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 – in what we call the *"Unfamiliar Period"*, especially when domain shifts happen suddenly and significantly. 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 and DG in handling the *Unfamiliar Period* during continual domain shift, we propose `Twofer`, a framework that focuses on improving learning models' target domain generalization (TDG) capability, while also achieving effective target domain adaptation (TDA) capability right after training on certain domains and forgetting alleviation (FA) capability on past domains. `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 DomainNet demonstrate that `Twofer` significantly outperforms state-of-the-art works from Continual DA, Source-Free DA, Test-Time/Online DA, Single DG, Multiple DG and Unified DA&DG in TDG, and achieves comparable TDA and FA capabilities.

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 for a period of time, in we call the *"Unfamiliar Period"*, 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*. In the following, we will first discuss related works, highlight their limitations in addressing the poor model performance during the *Unfamiliar Period*, 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, i.e., in 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 the *Unfamiliar Period*, i.e., before seeing a lot of target domain data. For instance in environmental 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 focus on the study of Continual Domain Shift Learning (CDSL) problem, in which 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. Our goal, in particular, is to improve model performance *before and during* the training stage of each previously-unseen target domain (i.e., in the *Unfamiliar Period*), while also maintains good performance in the time periods *after* the training. Thus, we propose a framework called `Twofer` that optimizes three objectives: (1) to improve the model generalization performance on a new target domain *before and during* its training, namely the *target domain generalization* (TDG) performance, (2) to provide good model performance on a target domain *right after* its training, namely the *target domain adaptation* (TDA) performance, and (3) to maintain good performance on a trained domain *after* the model is trained with other domains, namely the *forgetting alleviation* (FA) performance. 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 training domain. For TDA, `Twofer` includes a Top^2 Pseudo Labeling mechanism that lays more emphasis on samples with higher possibility of correct classification, 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). Comprehensive experiments and ablation studies on Digits, PACS and DomainNet demonstrate that `Twofer` can significantly outperform state-of-the-art works in TDG, including Continual DA, Source-Free DA, Test-Time/Online DA, Single DG, Multiple DG, and Unified DA&DG. `Twofer` can also produce comparable (and in many cases better) performance in TDA and FA as these baselines. The major contributions of our work can be summarized as:

- We tackle an important problem in practical scenarios with continual domain shifts, i.e., to improve model performance *before and during* training on a new target domain, in what we call the *Unfamiliar Period*. And we also try to achieve good model performance *after* training, providing the model with capabilities of target domain adaptation and forgetting alleviation.

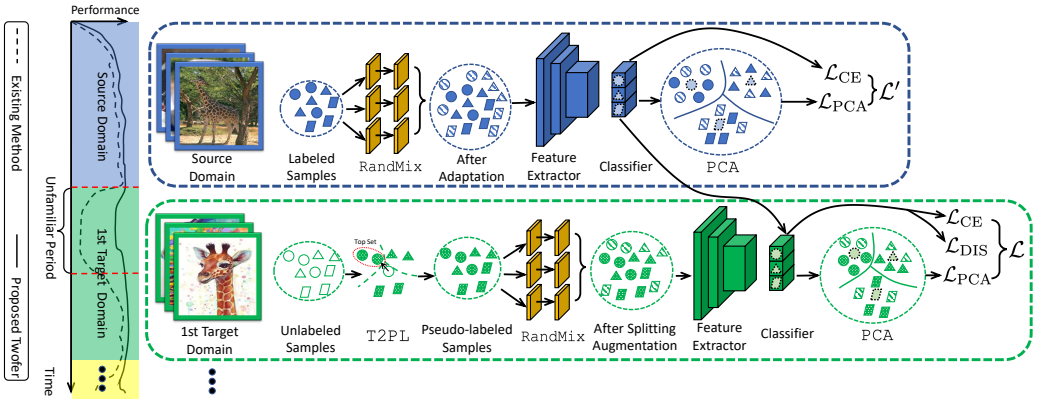


Figure 1: **Overview of applying our framework `Twofer` for Continual Domain Shift Learning (CDSL).** `Twofer` first starts with a labeled source domain, applies `RandMix` on the full set of source data to generate augmentation data, and uses a simplified version $\mathcal{L}'$ of `PCA` for model optimization. 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 `Twofer` to achieve our goal. The framework includes a training-free data augmentation module that generates more data for improving model's generalization ability, a new pseudo labeling mechanism to provide more accurate labels for domain adaptation, and a prototype contrastive alignment algorithm that effectively aligns domains to simultaneously improve the generalization ability and achieve target domain adaptation and forgetting alleviation.
- 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`. 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, which is a crucial technique for improving model generalization in the *Unfamiliar Period*. Section 2.3 presents a Top^2 Pseudo Labeling approach `T2PL` that provides accurate labels to achieve target domain adaptation (TDA) for continually arriving target domains. Finally, Section 2.4 presents a Prototype Contrastive Alignment algorithm `PCA` that optimizes the model to achieve TDG, TDA, and forgetting alleviation (FA) on seen domains. 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 dataset $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \| \boldsymbol{x}_i \sim \mathcal{P}_X^{\mathcal{S}}, \boldsymbol{y}_i \sim \mathcal{P}_Y^{\mathcal{S}}\}_{i=1}^{N_{\mathcal{S}}}$ 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 $\mathbf{T} = \{\mathcal{T}^t\}_{t=1}^{T}$, 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., $\mathbf{S} = \{\mathcal{S}, \cup_{j=1}^{t-1} \mathcal{T}^j\}$. 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 $\mathbf{S}$, where $\frac{|\mathcal{M}|}{t} \ll N_{\mathcal{T}}$, $\frac{|\mathcal{M}|}{t} \ll N_{\mathcal{S}}$. Moreover, the memory size is set to be fixed during CDSL. We assume that the model is a deep neural network, and without loss of generality, the neural network consists of a feature extractor f_θ at the bottom and a classifier g_ω at the top. For each target domain dataset $\mathcal{T}_t$, in addition to the current model $f_\theta^t \circ g_\omega^t$,

