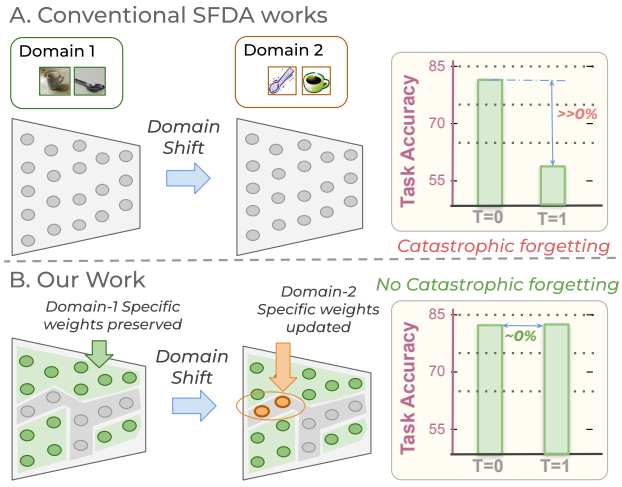

Figure 1. **A.** Conventional Source-Free Domain Adaptation (SFDA) works do not preserve the source domain performance after domain shift occurs, leading to catastrophic forgetting and a sharp decline in the source domain accuracy. **B.** Our method preserves domain-specific characteristics inherently within the model to prevent catastrophic forgetting of previously-seen domains.

Prior works on source-free domain adaptation [15, 30] aim to enhance the in-domain performance by adapting the source model to the incoming target data. However, such an approach depletes the crucial domain-specific knowledge of the source domain, resulting in catastrophic forgetting (as shown in Fig. 1). As a result, these methods fail to scale sequentially changing domains. A possible solution is to store a set of samples from each domain [21], but such methods are ineffective for real-world applications with privacy concerns (where data sharing between parties is restricted) or high memory limitations (e.g., in mobile devices).

In our work, we provide a way to preserve the domain-

[1]Github Page: https://github.com/PrasannaB29/PACDA

specific properties within the model and prevent catastrophic forgetting without requiring additional storage. We also learn domain-specific features for a new target domain while maintaining the performance on the previously encountered domains. Since the source domain is inaccessible after source-side training, we explore the inherent potential of the model to preserve domain-specific features using pruning. We find that a fraction of model parameters is sufficient to preserve the domain-specific statistics. Hence, we develop a novel pruning-based algorithm **P**runing-**a**ided **C**ontinual **D**omain **A**daptation (**PaCDA**) for the challenging task of continual domain adaptation. We also propose a novel **B**atch **N**orm **S**tatistic **D**eviation (**BNSD**) metric to evaluate which model parameters to be used during inference for a test domain.
We outline the major contributions of our work as follows:

- We investigate and provide pruning-based results on how a model's inherent capacity could be leveraged to store domain-specific features. To this end, we develop our novel framework **P**runing-**a**ided **C**ontinual **D**omain **A**daptation (**PaCDA**) for enhancing domain-specific knowledge in the model
- We define a novel **B**atch **N**orm **S**tatistic **D**eviation (**BNSD**) metric for model parameter selection during inference. We also demonstrate the effectiveness of the proposed metric on the Office-Home dataset.
- We achieve state-of-the-art performance on a continual DA benchmark, significantly reducing catastrophic forgetting of the past seen domains.

2. Related Works

2.1. Source-free UDA

Unsupervised Domain adaptation [2, 19, 20, 24, 28, 32] aims to adapt a model trained on a source domain to a new target domain. Source-free domain adaptation (SFDA) [14, 30] is a more constrained setting where the source domain data is not accessible during target adaptation. However, these methods are directly not applicable for a continual learning setting [21] where the goal is to retain performance on previously trained domains.

