TWOFER: TACKLING CONTINUAL DOMAIN SHIFT WITH SIMULTANEOUS DOMAIN GENERALIZATION AND ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In real-world applications, deep learning models often run in non-stationary environments where the target data distribution continually shifts over time. There have been numerous domain adaptation (DA) methods in both online and offline modes to improve cross-domain adaptation ability. However, these DA methods typically only provide good performance *after* a long period of adaptation, and perform poorly on new domains before and during adaptation, especially when domain shifts happen suddenly and momentarily. On the other hand, domain generalization (DG) methods have been proposed to improve the model generalization ability on unadapted domains. However, existing DG works are ineffective for continually changing domains due to severe catastrophic forgetting of learned knowledge. To overcome these limitations of DA or DG in tackling continual domain shifts, we propose `Twofer`, a framework that simultaneously achieves target domain generalization (TDG), target domain adaptation (TDA), and forgetting alleviation (FA). `Twofer` includes a training-free data augmentation module to prepare data for TDG, a novel pseudo-labeling mechanism to provide reliable supervision for TDA, and a prototype contrastive alignment algorithm to align different domains for achieving TDG, TDA and FA. Extensive experiments on Digits, PACS, and Domain Net datasets demonstrate that `Twofer` substantially outperforms state-of-the-art works in Continual DA, Source-Free DA, Test-Time/Online DA, Single DG, Multiple DG and Unified DA&DG. We envision this work as a significant milestone in tackling continual data domain shifts, with improved performance across target domain generalization, adaptation, and forgetting alleviation abilities.

1 INTRODUCTION

A major concern in applying deep learning models to real-world applications is whether they are able to deal with environment changes over time, which present significant challenges with data distribution shift. When the shift is small, deep learning models may be able to handle it because their robustness is often evaluated and improved before deployment. However, when the data distribution shifts significantly, model performance on new scenarios could deteriorate to a much lower level. For example, surveillance cameras used for environmental monitoring can work normally with excellent performance on clear days, but have inferior performance or even become 'blind' when the weather turns bad or the lighting conditions become poor (Bak et al., 2018). As another example, consider conducting lung imaging analysis for corona-viruses, deep learning models may present excellent performance after being trained on a large number of samples for certain variant (e.g., the Alpha variant of COVID-19), but are difficult to provide accurate and timely analysis for later variants (e.g., the Delta or Omicron variant) and future types of corona-viruses (Singh et al., 2020) when they just appear. Addressing such continual data shifts is very challenging. In the following, we will first discuss some of the related works in domain adaptation and generalization, highlight their limitations, and then introduce our approach.

Domain adaptation (DA) methods have been proposed to tackle continual data drifts in dynamic environment in either online or offline mode. For example, Continual DA (Liu et al., 2020; Rostami, 2021) starts from a labeled source domain and continually adapts the model to various target domains, while keeping the model performance from degrading significantly on seen domains.

However, existing Continual DA works often assume that the source domain can be accessed at all time, which may be difficult to guarantee in practical scenarios, especially considering the possible limitation on memory storage and regulations on privacy or intellectual property. Source-Free DA (Yang et al., 2021; Qu et al., 2022) can overcome this issue and achieve target adaptation without the source domain data. In addition, Test-Time or Online DA (Wang et al., 2022; Iwasawa & Matsuo, 2021; Panagiotakopoulos et al., 2022) can improve the target model performance with a small training cost; however, the target domain data is only learned once by the model and the performance improvement is limited (higher improvement would require a large amount of data). With these DA methods, although the model may perform better on the new target domain *after* sufficient adaptation, its performance on the target domain *before and during* the adaptation process, which we call the *'unfamiliar period'*, is often poor. In cases where the domain shift is sudden and the duration of seeing new target domain is short, this problem becomes even more severe. In this work, we believe that for many applications, it is very important to ensure that the model can also perform reasonably well in such *'unfamiliar period'* (i.e., before seeing a lot of target domain data). For instance in environment surveillance, having poor performance under uncommon/unfamiliar weather or lighting conditions may cause significant security and safety risks. In the example of lung imaging analysis for corona-viruses, being able to quickly provide good performance for detecting new variant is critical for the early containment and treatment of the disease.

Domain generalization (DG) methods also solve the learning problem on multiple data domains, especially for cases where the target domain is unavailable or unknown during training. However, existing DG works are typically based on accurate supervision knowledge of the source domain data, whether it is drawn from a single domain (Wang et al., 2021; Li et al., 2021) or multiple domains (Yao et al., 2022; Zhang et al., 2022), which may not be achievable in continually changing scenarios. Moreover, when DG is applied in scenarios with continual domain shifts, as it focuses more on the target domain, there could be severe catastrophic forgetting on domains that have been learned. There are also some works unifying DA and DG (Ghifary et al., 2016; Motiian et al., 2017; Jin et al., 2021); however they can only be used in standard DA or DG individually, thus still suffering their limitations. Bai et al. (2022) and Nasery et al. (2021) study the smooth temporal shifts of data distribution, but they cannot handle large domain shifts over time.

Our Approach and Contribution. In this work, we initiate the study of Continual Domain Shift Learning (CDSL) problem. In CDSL, we assume that the learning model is first trained on a labeled source domain and then face a series of unlabeled target domains that appear continually. The model can be trained on these target domains as each domain is assumed to last for a period of time called a training stage. The goal is to achieve reasonably good performance before and during the training stage of each previously-unseen target domain (addressing the limitations of DA) and also prevent catastrophic forgetting of these domains after they are trained (addressing the limitations of DG).

To solve CDSL, we propose a framework called `Twofer` that provides the capabilities of both DG and DA by optimizing three objectives: (1) to improve the model generalization performance on a new target domain *before and during* its training, which is called *target domain generalization* (TDG), (2) to provide good model performance on a target domain *right after* its training, called *target domain adaptation* (TDA), and (3) to maintain good performance on a trained domain *after* the model is trained with other domains, called *forgetting alleviation* (FA). For improving TDG, `Twofer` includes a training-free data augmentation module that is based on Random Mixup, and this module can generate data outside of the current target domain. For TDA, `Twofer` includes a Top^2 Pseudo Labeling mechanism that lays more emphasis on samples with higher distinguishability, which can produce more accurate pseudo labels. Finally, for optimizing the model towards TDG, TDA, and FA at the same time, `Twofer` includes a Prototype Contrastive Alignment algorithm that is inspired by prototype learning (Saito et al., 2019). Extensive experiments and comprehensive ablation studies on Digits, PACS and Domain Net datasets demonstrate that `Twofer` can substantially outperform state-of-the-art works from Continual DA, Source-Free DA, Test-Time/Online DA, Single DG, Multiple DG, and Unified DA&DG, on objectives of TDG and FA. `Twofer` can also produce comparable state-of-the-art TDA performance as these baselines.

The major contributions of our work can be summarized as:

- We present the first work to consider a practical and important problem, namely Continual Domain Shift Learning, with the goal to achieve three objectives (i.e., TDG, TDA, and FA) for continually arriving domains with data distribution shift.

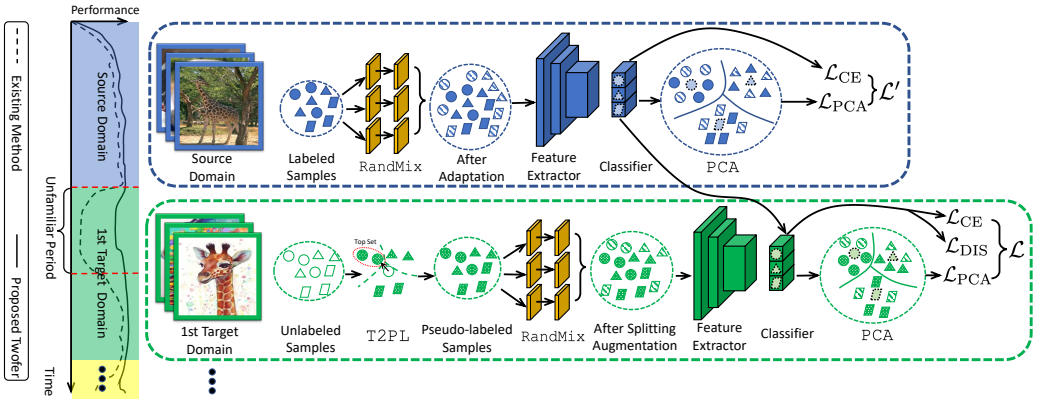


Figure 1: **Overview of applying our framework `Twofer` for Continual Domain Shift Learning (CDSL).** `Twofer` first starts with a labeled source domain, applies `RandMix` on the full set of source data to generate augmentation data, and uses a simplified version $\mathcal{L}'$ of `PCA` for model optimization. Then, for continually arriving target domains, `Twofer` uses `T2PL` to generate pseudo labels for all unlabeled samples, applies `RandMix` on a top subset of these samples based on their softmax confidence, and optimizes the model by `PCA`.