2.2. Continual Domain Adaptation

Prior works [21, 29] propose to tackle the novel setting of continual DA where domain shifts occur sequentially. Rostami et al. [21] use experience replay from a memory buffer that stores a fixed number of confident samples per class per domain. This buffer is appended with the current training data. Our work adheres to the restriction of source-free DA [10] since storing samples for replay is prohibited due to privacy restrictions. On the other hand, GSFDA [31] does not use experience replay but assumes the availability of domain-id of the inference samples to use appropriate domain attention vectors. Our work PaCDA uses a novel strategy to identify the domain of the incoming batch of samples during inference time and does not require prior knowledge of the domain to which inference samples belong.

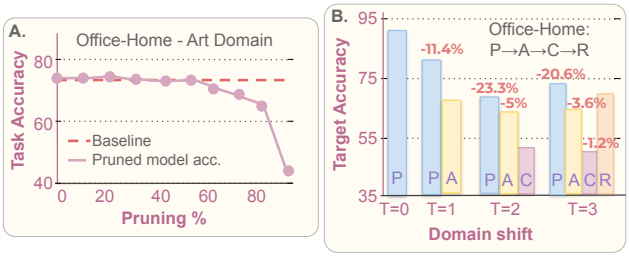

Figure 2. **A.** Accuracy vs. Pruning % on a model trained using Art domain data from Office-home. Note that, until 50% pruning, the model's accuracy is approximately as good as the original model. The model's performance does not deteriorate significantly until 70% pruning. **B.** Catastrophic forgetting of previously trained domains over PACR sequence. Source domain acc. drops by 23.3%, which significantly impacts performance.

2.3. Pruning for Incremental/Continual Learning

Prior works [11, 13] use knowledge distillation strategies to minimize forgetting in a task-incremental setting. Pruning-based works [7, 16, 18, 22, 33] aim to sparsify and create different network parameters for every new task. PackNet [18] uses iterative pruning for learning new tasks and retains the performance on previously learned tasks while training on subsequent tasks. Pruning-based DA works also exist in medical imaging [1] and automatic speech recognition [17]. We draw motivation from these prior pruning works and aim to leverage weight-masking for faster domain adaptation without increasing the model's parameters.

3. Approach

3.1. Notation

We consider a labeled source domain dataset $\mathcal{D}_s = \{(x_s, y_s) : x_s \in \mathcal{X}, y_s \in \mathcal{Y}\}$ where $\mathcal{X}$ denotes the input data space and $\mathcal{Y}$ denotes the task label set. We operate in a practical setting of continual DA [21] where the source-trained model encounters new target domains sequentially and data-sharing among domains is restricted. Our work follows a practical source-free vendor-client setting [10] where a vendor shares a source model but with multiple clients, one after the other. Hence, the vendor-trained source model encounters different target domains sequentially. Our goal is to update this source-trained model continually on the subsequent target domains $\mathcal{D}_t = \{x_t : x_t \in \mathcal{X}\}$ where $t \in \{1, 2, ...\text{T}\}$, such that the model achieves best performance on all the domains and

catastrophic forgetting on past seen domains are minimized. Please note that we assume that all domains share the same task label set $\mathcal{Y}$.

Prior works of domain invariance [6] propose to capture domain-invariant features that generalize well to multiple domains. However, an optimal model requires domain-specific knowledge [3] to achieve the best performance. Conventional SFDA works [14,15,30] enhance the domain-specific knowledge to improve adaptation performance, but this often leads to catastrophic forgetting of the past seen source domain. As a Baseline, we extend the SHOT approach [14] to a Continual DA setting (shown in Table 1), where we adapt to successive domains using SHOT without any mechanism to mitigate catastrophic forgetting. This leads to a sharp decline in the accuracy of past-seen domains, proving the unsuitability of the SFDA methods in a practical scenario of continual DA. In practice, a deployed model often encounters data from different domains sequentially. Hence, it is crucial to leverage the knowledge learned from the old domains to not only improve learning in new domains but also prevent catastrophic forgetting. Therefore, in our work, we propose to address the critical question: *"How do we build a framework for learning continually changing domains while maintaining good performance on all previously-seen domains?"*

3.2. Exploiting the inherent potential of the source model

In order to build our framework, we propose to preserve the domain-specific knowledge for each domain in the model itself to prevent forgetting. The Lottery Ticket Hypothesis [4] states that a subset of model parameters, i.e., a "winning" ticket, is sufficient for attaining the desired task accuracy. So pruning away the other parameters does not cause a significant performance drop. To investigate this phenomenon in the context of domain adaptation, we first train a model on a source domain (domain Art in Office-Home) and observe the drop in the task accuracy after different percentages of L1-based weight pruning [7,8]. Figure 2 shows that the task performance declines only after 50% of the model parameters are pruned. This demonstrates that the inherent redundancy of a neural network could be leveraged to compress the model capacity and utilize the remaining parameters for storing additional domain-specific knowledge encountered in future domains. Hence, this motivates us to explore a pruning-based framework for domain adaptation in a continual DA setting.

3.3. Training algorithm

We propose our novel algorithm **P**runing-**a**ided **C**ontinual **D**omain **A**daptation (**PaCDA**), that mainly comprises of Domain-specific weight selection and training phase, and an inference phase. The vendor uses the weight selection algorithm to prepare and share the model with the client. During inference, the client uses a batch norm based inference metric to predict the network mask and identify the model parameters for the incoming data batch. The following sections explain the details of each component of the algorithm.

3.3.1 Source Model Training

The source model consists of two modules: the feature encoding module $g_s : \mathcal{X} \rightarrow \mathcal{R}^d$ and the classifier module $h_s : \mathcal{R}^d \rightarrow \mathcal{R}^K$, i.e., $f_s(x) = h_s\left(g_s(x)\right)$. Here, d denotes the feature dimension, and K denotes the number of task labels. The source model is trained on labeled source domain data $\mathcal{D}_\mathcal{S}$ by minimizing the cross-entropy loss as follows:

$$\mathcal{L}_s(f_s; x_s, y_s) = -\mathbb{E}_{(x_s,y_s)\in\mathcal{X}\times\mathcal{Y}} \sum_{k=1}^{K} q_k^{ls} log\delta_k(f_s(x_s)) \tag{1}$$

where q_k^{ls} is obtained by label smoothing [14].

Domain-Specific Weight Selection: At first, the entire source model is trained with the supervised cross-entropy loss. Once the training completes, we use a procedure of L1-norm-based pruning to prune $1 - p_0$ fraction of the parameters, retaining p_0 fraction for the source domain 0. The weights in each layer of the backbone feature extractor and bottleneck layers are pruned by $1 - p_0$ fraction, while the classifier layer is not. Once pruning is done, we fine-tune the p_0 fraction of the model parameters to regain the performance. We refer to these weights as domain-specific weights of the source since they hold the crucial source domain features. After this, the vendor sends the fine-tuned network to the client side for adaptation.

3.3.2 Client-side Adaptation

The client-side adaptation is done by optimizing on Information-Maximization (IM) loss [14]. The IM loss is a constitution of $\mathcal{L}_{ent}$ and $\mathcal{L}_{div}$ as follows:

$$\begin{aligned}
\mathcal{L}_{ent}(f_t; X_t) &= -\mathbb{E}_{x_t\in X_t} \sum\nolimits_{k=1}^{K} \delta_k(f_t(x_t)) \log \delta_k(f_t(x_t)), \\
\mathcal{L}_{div}(f_t; \mathcal{X}_t) &= \sum\nolimits_{k=1}^{K} \hat{p}_k \log \hat{p}_k \\
&= D_{KL}(\hat{p}, \frac{1}{K}\mathbf{1}_K) - \log K,
\end{aligned} \tag{2}$$

where $f_t(x) = h_t(g_t(x))$ is the K-dimensional output of each target sample, $\mathbf{1}_K$ is a K-dimensional vector with all ones, and $\hat{p} = \mathbb{E}_{x_t\in\mathcal{X}_t}[\delta(f_t(x_t))]$ is the mean output embedding of the whole target domain.