- We propose a novel framework called `Twofer` to solve CDSL. `Twofer` includes a training-free data augmentation module that generates more data for improving TDG, a new pseudo labeling mechanism to provide more accurate labels for improving TDA, and a prototype contrastive alignment algorithm that effectively aligns domains and simultaneously improves TDG, TDA, and FA.
- We conducted extensive experiments and comprehensive ablation studies that demonstrate the advantages of our `Twofer` framework over a number of state-of-the-art baseline methods.

2 Methodology

We first formulate the problem of Continual Domain Shift Learning (CDSL) in Section 2.1, and then introduce the three major modules of our framework `Twofer` for solving CDSL. Specifically, Section 2.2 presents a Random Mixup data augmentation module `RandMix` that generates data for improving the target domain generalization (TDG) on unadapted domains. Section 2.3 presents a Top^2 Pseudo Labeling approach `T2PL` that provides accurate labels for achieving target domain adaptation (TDA) for continually arriving target domains. Finally, Section 2.4 presents a Prototype Contrastive Alignment algorithm `PCA` to optimize the model for achieving TDG, TDA, and forgetting alleviation (FA) on seen domains together. Figure 1 shows the overall pipeline of `Twofer`.

2.1 Problem Formulation of Continual Domain Shift Learning

We assume that there is a labeled source domain $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \| \boldsymbol{x}_i \sim \mathcal{P}_X^{\mathcal{S}}, \boldsymbol{y}_i \sim \mathcal{P}_Y^{\mathcal{S}}\}_{i=1}^{N_{\mathcal{S}}}$ at the beginning of CDSL. Here $\mathcal{P}_X$ and $\mathcal{P}_Y$ are the input and label distributions, respectively, while $N_{\mathcal{S}}$ is the sample quantity. This paper chooses visual recognition as the learning task, in which the number of data classes is K. Then, we consider a sequence of continually arriving target domains $\mathbb{T} = \{\mathcal{T}^t\}_{t=1}^{T}$, where T denotes the total number of domains. The t-th domain contains $N_{\mathcal{T}^t}$ data samples $\boldsymbol{x}$ but no labels $\boldsymbol{y}$, *i.e.*, $\mathcal{T}^t = \{\boldsymbol{x}_i \| \boldsymbol{x}_i \sim \mathcal{P}_X^{\mathcal{T}^t}\}_{i=1}^{N_{\mathcal{T}^t}}$, and all the target domains share the same label space with the source domain. Note that the domain order is randomly determined (including the source domain), which means that the superscript t does not always indicate the same domain.

The generalization performance for the t-th domain $\mathcal{T}^t$ depends on both the previously seen $t-1$ target domains and the original source domain, *i.e.*, $\mathbb{S} = \{\mathcal{S}, \cup_{j=1}^{t-1} \mathcal{T}^j\}$. Considering the possible limitation on memory storage and regulations on privacy or intellectual property, like regular continual learning (Rebuffi et al., 2017), we also set an exemplar memory $\mathcal{M}$ to store $\frac{|\mathcal{M}|}{t}$ exemplars that are closest to the class centroids for each domain in $\mathbb{S}$, where $\frac{|\mathcal{M}|}{t} \ll N_{\mathcal{T}}$, $\frac{|\mathcal{M}|}{t} \ll N_{\mathcal{S}}$. We assume that the model is a deep neural network, and without loss of generality, the neural network consists of a feature extractor Θ at the bottom and a classifier Ω at the top. For each target domain $\mathcal{T}_t$, in addition to the current model $\Omega^t \circ \Theta^t$, its inherited version $\Omega^{t-1} \circ \Theta^{t-1}$ from the last domain is also stored for later usage. To solve CDSL, we aim to continually generalize a model $\Omega \circ \Theta$ to new unlabeled domains $\mathbb{T}$, starting from a labeled source domain $\mathcal{S}$. The objective lies in three aspects: improving the

generalization performance on a new target domain before and during its training (TDG), providing good adaptation performance right after its training (TDA), and preventing catastrophic forgetting of it after the model is trained with other domains (FA).

2.2 RANDOM MIXUP AUGMENTATION

At the beginning of CDSL, only a labeled source domain is available. Thus CDSL faces a pure singe-domain generalization problem where the model is trained on a single domain $\mathcal{S}$ and tested on the other unseen domains $\mathbb{T}$ (each domain in $\mathbb{T}$ is possible as the domain order is random). We apply data augmentation and design a Random Mixup (`RandMix`) method to solve this problem. `RandMix` is not only helpful to improve cross-domain transferability from $\mathcal{S}$ to $\mathcal{T}^1$, but also beneficial for remaining model's order agnostic generalization when it encounters low quality domains. `RandMix` relies on N_{aug} simple autoencoders $\mathbb{G} = \{\Gamma_i\}_{i=1}^{N_{\text{aug}}}$, where each autoencoder consists of an encoder ξ and a decoder ζ. We want `RandMix` to be as simple as possible, preferably to work without training. Inspired by (Wang et al., 2021), the encoder ξ and the decoder ζ are implemented as a convolutional layer and a transposed convolutional layer. With such implementation, even if the parameters of ξ and ζ are randomly initialized from a normalized distribution, the autoencoder can still generate reasonable augmentation data. In order to introduce more randomness, we apply AdaIN (Karras et al., 2019) to inject noise to the autoencoder. Specifically, the used AdaIN contains two linear layers ϕ_1, ϕ_2, and when ϕ_1 and ϕ_2 are fed with a certain noisy input drawn from a normalized distribution ($n \sim \mathcal{P}_{\text{N}(0,1)}$), they can produce two corresponding noisy outputs with smaller variances. As observed in our experiments (Section 3.1), blindly pushing augmentation data away from the original (Li et al., 2021; Wang et al., 2021) possibly hurts the model generalization ability, and injecting randomness with a smaller variance is a better solution. The two noisy outputs are injected to the representations of the autoencoder as multiplicative and additive noises:

$$\Gamma(\boldsymbol{x}) = \zeta\left(\phi_1(n) \times \text{IN}(\xi(\boldsymbol{x})) + \phi_2(n)\right), \tag{1}$$

where $\text{IN}(\cdot)$ represents element-wise normalization operation. `RandMix` works as feeding training data to all autoencoders and mixing the outputs with random weights drawn from a normalized distribution ($\boldsymbol{m} \sim \mathcal{P}_{\text{N}(0,1)}$). Finally, the mixture is scaled by a sigmoid function $\sigma(x) = 1/(1+e^{-x})$:

$$\mathbb{G}(\boldsymbol{x}) = \sigma\left(\frac{1}{\sum_{i=0}^{N_{\text{aug}}} m_i}\left[m_0\boldsymbol{x} + \sum_{i=1}^{N_{\text{aug}}}\left(m_i\Gamma_i(\boldsymbol{x})\right)\right]\right). \tag{2}$$

In mini-batch training, every time there is a new data batch, autoencoders (ξ, ζ), AdaIN-s (ϕ_1, ϕ_2), AdaIN noisy inputs (n) and mixup weights ($\boldsymbol{m}$) are all required to be initialized again. With `RandMix`, we can generate augmentation data with the same labels corresponding to all labeled samples from the source domain. Then, these labeled augmentation samples will work together with the original source data and be fed into the model for training. However, for the following continually arriving target domains, conducting `RandMix` on all target data is unreasonable. In CDSL, all target domains $\mathbb{T} = \{\mathcal{T}^t\}_{t=1}^{T}$ are unlabeled. While we can apply approaches to produce pseudo labels, the supervision is likely unreliable and inaccurate. Therefore, to avoid error propagation and accumulation, we augment a subset of the target data rather than the full set. Specifically, we determine whether a target sample is supposed to be augmented based on its prediction confidence:

$$\tilde{\boldsymbol{x}} = \begin{cases} \mathbb{G}(\boldsymbol{x}), & \text{if} \max\left[\Omega(\Theta(\boldsymbol{x}))\right]_K \geq p_{\text{th}} \\ \emptyset, & \text{otherwise} \end{cases}, \boldsymbol{x} \sim \mathcal{P}_X^{\mathcal{T}}, \tag{3}$$

where $\max[\cdot]_K$ denotes the maximum of a vector with K dimensions, and p_{th} is a confidence threshold that is set to 0.8 in our implementation (sensitivity analysis of p_{th} is provided in Appendix).