On the client side, target adaptation is performed keeping the source domain-specific weights frozen. Since domains are encountered sequentially, a client with target domain t

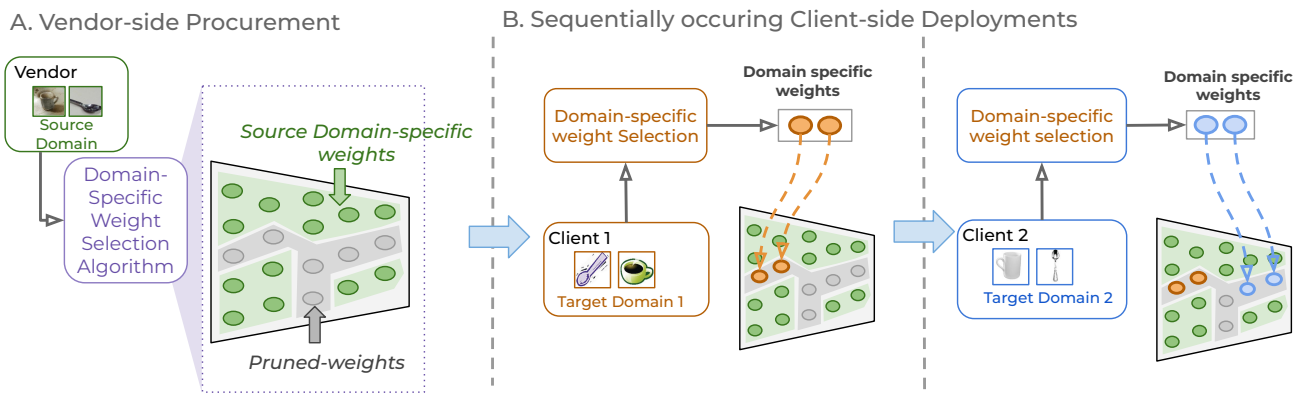

Figure 3. **PaCDA Approach:** ***A. Vendor side Procurement:*** The vendor uses the domain-specific weight selection algorithm to select domain-specific weights into the source model. ***B. Client-side adaptation:*** The vendor-trained Source model is shared with client 1 to update only domain-specific weights from the remaining pruned ones. Client 2 obtains the Client 1 trained model and follows the same procedure. During inference, a batch-norm based metric is used to obtain the final model predictions.

might receive a model trained on the previous $t-1$ domains. For each domain, p_t fraction of domain-specific weights is preserved. Hence, the client uses the domain-specific weight selection algorithm to prune weights from the remaining $1-\sum_{i=0}^{t-1} p_i$ parameters. Once the weights are pruned, fine-tuning is performed to obtain p_t fraction of domain-specific weights for the domain t. We also maintain a separate set of batch norm parameters for each domain as prior works [12] indicate that batch norm layers preserve domain-specific characteristics. A parameter mask M_i indicates the set of active weights for a domain with 0s corresponding to the pruned weights and 1s for the active weights. Also, during the training of domain t, the mask $1-M_{t-1}$ is used to freeze the parameters of previously trained domains. Hence, in this way, the model keeps training for sequentially arriving client domains, ensuring that we preserve domain-specific knowledge for each domain to reduce catastrophic forgetting significantly. These masks need to be stored for use during inference, but since these are binary masks, they can be stored efficiently without significant extra memory requirements.

3.3.3 Batch norm based Inference

Once the model is trained on a source domain and T $-$ 1 target domains, we have T parameter masks and T corresponding Batch Norm layers. During inference, a sample from any of the trained domains might be encountered. Hence, the task is to identify the parameter mask and Batch Norm layer to be chosen as the final model to get the best performance on inference samples. For this, we introduce a novel metric ***Batch Norm Statistic Deviation (BNSD)***. Batch Norm layers for each domain contain the running-mean and running-variance statistics. Note that, we only use the mean and variance statistics of each domain, without storing any additional samples for training or inference. BNSD measures the deviation of the inference batch's mean and variance from the stored running statistics associated with the Batch Norm parameters. We propose to use only the model's first layer batch norm since it is closest to the input layer and captures domain-specific information from the input. The BNSD metric is defined as follows:

Algorithm 1 Pruning-aided Continual Domain Adaptation

Input: Source model $f_s = h_s(g_s)$, Target Data $\mathcal{D}_t = (\mathcal{X}_t)$, where $t \in \{1, 2, ...T\}$, n_epochs_train, n_epochs_finetune
Initialize f_t to f_s and freeze h_t
for each domain d **do**
1. Train f_t for n_epochs_train by masking the gradient using $1-M_{t-1}$
2. Prune $1-\sum_{i=0}^{t} p_i$ fraction of the model using L1-pruning and obtain M_t
3. Store M_t and Batch Norm layers for current domain t
4. Fine-tune f_t for n_epochs_finetune
5. Reset Batch Norm layers to Source Model initialization

end for

$$\begin{aligned}\text{BNSD}(b_i, \text{BN'}, \mathcal{M}'; f) = \\ (\text{Mean}(f_{\mathcal{M}'}(b_i)) - \text{BN'.running-mean})^2 + \\ (\text{Var}(f_{\mathcal{M}'}(b_i)) - \text{BN'.running-var})^2 \end{aligned} \quad (3)$$

where b_i is the i^{th} batch of inference data, BN' is the first Batch Norm layer with which the BNSD of the input batch is calculated, $\mathcal{M}'$ is the mask corresponding to BN' and f is the final model obtained after training on all domains. $f_{\mathcal{M}'}$ is the resultant model to be used when the mask $\mathcal{M}'$ is applied on the model f. The PaCDA algorithm 1 chooses the Batch Norm layer and corresponding parameter mask which gives the least BNSD for a given batch of samples.

Method	A → C → P → R				C → A → P → R				P → A → C → R				R → A → C → P			
	A	C	P	R	C	A	P	R	P	A	C	R	R	A	C	P
Baseline	61.3 (-12.6)	54.7 (-1.3)	68.6 (+0.25)	70.8 (+0)	57.7 (-23.2)	67.7 (-0.6)	73.9 (-0.4)	73.2 (+0)	72.1 (-20.6)	64.5 (-3.7)	51.2 (-1.2)	70.55 (+0)	74.3 (-11.6)	65.7 (-5.0)	55.8 (-1.2)	72.1 (+0)
GSFDA [31]	72.6 (-1.9)	55.6 (-1.0)	**72.7** (-0.3)	**77.2** (+0)	78.6 (-3.6)	64.9 (-0.5)	**72.8** (-0.1)	72.4 (+0)	88.6 (-3.4)	63.1 (-0.5)	51.5 (-1.6)	76.5 (+0)	84.2 (-1.8)	69.1 (-3.3)	57.4 (-1.7)	**80.5** (+0)
Ours(with Domain-id)	73.7 (+0)	56.0 (+0)	71.8 (+0)	76.9 (+0)	80.9 (+0)	67.1 (+0)	72.7 (+0)	**76.4** (+0)	92.9 (+0)	67.2 (+0)	55.8 (+0)	**78.1** (+0)	86.2 (+0)	**73.3** (+0)	58.1 (+0)	79.2 (+0)
Ours(w/o Domain-id)	**73.7** (+0)	**56.0** (+0)	71.7 (-0.1)	76.1 (+0)	**80.9** (+0)	**67.2** (+0.1)	72.6 (-0.1)	75.4 (+0)	**92.9** (+0)	**67.2** (+0)	**55.8** (+0)	77.8 (+0)	**86.2** (+0)	72.8 (-0.3)	**58.1** (+0)	79.2 (+0)

Table 1. Accuracy (%) of each method on various Continual Domain Adaptation sequences of Office-Home dataset using ResNet-50 as the backbone. The values in parentheses denote the difference between the model's accuracy at the end of the sequence and the accuracy right after the training of that domain. The lower the value in the parentheses, the higher the catastrophic forgetting. It can be noted that our method performs significantly better at reducing catastrophic forgetting compared to baseline and GSFDA [31].