2.3 TOP^2 PSEUDO LABELING

In CDSL, all target domains arrive only with the data but no label. Thus we need a pseudo labeling mechanism to provide reasonably accurate supervision for subsequent optimizations. However, existing pseudo labeling approaches have various limitations. For example, softmax-based pseudo labeling (Lee et al., 2013) produces hard labels for unlabeled samples, but training with such hard labels sacrifices inter-class distinguishability within the softmax predictions. To preserve inter-class distinguishability, SHOT (Liang et al., 2020) proposes a clustering-based approach to align unlabeled samples with cluster centroids, but treats all samples equally during clustering. In this case,

samples with high distinguishability are not used well and those with low distinguishability pose a negative impact to cluster construction. To address such issues, we propose a novel mechanism called Top^2 Pseudo Labeling (T2PL). Specifically, we use the softmax confidence to measure the distinguishability of data samples, and select the top 50% set to construct class centroids:

$$\mathcal{I}_k = \cup^{\frac{N_{\mathcal{T}^t}}{p_{\text{top}} \cdot K}} \left\{ \arg\max_{\boldsymbol{x} \in \mathcal{T}^t} \left[\Omega_k(\Theta(\boldsymbol{x})) \right] \right\}, \mathcal{F} = \cup_{k=1}^{K} \left\{ \cup_{i \in \mathcal{I}_k, \boldsymbol{x} \in \mathcal{T}^t} \{\boldsymbol{x}_i\} \right\}, \tag{4}$$

where $\Omega_k(\cdot)$ denotes the k-th element of the classifier outputs. p_{top} controls the size of the selected top set, and if the data is class balanced, top 50% corresponds to $p_{\text{top}} = 2$. Then the top set $\mathcal{F}$ is used to construct class centroids by a prediction-weighted aggregation on representations:

$$c_k = \frac{\sum_{\boldsymbol{x}_i \in \mathcal{F}} \Omega_k \left(\Theta(\boldsymbol{x}_i) \right) \cdot \Theta(\boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_i \in \mathcal{F}} \Omega_k \left(\Theta(\boldsymbol{x}_i) \right)}. \tag{5}$$

Subsequently, we can compute the cosine similarity between these centroids and representations of all current unlabeled data, and also select the top half set of samples that have much higher similarity to the centroids than the rest for each class:

$$\mathcal{I}'_k = \cup^{\frac{N_{\mathcal{T}^t}}{p_{\text{top}} \cdot K}} \left\{ \arg\max_{\boldsymbol{x} \in \mathcal{T}^t} \left[\frac{\Theta(\boldsymbol{x}) \cdot c_k^\top}{\|\Theta(\boldsymbol{x})\| \|c_k\|} \right] \right\}, \mathcal{F}' = \cup_{k=1}^{K} \left\{ \cup_{i \in \mathcal{I}'_k, \boldsymbol{x} \in \mathcal{T}^t} \{(\boldsymbol{x}_i, k)\} \right\}. \tag{6}$$

Then we use $\mathcal{F}'$ to fit a k-Nearest Neighbor (kNN) classifier and assign the pseudo label as follows:

$$\hat{\boldsymbol{y}} = \text{kNN}\left(\boldsymbol{x}, \mathcal{F}'\right)^{\frac{N_{\mathcal{T}^t}}{p'_{\text{top}} \cdot K}}_{\text{Euclidean}}, \tag{7}$$

where the superscript $\frac{N_{\mathcal{T}^t}}{p'_{\text{top}} \cdot K}$ and subscript Euclidean mean that the kNN works to find $\frac{N_{\mathcal{T}^t}}{p'_{\text{top}} \cdot K}$ closest neighbors from $\mathcal{F}'$ with the measurement of Euclidean distance. We select the top 5% for each class ($p'_{\text{top}} = 20$) in our implementation (sensitivity analyses of p_{top} and p'_{top} are provided in the Appendix). With this kNN, T2PL makes better use of unlabeled samples with relatively high distinguishability than SHOT, and provides more accurate pseudo labels.

2.4 Prototype Contrastive Alignment

Prototype learning (PL) (Pan et al., 2019; Kang et al., 2019; Tanwisuth et al., 2021; Dubey et al., 2021) has been demonstrated to be effective for solving cross-domain sample variability and sample insufficiency. These two issues are reflected in our problem as the random uncertainty of augmentation data and limited data quantity of seen domains in the exemplar memory $\mathcal{M}$. As a result, in this work, we adopt the idea of PL and propose a new Prototype Contrastive Alignment (PCA) algorithm to align domains together for improving model generalization ability.

First, we need to point out that regular PL is unsuitable to our problem. In regular PL, all samples are fed into the model to extract representations, and these representations are aggregated class-by-class for constructing prototypes $w^t = \{w_k^t\}_{k=1}^K$ of a certain domain $\mathcal{T}^t$, and then the model is optimized with an alignment loss:

$$w_k^t = \frac{\sum_{(\boldsymbol{x}_i, \hat{\boldsymbol{y}}_i) \in \mathcal{T}^t} \mathbb{I}_{\hat{\boldsymbol{y}}_i = k} \Theta\left(\boldsymbol{x}_i\right)}{\sum_{(\boldsymbol{x}_i, \hat{\boldsymbol{y}}_i) \in \mathcal{T}^t} \mathbb{I}_{\hat{\boldsymbol{y}}_i = k}}, \mathcal{L}_{\text{PL}} = \mathbb{E}_{\boldsymbol{x}_i \sim \mathcal{P}_X^{\mathcal{T}^t}} \left[\sum_{k=1}^{K} -\log \frac{\mathbb{I}_{\hat{\boldsymbol{y}}_i = k} \exp\left(w_k^{t\top} \Theta(\boldsymbol{x}_i) \right)}{\sum_{c=1}^{K} \exp\left(w_c^{t\top} \Theta(\boldsymbol{x}_i) \right)} \right]. \tag{8}$$

$\mathbb{I}_{(\cdot)}$ is an indicator function that if the subscript condition is true, $\mathbb{I}_{\text{True}} = 1$, otherwise, $\mathbb{I}_{\text{False}} = 0$. To achieve cross-domain alignment, prototypes of different domains for the same classes are aggregated and averaged, and the alignment loss is computed on these average prototypes for training. However, such domain alignment has a problem of adaptivity gap (Dubey et al., 2021). As shown in Figure 2, we assume that there is an optimal prototype location w_k^* of class k for all domains ($\mathcal{S} \cup \mathbb{T}$), and the objective is to move the source prototype w_k^s to the optimal by training on target domains ($\mathbb{T}$) one-by-one. Meanwhile, we also regard that there are optimal prototypes w_k^t for each domain itself, and suppose that the regular PL is applied to align prototypes of all these domains. To illustrate the problem of adaptivity gap, we take the alignment between the source $\mathcal{S}$ and the first target domain $\mathcal{T}^1$ as an example, and assume that the distance between w_k^1 and w_k^* is larger than the distance between w_k^s and w_k^*. In this case, after the domain alignment, the location of the current prototype is worse than the source prototype, which means that the adaptivity gap is enlarged. Although we

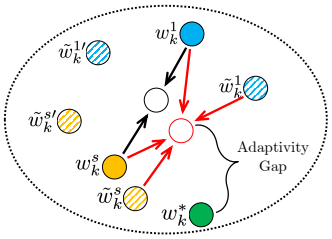


Figure 2: When the model encounters a domain whose prototypes (w_k^1) are far from the optimal ones (w_k^*), domain alignment with regular prototype learning will enlarge the adaptivity gap. Although `RandMix` ($\tilde{w}_k^s, \tilde{w}_k^{s\prime}, \tilde{w}_k^1, \tilde{w}_k^{1\prime}$) can help reduce the gap, regular prototype construction cannot take advantage of it.

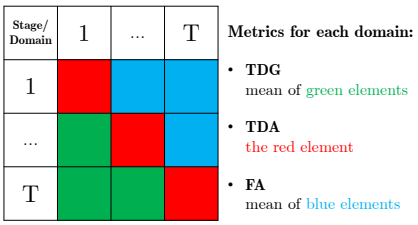


Figure 3: Three objectives/metrics of `Twofer`: Target Domain Generalization (TDG), Target Domain Adaptation (TDA), and Forgetting Alleviation (FA).

can use `RandMix` to generate more data ($\tilde{w}_k^s$, $\tilde{w}_k^1$) that might be helpful for reducing the adaptivity gap, there is randomness for each use of `RandMix` and we cannot guarantee the benefit of reducing gap every time (*e.g.*, what if the augmentation data is $\tilde{w}_k^{s\prime}$ and $\tilde{w}_k^{1\prime}$ in Figure 2). Therefore, a better prototype construction strategy is needed to provide more representative prototypes.