4. Experiments and Results

4.1. Dataset

We demonstrate the efficacy of our approach on the **Office-Home** dataset [26], which contains a total of 15,500 images belonging to 4 domains with 65 categories each. We propose to use this dataset since it contains a sizeable inter-domain gap and the effects of catastrophic forgetting would be more evident.

4.2. Implementation details

We follow the same experimental setting as SHOT [14] and initialize the backbone with an ImageNet pre-trained ResNet-50 [9] model. We optimize the loss mentioned in Eq. 1, Eq. 2 using Stochastic Gradient Descent with a momentum of 0.9 and weight decay of $1e^{-3}$. We ran our experiments on 12 GB NVIDIA GeForce GTX Titan X GPU.

4.3. Comparison with other methods

In Table 1, we evaluate our method against other methods in terms of accuracy(%) at the end of each sequence and catastrophic forgetting relative to the accuracy right after training. The Baseline method (shown in Table 1) is a sequential training approach of domains using SHOT [14] method. Since SHOT is well-suited to maximise accuracy on target domains only, we observe significant catastrophic forgetting across all sequences. We evaluated both variants of our method - with Domain-id (having access to domain-ID of inference samples) and w/o domain-ID (using BNSD metric to map the inference batch to the appropriate domain-specific parameter mask). We also compare our results with GSFDA [31], which reported Continual DA results for the same four Office-Home sequences. We also use the same 0.8/0.2 train-test split on the source domain as GSFDA, and source-domain accuracies are reported on the test split. Our method not only achieves higher accuracy than GSFDA, but even without using domain-id, PaCDA significantly reduces catastrophic forgetting.

	A	C	P	R		C	A	P	R
A	**73.7**	44.1	64.5	73.1	**C**	**80.9**	50.8	60.1	63.7
C	67.1	**56.0**	60.4	67.7	**A**	70.33	**67.1**	65.5	69.5
P	66.3	53.0	**71.8**	73.3	**P**	72.3	66.7	**72.7**	75.5
R	68.1	53.1	71.4	**76.9**	**R**	71.0	67.6	71.7	**76.4**
	P	**A**	**C**	**R**		**R**	**A**	**C**	**P**
P	**92.9**	52.1	41.7	73.0	**R**	**86.2**	64.8	45.8	76.7
A	84.5	**67.2**	44.1	74.2	**A**	82.6	**73.3**	48.7	75.1
C	80.0	63.0	**55.8**	69.9	**C**	78.0	67.4	**58.1**	70.2
R	84.3	66.6	52.8	**78.1**	**P**	80.5	68.2	55.2	**79.2**

Table 2. Accuracy of each domain data when inferred using different domain-specific weights for four sequences on the Office-Home dataset. Each quadrant is a sequence in which parameter mask changes along the rows and input domain changes along the column. In each column of a sequence, the bold value is the expected best accuracy, and the underlined value is the actual best accuracy obtained.

Our inference process, using the BNSD metric, selects the suitable mask and batch norm layer almost perfectly, and the performance of our PaCDA method without domain-id is almost as good as PaCDA with domain-id method.
In Table 2, for the same set of sequences, we evaluate the accuracy of each domain when using different parameter masks (and their corresponding batch norm layers) on the final model obtained at the end of the sequence. As expected, the corresponding parameter mask for each domain gives the best results. One intriguing result to note is, in Table 2 sequence CAPR, the accuracy on domain A using parameter mask of R is slightly higher than the accuracy obtained using parameter mask of A. This also reflects in Table 1 sequence CAPR, when inferred on domain A, PaCDA w/o Domain-id accuracy is higher than with domain-id.

5. Conclusion

In this work, we address the practical problem of Continual Domain Adaptation using a pruning-aided approach (PaCDA). At first, we investigate and find that pruning model parameters provides room for storing crucial

domain-specific knowledge. We introduce a novel domain-specific weight selection algorithm for multiple client domains and reduce catastrophic forgetting of past-seen domains to a vast extent. We use a novel batch norm based metric to infer on test domain batches during inference. Hence, we not only achieve state-of-the-art results on standard DA benchmarks, but our results also demonstrate that appropriate model parameters are chosen during inference. As a part of future analysis, we plan to extend our work using model expansion approaches to enable the learning of a larger number of domains sequentially.

References

[1] Nourhan Bayasi, Ghassan Hamarneh, and Rafeef Garbi. Culprit-prune-net: Efficient continual sequential multi-domain learning with application to skin lesion classification. In *Medical Image Computing and Computer Assisted Intervention - MICCAI 2021 - 24th International Conference, Strasbourg, France, September 27 - October 1, 2021, Proceedings, Part VII*, volume 12907 of *Lecture Notes in Computer Science*, pages 165–175. Springer, 2021. 2

[2] Jiahua Dong, Yang Cong, Gan Sun, Yuyang Liu, and Xiaowei Xu. Cscl: Critical semantic-consistent learning for unsupervised domain adaptation. In *ECCV*, 2020. 2

[3] Abhimanyu Dubey, Vignesh Ramanathan, Alex Pentland, and Dhruv Mahajan. Adaptive methods for real-world domain generalization. In *CVPR*, 2021. 3

[4] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 3

[5] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 1

[6] 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. 3

[7] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Shijian Tang, Erich Elsen, Bryan Catanzaro, John Tran, and William J. Dally. DSD: regularizing deep neural networks with dense-sparse-dense training flow. *CoRR*, abs/1607.04381, 2016. 2, 3

[8] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015. 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[10] Jogendra Nath Kundu, Naveen Venkat, M V Rahul, and R. Venkatesh Babu. Universal source-free domain adaptation. In *CVPR*, 2020. 1, 2

[11] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *Advances in neural information processing systems*, 30, 2017. 2

[12] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016. 4

[13] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 2

[14] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020. 2, 3, 5

[15] Jian Liang, Dapeng Hu, Yunbo Wang, Ran He, and Jiashi Feng. Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1, 3

[16] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017. 2

[17] Vasista Sai Lodagala, Sreyan Ghosh, and S. Umesh. Pada: Pruning assisted domain adaptation for self-supervised speech representations. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 136–143, 2023. 2

[18] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. *CoRR*, abs/1711.05769, 2017. 2

[19] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. In *ECCV*, 2020. 2

[20] Harsh Rangwani, Sumukh K Aithal, Mayank Mishra, Arihant Jain, and Venkatesh Babu Radhakrishnan. A closer look at smoothness in domain adversarial training. In *International Conference on Machine Learning*, pages 18378–18399. PMLR, 2022. 2

[21] Mohammad Rostami. Lifelong domain adaptation via consolidated internal distribution. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11172–11183. Curran Associates, Inc., 2021. 1, 2

[22] Yang Sui, Miao Yin, Yi Xie, Huy Phan, Saman Aliari Zonouz, and Bo Yuan. Chip: Channel independence-based pruning for compact neural networks. *Advances in Neural Information Processing Systems*, 34:24604–24616, 2021. 2

[23] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 443–450. Springer, 2016. 1

[24] Marco Toldo, Umberto Michieli, and Pietro Zanuttigh. Unsupervised domain adaptation in semantic segmentation via orthogonal and clustered embeddings. In *WACV*, 2021. 2

[25] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017. 1

[26] 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. 5

[27] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022. 1

[28] Tongkun Xu, Weihua Chen, Pichao Wang, Fan Wang, Hao Li, and Rong Jin. CDTrans: Cross-domain transformer for unsupervised domain adaptation. In *ICLR*, 2022. 2

[29] Yinsong Xu, Zhuqing Jiang, Aidong Men, Yang Liu, and Qingchao Chen. Delving into the continuous domain adaptation. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 6039–6049, 2022. 2

[30] Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. In *NeurIPS*, 2021. 1, 2, 3

[31] Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Generalized source-free domain adaptation. *CoRR*, abs/2108.01614, 2021. 2, 5

[32] Qiming Zhang, Jing Zhang, Wei Liu, and Dacheng Tao. Category anchor-guided unsupervised domain adaptation for semantic segmentation. In *NeurIPS*, 2019. 2

[33] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017. 2