Inspired by a semi-supervised learning work (Saito et al., 2019), we view neuron weights of the classifier Ω as the prototypes. The classifier Ω consists of a single linear layer and is placed exactly behind the feature extractor Θ. Thus the dimension of its neuron weights is the same as the hidden dimension of extracted representations. We construct such prototypes with a CrossEntropy Loss:

$$\mathcal{L}_{\text{CE}} = \mathbb{E}_{\boldsymbol{x}_i \sim \mathcal{P}_X^{\mathcal{T}^t}} \left[\sum_{k=1}^{K} -\log \frac{\mathbb{I}_{\hat{\boldsymbol{y}}_i=k} \exp\left(\boldsymbol{w}_k^{t\top} \Theta(\boldsymbol{x}_i) + b_k\right)}{\sum_{c=1}^{K} \exp\left(\boldsymbol{w}_c^{t\top} \Theta(\boldsymbol{x}_i) + b_c\right)} \right], \tag{9}$$

where b_k, b_c are biases of the linear layer, and we set them as zeros during training. Compared with regular prototypes, such linear layer prototypes are built on a lot more data because `RandMix` is initialized and used for every mini-batch during model training. Moreover, `RandMix` has general effect on improving generalization ability, as shown in our experiments later in Section 3.2. Thus we believe that linear layer prototypes have smaller adaptivity gaps than regular ones.

With linear layer prototypes, we can align domains in a better way. Unlike the alignment on regular prototypes, we do not sum up and average prototypes of different domains. Instead, we simultaneously maximize the similarities of data samples to prototypes of different domains, although in practice we may have prototypes of just two domains since there is only one old model being stored (Section 2.1). We also introduce contrastive comparison into the domain alignment process that can enhance the distinguishability of different classes and be beneficial for our pseudo labeling. The contrastive comparison includes negative pairs of a particular sample and samples from other classes. In this case, our prototype contrastive alignment is formulated as follows:

$$\begin{aligned}
&\mathcal{L}_{\text{PCA}} = \mathbb{E}_{\boldsymbol{x}_i \sim \mathcal{P}_X^{\mathcal{T}^t}} \left\{ \sum_{k=1}^{K} -\log \frac{\mathbb{I}_{\hat{\boldsymbol{y}}_i=k} \left[\exp\left(\boldsymbol{w}_k^{t\top} \Theta(\boldsymbol{x}_i)\right) + \exp\left(\boldsymbol{w}_k^{t-1\top} \Theta(\boldsymbol{x}_i)\right)\right]}{\Delta} \right\}, \text{where} \\
&\Delta = \sum_{c=1}^{K} \exp\left(\boldsymbol{w}_c^{t\top} \Theta(\boldsymbol{x}_i)\right) + \sum_{c=1}^{K} \exp\left(\boldsymbol{w}_c^{t-1\top} \Theta(\boldsymbol{x}_i)\right) + \sum_{\boldsymbol{x}_j \in \mathcal{T}^t, j \neq i} \mathbb{I}_{\hat{\boldsymbol{y}}_i \neq \hat{\boldsymbol{y}}_j} \exp\left(\Theta(\boldsymbol{x}_i)^\top \Theta(\boldsymbol{x}_j)\right).
\end{aligned} \tag{10}$$

Furthermore, we also adopt knowledge distillation from the stored old model to the current model for forgetting compensation. In this case, our final optimization objective is shown as follows:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{PCA}} + \mathcal{L}_{\text{DIS}}, \text{where } \mathcal{L}_{\text{DIS}} = \mathcal{D}_{\text{KL}} \left[\Omega^{t-1}\left(\Theta^{t-1}(\boldsymbol{x})\right) \| \Omega^t\left(\Theta^t(\boldsymbol{x})\right) \right]_{\boldsymbol{x} \sim \mathcal{P}_X^{\mathcal{T}^t}}. \tag{11}$$

Note that this optimization objective is used in all target domains. As for the source domain, there is no distillation $\mathcal{L}_{\text{DIS}}$ and the nominator of $\mathcal{L}_{\text{PCA}}$ only contains the first term, because there is no stored old model in the stage of source training. We denote this simplified optimization loss as $\mathcal{L}'$.

3 EXPERIMENTS

Our code is implemented in PyTorch (and provided in the Supplementary Materials). All experiments are conducted on a server running Ubuntu 18.04 LTS, equipped with NVIDIA RTX A5000 GPUs. The datasets, experiment settings and comparison baselines are introduced below.

Datasets. Digits consists of 5 different domains with 10 classes, including MNIST (MT), SVHN (SN), MNIST-M (MM), SYN-D (SD) and USPS (US). There are 266,504 images in total, and each one is shaped with the size 32×32. **PACS** contains 4 domains with 7 classes, including Photo (P), Art painting (A), Cartoon (C) and Sketch (S). Each image is shaped with the size 224×224, and there are 9,991 images in total. **Domain Net** is the most challenging cross-domain dataset, including Quickdraw (Qu), Clipart (Cl), Painting (Pa), Infograph (In), Sketch (Sk) and Real (Re). 26,013 images are selected (to reduce class imbalance; more in the Appendix), with the size 224×224.

Experiment Settings. We apply ResNet-50 as the feature extractor for both PACS and Domain Net, and apply DTN as the feature extractor for Digits (Liang et al., 2020). Unlike regular domain adaptation works that use training data for testing, we split 80% data as the training set and the rest 20% as the testing set. The SGD optimizer with an initial learning rate of 0.01 is used for Digits, and 0.005 for PACS and Domain Net. The exemplar memory size is set as 200 for all datasets, and the batch size is 64. For all experiments, we conduct multiple runs with three seeds (2022, 2023, 2024), and report the average performance.

Comparison Baselines. To our best knowledge, there is no existing work that targets the exact problem of CDSL. Therefore, to better demonstrate the effectiveness of our work, we compare `Twofer` with a comprehensive set of state-of-the-art works from Continual DA [CoT (Wang et al., 2022), AuC (Rostami, 2021)], Source-Free DA [SHO (Liang et al., 2020), GSF (Yang et al., 2021), BMD (Qu et al., 2022)], Test-Time/Online DA [TEN (Wang et al., 2020), T3A (Iwasawa & Matsuo, 2021)], Single DG [L2D (Wang et al., 2021), PDE (Li et al., 2021)], Unified DA&DG [SNR (Jin et al., 2021)], and Multiple DG [PCL (Yao et al., 2022), EFD (Zhang et al., 2022)].

3.1 EFFECTIVENESS OF THE TWOFER FRAMEWORK

We assume that each domain corresponds to a single training stage. After the training of every stage, we test the model performance on all domains. In this case, we can obtain an accuracy matrix in which each row denotes the test performance of a particular domain at different stages, while each column represents the test accuracy of different domains at a certain stage, as shown in Figure 3. We design three metrics that measure TDG, TDA, and FA, respectively. TDG is measured by the mean model performance on a domain *before* its training stage (mean of the green elements in Figure 3), TDA by the performance on a domain *right after* finishing its training stage (the red element), and FA by the mean performance on a domain *after* the model has been updated with other domains (mean of the blue elements). All these metrics are the higher, the better. Due to space limitation, we report experiment results for Digits (Tables 1) and PACS (Table 2) in two domain orders, and for Domain Net (Table 3) in one domain order. Results for more domain orders can be found in the Appendix. We can observe that for all three datasets, **`Twofer` achieves much higher performance than all baselines across all three metrics**. Specifically, `Twofer` substantially outperforms the second-best baseline on TDG in all cases and by 3.1∼10.7% in average, and the performance improvement is particularly large over DA methods. For TDA and FA, `Twofer` is also the best in most cases and clearly outperforms the second best; and even if not, very close to the best. These results demonstrate that our method can effectively improve the model performance during the '*unfamiliar period*', adapt the model to different domains, and alleviate catastrophic forgetting on seen domains.

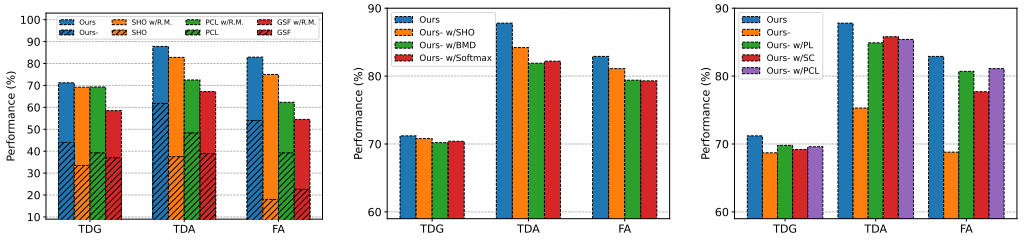
3.2 ABLATION STUDIES

To demonstrate the effectiveness of each module in `Twofer`, we conduct comprehensive ablation studies for `RandMix`, `T2PL`, and `PCA` on Digits with the domain order US→SD→SN→MM→MT.

Effectiveness of `RandMix`. To evaluate the effectiveness of `RandMix`, we first remove it from `Twofer` and see how the model performance will change. We also try to apply `RandMix` on a few other baselines and explore if there will be a performance improvement, including SHO (Liang et al., 2020), GSF (Yang et al., 2021), and PCL (Yao et al., 2022). The experiment results in Figure 4(a) demonstrate that, removing `RandMix` from `Twofer` hurts the performance significantly,

Table 1: Performance comparisons between ours and other methods on Digits in TDG, TDA, and FA under two domain orders (shown with ↓). We blue and bold **the best**, and bold **the second best**.

Order	Metric		CoT	AuC	SHO	GSF	BMD	TEN	T3A	L2D	PDE	SNR	PCL	EFD	Ours
MT ↓ MM ↓ SN ↓ SD ↓ US	TDG	MM	52.3	44.9	46.7	55.2	46.7	52.3	52.3	**73.1**	66.6	70.1	51.0	50.5	**86.1**
		SN	24.1	15.8	19.6	28.3	18.6	30.6	26.8	**36.0**	35.4	35.7	23.1	12.1	**45.7**
		SD	34.3	29.3	30.2	30.7	28.4	41.7	34.6	**56.3**	56.0	56.1	35.9	16.3	**67.7**
		US	**90.0**	83.6	49.2	50.5	47.5	84.9	71.0	84.8	83.2	81.7	73.3	35.7	**92.2**
		Avg.	50.2	43.4	35.3	41.2	35.3	52.4	46.2	**62.6**	60.3	60.9	45.8	28.7	**72.9**
	TDA	MT	99.0	99.1	**99.3**	**99.4**	99.2	99.0	99.0	**99.3**	99.1	99.2	99.2	99.2	**99.4**
		MM	41.7	41.4	75.7	54.3	75.9	57.5	51.5	**87.4**	84.4	85.0	55.0	14.0	**90.3**
		SN	17.8	18.5	9.7	13.7	9.9	29.4	25.2	**55.3**	50.0	51.1	22.6	9.5	**70.6**
		SD	27.8	30.1	10.6	13.1	9.3	40.2	32.4	**59.9**	50.9	45.2	28.0	11.4	**82.7**
		US	89.9	79.5	23.0	14.8	16.5	90.5	74.5	**91.6**	90.0	91.1	59.2	16.1	**94.9**
		Avg.	55.2	53.7	43.7	39.1	42.2	63.3	56.5	**78.7**	74.9	74.3	52.8	35.8	**87.6**
	FA	MT	93.9	92.6	36.2	33.9	36.0	**98.9**	86.0	92.4	91.5	92.0	83.2	34.4	**96.1**
		MM	33.1	48.5	12.7	10.5	11.8	55.2	44.9	70.7	69.8	**71.1**	38.2	16.2	**82.0**
		SN	17.3	20.4	9.6	9.0	9.6	29.4	16.8	**56.1**	50.3	55.4	20.8	12.1	**70.3**
		SD	27.3	28.8	12.5	10.4	10.8	40.2	27.7	70.0	69.1	**71.3**	28.6	16.3	**81.9**
		Avg.	42.9	47.6	17.8	16.0	17.1	55.9	43.9	72.3	70.2	**72.5**	42.7	19.8	**82.6**
US ↓ SD ↓ SN ↓ MM ↓ MT	TDG	SD	36.4	33.6	39.6	37.2	39.6	36.4	36.4	**61.7**	60.0	61.5	37.1	36.8	**68.2**
		SN	19.2	19.9	19.4	22.5	19.3	26.4	20.4	**52.6**	51.1	52.5	21.9	14.4	**61.5**
		MM	30.9	33.8	29.5	32.6	29.8	40.4	34.6	**63.6**	60.7	60.5	34.9	19.2	**68.0**
		MT	54.2	63.0	45.4	55.6	47.9	77.2	58.9	**84.0**	79.9	81.1	63.0	29.0	**87.0**
		Avg.	35.2	37.6	33.5	37.0	34.2	45.1	37.6	**65.5**	62.9	63.9	39.2	24.9	**71.2**
	TDA	US	98.5	98.6	98.7	98.3	98.7	98.5	98.5	**99.0**	98.8	**98.9**	98.6	98.3	**98.9**
		SD	22.5	24.8	53.2	40.1	47.9	38.5	29.9	**72.5**	71.0	70.8	37.8	9.9	**82.7**
		SN	14.3	18.7	13.8	17.2	11.0	28.3	20.1	**69.1**	65.6	64.0	20.0	10.0	**78.2**
		MM	25.9	32.8	12.3	19.6	14.3	39.5	34.2	**76.0**	75.5	75.0	29.4	10.3	**83.4**
		MT	44.3	55.0	9.3	19.5	28.7	79.5	67.4	**87.0**	85.4	83.9	55.5	19.6	**95.7**
		Avg.	41.1	46.0	37.5	38.9	40.1	56.9	50.0	**80.7**	79.3	78.5	48.3	29.6	**87.8**
	FA	US	95.3	91.9	37.6	49.7	39.7	**98.4**	76.1	93.6	91.0	92.2	82.1	11.3	**96.3**
		SD	19.9	25.8	13.1	22.1	15.3	37.1	23.5	**72.7**	70.4	69.5	27.6	10.7	**82.0**
		SN	13.7	19.7	11.3	8.7	10.4	27.9	12.3	**56.8**	55.9	56.0	18.2	14.1	**68.2**
		MM	26.1	30.1	10.1	9.8	17.2	40.6	16.4	**74.2**	70.1	71.2	29.0	29.0	**85.1**
		Avg.	38.8	41.9	18.0	22.6	20.7	51.0	32.1	**74.3**	71.9	72.2	39.2	16.3	**82.9**



(a) `RandMix` (R.M.) (b) `T2PL` (c) `PCA`

Figure 4: Ablation studies of `RandMix`, `T2PL`, and `PCA` on Digits. Average performance of all domains for three metrics (TDG, TDA, FA) are shown. 'Ours' here denotes the full framework of `Twofer`, while 'Ours-' represents removing a corresponding module (a, b or c) from `Twofer`, 'Ours- w/' means replacing the corresponding module with a new one.

and attaching `RandMix` to other baseline methods clearly improves the model performance. Such observations prove the effectiveness of `RandMix`.

Effectiveness of `T2PL`. We replace `T2PL` in `Twofer` with other pseudo labeling approaches to investigate the gain from `T2PL`, including Softmax (Lee et al., 2013), SHO (Liang et al., 2020), and BMD (Qu et al., 2022). Figure 4(b) shows that `Twofer` performs the best when it has `T2PL`, proving the effectiveness of `T2PL`.

Effectiveness of `PCA`. We first remove $\mathcal{L}_{\text{PCA}}$ from the optimization objective, leaving only CrossEntropy Loss and Logits Distillation. We then try to replace `PCA` with regular forwarding prototype learning (PL), supervised contrastive learning (SC), and PCL (Yao et al., 2022). Figure 4(c) shows that the performance of `Twofer` degrades significantly without $\mathcal{L}_{\text{PCA}}$, and when we use other methods instead, the performance is still worse. This validates the effectiveness of `PCA`.

Table 2: Performance comparisons between ours and other methods on PACS in TDG, TDA, and FA under two domain orders (shown with ↓). We blue and bold **the best**, and bold **the second best**.

Order	Metric		CoT	AuC	SHO	GSF	BMD	TEN	T3A	L2D	PDE	SNR	PCL	EFD	Ours
P ↓ A ↓ C ↓ S	TDG	A	68.0	66.6	65.9	67.3	65.9	68.0	68.0	67.2	67.2	67.3	**68.1**	67.4	**70.0**
		C	48.6	32.1	42.1	47.6	42.6	48.6	34.8	**49.3**	47.2	43.1	40.5	40.4	**51.4**
		S	45.3	39.3	41.0	46.3	42.5	45.5	38.1	**52.6**	52.2	47.7	41.6	42.2	**57.1**
		Avg.	54.0	46.0	49.7	53.7	50.3	54.0	47.0	**56.4**	55.5	52.7	50.1	50.0	**59.5**
	TDA	P	**99.4**	**99.4**	**99.7**	98.8	**99.7**	**99.4**	**99.4**	99.1	99.2	99.0	98.8	99.1	**99.7**
		A	73.2	64.9	**88.5**	**87.6**	86.8	72.9	68.8	72.4	71.1	72.0	77.1	74.9	87.1
		C	66.1	56.9	56.3	**71.9**	58.2	66.7	61.2	61.4	60.7	62.0	63.1	35.0	**75.3**
		S	51.0	58.1	56.1	**70.6**	68.8	52.3	61.2	68.2	65.5	66.6	64.6	32.8	**70.0**
		Avg.	72.4	69.8	75.2	**82.2**	78.4	72.8	72.7	75.3	74.1	74.9	75.9	60.5	**83.0**
	FA	P	**98.5**	97.1	95.8	70.8	96.2	**98.5**	**98.6**	94.2	94.0	95.5	74.5	81.4	98.1
		A	73.2	73.3	80.9	41.4	**82.1**	73.2	65.5	67.4	66.0	67.3	60.8	60.1	**87.0**
		C	66.1	**67.8**	59.7	32.0	65.5	67.0	59.1	60.1	58.8	61.3	41.6	49.0	**72.5**
		Avg.	79.3	79.4	78.8	48.1	**81.3**	79.6	74.4	73.9	72.9	74.7	59.0	63.5	**85.9**
S ↓ C ↓ A ↓ P	TDG	C	61.6	53.7	44.6	61.0	44.6	61.6	61.6	64.1	63.1	64.0	50.3	**64.7**	**68.4**
		A	**55.1**	41.6	38.9	47.0	47.2	**55.1**	54.5	51.0	50.4	51.3	35.0	32.2	**66.6**
		P	60.7	52.8	71.6	59.7	**73.7**	60.7	67.2	69.8	70.0	71.1	63.7	34.6	**83.3**
		Avg.	59.1	49.4	51.7	55.9	55.2	59.1	61.1	61.6	61.2	61.2	**62.1**	43.8	**72.8**
	TDA	S	95.4	95.5	95.0	**96.2**	95.0	95.4	95.4	**96.3**	96.0	96.1	96.1	95.9	**96.3**
		C	68.2	66.5	78.0	**79.7**	78.7	68.0	76.5	**79.4**	79.0	79.0	75.9	68.0	76.5
		A	61.5	69.3	67.8	**86.8**	83.7	61.5	70.2	69.8	67.9	70.0	56.3	13.7	**89.3**
		P	59.3	74.3	94.3	68.0	**97.0**	59.3	72.5	86.8	85.5	86.7	92.2	13.8	**99.1**
		Avg.	71.1	76.4	83.8	82.7	**88.6**	71.1	78.7	83.1	82.1	83.0	80.1	47.9	**90.3**
	FA	S	**94.0**	71.5	85.2	87.2	86.0	**95.0**	93.9	88.6	88.0	89.3	90.7	81.8	**94.0**
		C	68.2	74.2	78.3	63.5	**79.1**	68.0	69.4	78.0	74.9	78.2	75.0	52.9	**80.9**
		A	61.5	52.8	72.0	81.2	**84.4**	61.7	69.3	71.7	70.4	71.0	60.2	21.2	**87.8**
		Avg.	74.6	66.2	78.5	77.3	**83.2**	74.9	77.5	79.4	77.8	79.5	75.3	52.0	**87.6**

Table 3: Performance comparisons between ours and other methods on Domain Net in TDG, TDA, and FA (domain order shown left with ↓). We blue and bold **the best**, and bold **the second best**.

Order	Metric		CoT	AuC	SHO	GSF	BMD	TEN	T3A	L2D	PDE	SNR	PCL	EFD	Ours
Qu ↓ Sk ↓ Cl ↓ In ↓ Pa ↓ Re	TDG	Sk	28.2	28.6	23.6	29.7	23.6	28.2	28.2	**31.7**	29.8	29.9	27.9	30.3	**32.8**
		Cl	51.9	42.6	30.8	**52.2**	28.5	52.1	49.4	52.1	50.2	51.0	41.4	28.0	**55.0**
		In	17.1	15.7	12.1	13.6	12.0	**17.2**	16.1	13.2	12.7	11.1	13.2	11.5	**17.5**
		Pa	25.4	23.1	13.9	**26.6**	12.5	25.6	20.8	24.1	23.0	24.3	13.9	13.3	**42.9**
		Re	52.8	41.7	34.8	44.4	33.7	**53.2**	49.2	39.6	37.9	37.0	36.7	18.1	**65.8**
		Avg.	35.1	30.3	23.0	33.3	22.1	**35.3**	32.7	32.1	30.7	30.7	26.6	20.2	**42.8**
	TDA	Qu	**92.7**	**92.7**	91.0	**94.8**	91.0	92.0	92.1	91.8	92.0	91.8	91.5	90.9	92.0
		Sk	36.9	31.1	34.2	**45.5**	30.1	37.0	36.7	37.4	36.6	37.0	31.7	12.3	**47.8**
		Cl	59.4	52.9	27.3	58.9	40.1	**60.1**	59.2	55.1	54.1	55.7	49.3	8.7	**66.7**
		In	19.1	18.5	9.4	**20.9**	12.5	**19.7**	19.6	13.3	13.0	15.1	16.0	9.3	18.5
		Pa	**30.8**	30.2	14.2	28.1	11.8	31.2	31.2	16.0	15.0	20.3	10.6	12.6	**50.6**
		Re	57.7	45.1	38.7	44.3	36.4	**58.9**	58.3	41.9	41.1	42.0	33.6	4.2	**78.2**
		Avg.	49.4	45.1	35.8	48.8	37.0	**49.8**	49.5	42.6	42.0	43.7	38.8	23.0	**59.0**
	FA	Qu	**91.4**	84.4	49.6	75.9	54.9	**91.3**	82.3	79.6	77.9	80.0	80.8	64.2	89.3
		Sk	36.9	35.0	29.7	36.2	28.3	**37.3**	36.6	32.5	31.7	32.8	30.3	15.7	**53.8**
		Cl	59.0	54.3	26.1	42.8	38.7	**60.5**	53.5	44.4	43.3	45.0	42.7	20.0	**66.1**
		In	19.3	18.0	13.2	11.9	16.6	19.7	**20.1**	12.9	11.9	12.0	14.3	11.6	**20.3**
		Pa	**31.7**	31.2	13.2	30.8	14.5	31.0	30.2	15.3	17.3	20.1	8.1	7.5	**54.5**
		Avg.	47.7	44.6	26.4	39.5	30.6	**48.0**	44.5	36.9	36.4	38.0	35.2	23.8	**56.8**

4 CONCLUSION

This paper is the first to consider a practical and important problem called Continual Domain Shift Learning (CDSL), and proposes a novel framework `Twofer` that includes a training-free data augmentation module `RandMix`, a pseudo labeling mechanism `T2PL`, and a prototype contrastive alignment training algorithm `PCA`. Extensive experiments demonstrate that `Twofer` can substantially improve the performance across target domain generalization, target domain adaptation and forgetting alleviation over various state-of-the-art methods from Continual DA, Source-Free DA, Test-Time/Online DA, Single DG, Multiple DG, and Unified DA&DG.

ETHICS STATEMENT

In this paper, our studies are not related to human subjects, practices to data set releases, discrimination/bias/fairness concerns, and also do not have legal compliance or research integrity issues. Our work is proposed to address continual domain shifts when applying deep learning models in real-world applications. In this case, if the trained models are used responsibly for good purposes, we believe that our proposed methods will not cause ethics issues or pose negative societal impacts.

REPRODUCIBILITY STATEMENT

The source code is provided in the Supplementary Materials. All datasets we use are public. In addition, we also provide detailed experiment parameters and random seeds in the Appendix.

REFERENCES

Guangji Bai, Ling Chen, and Liang Zhao. Temporal domain generalization with drift-aware dynamic neural network. *arXiv preprint arXiv:2205.10664*, 2022.

Slawomir Bak, Peter Carr, and Jean-Francois Lalonde. Domain adaptation through synthesis for unsupervised person re-identification. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 189–205, 2018.

Abhimanyu Dubey, Vignesh Ramanathan, Alex Pentland, and Dhruv Mahajan. Adaptive methods for real-world domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14340–14349, 2021.

Muhammad Ghifary, David Balduzzi, W Bastiaan Kleijn, and Mengjie Zhang. Scatter component analysis: A unified framework for domain adaptation and domain generalization. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1414–1430, 2016.

Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 34:2427–2440, 2021.

Xin Jin, Cuiling Lan, Wenjun Zeng, and Zhibo Chen. Style normalization and restitution for domain generalization and adaptation. *IEEE Transactions on Multimedia*, 2021.

Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4893–4902, 2019.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 896, 2013.

Lei Li, Ke Gao, Juan Cao, Ziyao Huang, Yepeng Weng, Xiaoyue Mi, Zhengze Yu, Xiaoya Li, and Boyang Xia. Progressive domain expansion network for single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 224–233, 2021.

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*, pp. 6028–6039. PMLR, 2020.

Hong Liu, Mingsheng Long, Jianmin Wang, and Yu Wang. Learning to adapt to evolving domains. *Advances in Neural Information Processing Systems*, 33:22338–22348, 2020.

Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5715–5725, 2017.

Anshul Nasery, Soumyadeep Thakur, Vihari Piratla, Abir De, and Sunita Sarawagi. Training for the future: A simple gradient interpolation loss to generalize along time. *Advances in Neural Information Processing Systems*, 34:19198–19209, 2021.

Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2239–2247, 2019.

Theodoros Panagiotakopoulos, Pier Luigi Dovesi, Linus Härenstam-Nielsen, and Matteo Poggi. Online domain adaptation for semantic segmentation in ever-changing conditions. *arXiv preprint arXiv:2207.10667*, 2022.

Sanqing Qu, Guang Chen, Jing Zhang, Zhijun Li, Wei He, and Dacheng Tao. Bmd: A general class-balanced multicentric dynamic prototype strategy for source-free domain adaptation. *arXiv preprint arXiv:2204.02811*, 2022.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Mohammad Rostami. Lifelong domain adaptation via consolidated internal distribution. *Advances in Neural Information Processing Systems*, 34:11172–11183, 2021.

Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8050–8058, 2019.

Dilbag Singh, Vijay Kumar, Manjit Kaur, et al. Classification of covid-19 patients from chest ct images using multi-objective differential evolution–based convolutional neural networks. *European Journal of Clinical Microbiology & Infectious Diseases*, 39(7):1379–1389, 2020.

Korawat Tanwisuth, Xinjie Fan, Huangjie Zheng, Shujian Zhang, Hao Zhang, Bo Chen, and Mingyuan Zhou. A prototype-oriented framework for unsupervised domain adaptation. *Advances in Neural Information Processing Systems*, 34:17194–17208, 2021.

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.

Jindong Wang and Wang Lu. Deepdg: Deep domain generalization toolkit. `https://github.com/jindongwang/transferlearning/tree/master/code/DeepDG`.

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*, pp. 7201–7211, 2022.

Zijian Wang, Yadan Luo, Ruihong Qiu, Zi Huang, and Mahsa Baktashmotlagh. Learning to diversify for single domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 834–843, 2021.

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*, pp. 8978–8987, 2021.

Xufeng Yao, Yang Bai, Xinyun Zhang, Yuechen Zhang, Qi Sun, Ran Chen, Ruiyu Li, and Bei Yu. Pcl: Proxy-based contrastive learning for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7097–7107, 2022.

Yabin Zhang, Minghan Li, Ruihuang Li, Kui Jia, and Lei Zhang. Exact feature distribution matching for arbitrary style transfer and domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8035–8045, 2022.

SUMMARY OF THE APPENDIX

This Appendix includes additional details for the ICLR 2023 submission *"Twofer: Tackling Continual Domain Shift with Simultaneous Domain Generalization and Adaptation"*, including more implementation details, additional experimental analysis and results, and sensitivity analysis on parameters used in the framework. The Appendix is organized as follows:

- Section A provides more implementation details, including detailed experiment settings, data splitting strategy, implementation details of the baseline methods, and network structures.
- Section B presents experimental results of additional domain orders on three datasets. T-SNE visualization is also given there.
- Section C provides detailed sensitivity analysis of three parameters used in `Twofer`.

A IMPLEMENTATION DETAILS

Experiment Settings. To classify the datasets, we apply ResNet-50 as the feature extractor for both PACS and Domain Net. For Digits, we follow Liang et al. (2020) to use DTN as the feature extractor. Unlike regular domain adaptation works that use all training data for testing, we follow Wang & Lu to split 80% data as the training set and the rest 20% as the testing set for all domains of these three datasets. For each dataset, we keep the training steps per epoch the same for all domains in all different domain orders. We use 800 steps for Digits, 50 steps for PACS and 70 steps for Domain Net. Training epoch is 30 for all datasets and domains. The SGD optimizer with initial learning rate 0.01 is used for Digits and 0.005 for PACS and Domain Net. Moreover, we use momentum of 0.9 and weight decay of 0.0005 to schedule the SGD. The exemplar memory is set as 200 for all datasets, and we set the batch size of mini-batch training as 64. Only ResNet-50 is initialized as the pre-trained version of ImageNet.

Subset of Domain Net. The original Domain Net is class imbalanced, with certain classes in some domains containing very few images ($\sim$10). This makes it hard to assign pseudo labels for these classes. Thus, we select the top 10 classes with most images in all domains, which contain 26,013 samples in total. This dataset is still class imbalanced, with the smallest sample number for a class as 32, while the largest is 901. Therefore, this subset is still quite challenging.

Baseline Implementations. To our best knowledge, there is no existing study targeting the same problem. For fair comparison, we try our best to extend all baseline methods in our problem settings. AuC (Rostami, 2021) shares the most similar setting with us. Their exemplar memory size is not limited and can be enlarged as more domains appear. They transform Digits to gray scale images while we treat them as RGB images. For Test-Time/Online DA [CoT (Wang et al., 2022), TEN (Wang et al., 2020), T3A Iwasawa & Matsuo (2021)], we keep the setting that adapts the model on each target domain only once. For Single DG [L2D (Wang et al., 2021), PDE (Li et al., 2021)] and Multiple DG [PCL (Yao et al., 2022), EFD (Zhang et al., 2022)], we use SHO (Liang et al., 2020) to assign pseudo labels for the optimization on target domains. SNR (Jin et al., 2021) directly modifies the model structures to match statistics of different domains, thus it can be viewed as a unified approach for both DA and DG. In our implementation, we use SNR to modify the convolution filters of the feature extractor and optimize the model with SHO. All baseline methods are equipped with the same exemplar memory and the same feature extractor.

Network Structures. Our `RandMix` augmentation network includes four autoencoders that consist of a convolutional layer and a transposed convolutional layer. All these layers have 3 channels and kernel sizes of 5, 9, 13, 17, respectively. The classification model contains a feature extractor, a bottleneck module and a classifier. Specifically, pre-trained ResNet-50 is used as the feature extractor for both PACS and Domain Net. Following Liang et al. (2020), DTN, a variant of LeNet is used as the feature extractor for Digits. We also introduce a bottleneck module between the feature extractor and the classifier. The bottleneck consists of a fully-connected layer (256 units for Digits and 512 units for both PACS and Domain Net), a Batch Normalization layer, a ReLU layer and another fully-connected layer (128 units for Digits and 256 units for both PACS and Domain Net). The classifier is a fully-connected layer without the parameter fo bias. Both contrastive loss and distillation loss are applied on the representation space after the bottleneck.

Table 4: Performance comparisons between ours and other methods on Digits in TDG, TDA, and FA under an additional domain order (besides the ones in Table 1 of the main text). We blue and bold **the best** average performance, and bold **the second best**.

Order	Metric		CoT	AuC	SHO	GSF	BMD	TEN	T3A	L2D	PDE	SNR	PCL	EFD	Ours
SN → MT → MM → SD → US	TDG	MT	71.2	70.6	76.2	75.2	76.2	71.2	71.2	70.1	70.0	71.0	71.8	71.9	73.8
		MM	43.2	49.0	57.1	41.7	57.8	52.6	51.0	59.7	55.5	57.9	57.6	43.8	67.6
		SD	90.2	80.3	70.9	77.2	72.8	90.8	64.6	80.1	76.6	77.0	82.7	44.3	84.8
		US	61.9	74.8	85.8	79.0	89.6	82.1	67.7	72.2	70.1	73.8	85.8	39.8	88.1
		Avg.	66.6	68.6	72.5	68.3	74.1	74.2	63.6	70.5	68.1	69.9	**74.5**	50.0	**78.6**
	TDA	SN	92.8	92.6	93.4	93.6	93.4	92.8	92.8	92.8	93.0	92.9	93.4	92.9	92.1
		MT	47.8	73.4	98.6	97.9	98.5	74.6	69.4	78.8	76.5	79.0	86.6	62.2	90.2
		MM	34.5	45.4	74.8	59.0	84.0	52.4	36.4	67.7	65.4	67.0	65.0	15.6	85.7
		SD	88.5	68.9	59.7	65.2	76.9	90.1	45.7	82.0	79.8	81.3	76.6	12.2	88.5
		US	53.3	77.7	83.8	95.6	96.0	81.6	60.5	88.9	88.0	88.5	87.1	12.6	92.7
		Avg.	63.4	71.6	82.0	**82.3**	**89.8**	78.3	61.0	82.0	80.5	81.7	81.7	39.1	**89.8**
	FA	SN	88.5	63.5	30.5	90.1	46.6	92.0	40.8	82.3	81.0	81.4	68.9	16.0	75.6
		MT	42.1	73.6	84.0	48.4	94.4	72.4	52.5	70.9	65.1	68.7	85.4	38.6	91.3
		MM	33.2	47.4	53.2	63.4	64.3	52.8	35.2	57.9	55.3	58.0	62.6	20.7	80.3
		SD	88.5	67.9	54.0	79.0	69.2	90.1	49.7	71.1	70.0	72.6	75.9	17.8	87.4
		Avg.	63.1	63.1	55.4	70.2	68.6	**76.8**	44.6	70.6	67.9	70.2	73.2	23.3	**83.7**

Table 5: Performance comparisons between ours and other methods on PACS in TDG, TDA, and FA under an additional domain order (besides the ones in Table 2 of the main text). We blue and bold **the best** average performance, and bold **the second best**.

Order	Metric		CoT	AuC	SHO	GSF	BMD	TEN	T3A	L2D	PDE	SNR	PCL	EFD	Ours
A → C → P → S	TDG	C	65.0	64.0	61.2	67.0	61.2	65.0	65.0	63.7	63.0	63.5	63.1	69.9	70.2
		P	97.3	93.7	96.4	96.6	96.3	97.3	97.3	97.3	97.0	97.0	96.4	95.7	96.7
		S	62.3	57.7	54.8	63.2	55.3	62.4	58.5	60.1	58.6	57.1	53.3	67.3	70.0
		Avg.	74.9	71.8	70.8	75.6	70.9	74.9	73.6	73.7	72.9	70.5	70.9	**77.6**	**79.0**
	TDA	A	96.1	96.3	95.6	97.3	95.6	96.1	96.1	96.0	95.9	96.0	96.3	94.9	95.9
		C	74.8	67.0	82.9	85.3	84.9	74.8	70.1	71.5	70.7	71.0	64.6	79.3	79.5
		P	97.0	93.1	97.9	98.2	98.2	97.0	97.3	98.0	97.5	98.0	99.1	95.5	99.1
		S	64.5	62.1	67.6	77.5	71.6	64.8	71.6	72.3	70.1	70.9	77.9	55.9	77.4
		Avg.	83.1	79.6	86.0	**89.6**	87.6	83.2	83.8	84.5	83.6	84.0	84.5	81.4	**88.0**
	FA	A	94.6	87.6	86.7	66.9	87.2	94.7	92.8	91.9	91.0	91.5	90.7	83.2	93.0
		C	74.8	74.7	72.2	52.2	76.4	74.8	68.7	70.9	70.3	71.0	66.4	71.1	74.9
		P	97.0	93.7	92.5	20.7	92.2	96.7	94.6	95.0	94.4	95.0	95.2	84.4	97.9
		Avg.	**88.8**	85.3	83.8	46.6	85.3	88.7	85.4	85.9	85.2	85.8	84.1	79.6	**88.9**

B Additional Experiment Results

We conduct more experiments with additional domain orders for the three datasets. For Digits, we select the most complicated domain SVHN as the source domain (Table 4). For PACS, we randomly select another order (Table 5). For Domain Net, we select the reverse order as the one we show in the main text (Table 6). Under these additional domain orders, `Twofer` still achieves the highest average performance in most cases, and is very close to the best in other cases. In particular, `Twofer` achieves the best average TDG on all three datasets. Combining these results with the ones shown in the main text, we can clearly see that `Twofer` is able to provide substantial and balanced improvements across TDG, TDA, and FA over previous state-of-the-art methods.

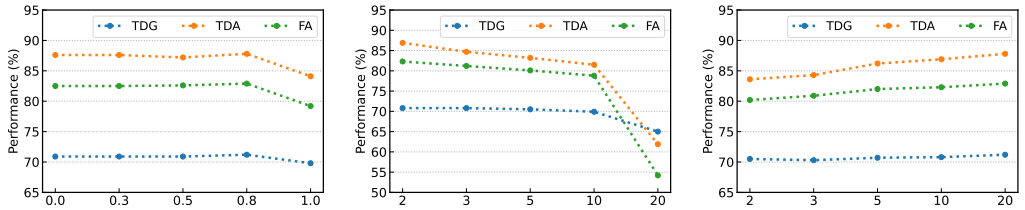
C Sensitivity Analysis

We conduct sensitive analysis of the confidence threshold p_{th} in splitting `RandMix`, and p_{top} and p'_{top} in `T2PL`. All sensitive analysis experiments are conducted on Digits with the domain order of US→SD→SN→MM→MT. For splitting `RandMix`, p_{th} controls the proportion of unlabeled data that needs to be augmented. As shown in Figure 5(a), different p_{th}-s correspond to different model performance, and we can obtain the best performance when $p_{\text{th}} = 0.8$, which is adopted in our experiments. As for the sensitivity analysis of the other two parameters, according to the results shown in Figures 5(b) and 5(c), we can observe that a larger p_{top} that corresponds to a

Table 6: Performance comparisons between ours and other methods on DomainNet in TDG, TDA, and FA under an additional domain order (besides the one in Table 3 of the main text). We blue and bold **the best** average performance, and bold **the second best**.

Order	Metric		CoT	AuC	SHO	GSF	BMD	TEN	T3A	L2D	PDE	SNR	PCL	EFD	Ours
		Pa	78.2	77.4	74.9	80.0	74.9	78.2	78.2	79.0	77.9	78.0	77.0	78.3	79.8
		In	30.7	26.0	23.7	23.7	24.0	30.7	25.1	26.1	25.0	25.0	26.0	24.0	29.7
	TDG	Cl	67.7	57.1	58.0	64.2	56.2	67.8	67.6	68.0	66.6	67.0	55.2	50.2	68.6
		Sk	64.1	60.5	58.1	58.5	55.8	64.3	62.6	63.3	62.5	62.8	55.7	42.0	64.7
Re		Qu	22.1	24.5	21.9	24.9	20.9	22.1	20.7	22.9	21.9	22.3	15.5	13.7	24.6
↓		Avg.	**52.6**	49.1	47.3	50.3	46.4	**52.6**	50.8	51.9	50.8	51.0	45.9	41.6	**53.5**
Pa		Re	98.4	98.4	98.2	98.6	98.2	98.4	98.4	98.4	98.3	98.3	98.0	99.0	98.2
↓		Pa	73.3	77.4	66.0	82.4	65.3	73.3	80.7	76.6	75.4	75.5	70.9	64.4	75.0
In	TDA	In	33.4	30.5	26.6	27.8	26.9	33.7	34.8	30.1	31.0	30.1	25.5	11.2	29.7
↓		Cl	68.8	61.1	68.1	67.2	55.8	69.1	72.5	70.7	69.1	70.0	56.0	34.1	70.3
Cl		Sk	64.6	62.8	54.2	58.8	51.9	65.1	67.5	65.5	65.0	65.2	53.1	23.6	66.7
↓		Qu	22.8	31.4	33.7	29.6	26.7	23.7	26.7	27.0	30.1	28.8	21.5	14.0	69.0
Sk		Avg.	60.2	60.3	57.8	**67.0**	54.1	33.7	63.4	61.4	61.5	61.3	54.2	41.1	**68.2**
↓		Re	97.8	94.3	89.7	75.6	87.8	97.6	95.0	94.1	92.2	93.0	91.4	54.9	97.5
Qu		Pa	73.4	78.2	65.2	53.3	63.2	73.5	75.7	71.3	70.0	71.1	64.0	37.8	74.3
	FA	In	33.4	31.4	23.4	14.0	22.5	33.8	28.4	28.0	27.1	27.5	23.4	14.9	29.0
		Cl	68.8	61.0	59.3	37.4	56.9	69.6	64.5	62.8	61.1	61.9	54.1	28.9	71.5
		Sk	64.6	60.7	43.0	24.1	44.5	64.9	53.1	60.3	59.7	60.7	49.5	29.2	68.2
		Avg.	67.6	65.1	56.1	40.9	55.0	**68.6**	63.3	63.3	62.0	62.8	56.5	33.1	**68.5**

smaller amount of fitting samples is detrimental to the model performance, while a larger p'_{top} that corresponds to fewer nearest samples for the kNN classification leads to slightly better performance. Therefore, we choose $p_{\text{top}} = 2$ and $p'_{\text{top}} = 20$ in our experiments.



(a) p_{th} (b) p_{top} (c) p'_{top}

Figure 5: Sensitivity analysis of confidence threshold p_{th} in `RandMix`; p_{top} and p'_{top} in `T2PL`.