its inherited version $f_\theta^{t-1} \circ g_\omega^{t-1}$ from the last domain is also stored for later usage. We aim to improve the generalization performance on a new target domain (TDG) before and during its training, i.e., in its *Unfamiliar Period* of CDSL, while also achieving effective target domain adaptation (TDA) and forgetting alleviation (FA).

2.2 Random Mixup Augmentation

At the beginning of CDSL, only a labeled source domain is available. Thus we face a pure singe-domain generalization problem where the model is trained on a single domain $\mathcal{S}$ and tested on the other unseen domains $\mathbf{T}$ (each domain in $\mathbf{T}$ 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 $\mathbf{R} = \{R_i\}_{i=1}^{N_{\text{aug}}}$, where each autoencoder consists of an encoder e_ξ and a decoder d_ζ. We want `RandMix` to be as simple as possible, preferably to work without training. Inspired by (Wang et al., 2021), the encoder e_ξ and the decoder d_ζ are implemented as a convolutional layer and a transposed convolutional layer. With such implementation, even if the parameters 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 l_{ϕ_1}, l_{ϕ_2}, and when l_{ϕ_1} and l_{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:

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

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

$$\mathbf{R}(\boldsymbol{x}) = \sigma \left(\frac{1}{\sum_{i=0}^{N_{\text{aug}}} \text{w}_i} \left[\text{w}_0 \boldsymbol{x} + \sum_{i=1}^{N_{\text{aug}}} \left(\text{w}_i R_i(\boldsymbol{x}) \right) \right] \right). \tag{2}$$

In mini-batch training, every time there is a new data batch, parameters of autoencoders (ξ, ζ), AdaIN-s (ϕ_1, ϕ_2), AdaIN noisy inputs (n) and mixup weights ($\mathbf{w}$) are all 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 $\mathbf{T} = \{\mathcal{T}^t\}_{t=1}^{T}$ 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:

$$\tilde{\boldsymbol{x}} = \begin{cases} \mathbf{R}(\boldsymbol{x}), & \text{if} \max \left[g_\omega(f_\theta(\boldsymbol{x})) \right]_K \geq r_{\text{con}} \\ \emptyset, & \text{otherwise} \end{cases}, \boldsymbol{x} \sim \mathcal{P}_X^{\mathcal{T}}, \tag{3}$$

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

2.3 Top^2 Pseudo Labeling

In CDSL, all target domains arrive with only 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. However, when the model is applied in a new domain, it is difficult to distinguish different classes since they are nearly overlapped together and the class boundary among them is extremely vague. Besides, such clustering-based methods usually use all training samples to construct centroids for each class. During clustering, for a particular cluster, samples that exactly belong to the same class are often not used well and those mis-classified pose a negative impact on centroid construction. In this case, such negative impact hurts the pseudo labeling accuracy by resulting in low-quality centroids. To address these issues, we propose a novel mechanism called Top^2 Pseudo Labeling (T2PL). Specifically, we use the softmax confidence to measure the possibility of correct classification for data samples, and select the top 50% set to construct class centroids:

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

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

$$p_k = \frac{\sum_{\boldsymbol{x}_i \in \mathcal{F}} g_{\omega,k}\left(f_\theta(\boldsymbol{x}_i)\right) \cdot f_\theta(\boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_i \in \mathcal{F}} g_{\omega,k}\left(f_\theta(\boldsymbol{x}_i)\right)}. \tag{5}$$

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:

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

Then we use $\mathcal{F}'$ to fit a k-Nearest Neighbor (kNN) classifier and assign the pseudo label as below. Note that kNN can decentralize the risk of mis-classification in assigning pseudo labels by a single comparison between class centroids and data samples, and is more suitable for clustering samples that are highly overlapped than center-based approaches:

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

where the superscript $\frac{N_{\mathcal{T}^t}}{r'_{\text{top}} \cdot K}$ and subscript Euclidean mean that the kNN works to find $\frac{N_{\mathcal{T}^t}}{r'_{\text{top}} \cdot K}$ closest neighbors from $\mathcal{F}'$ with the measurement of Euclidean distance. We select the top 5% for each class ($r'_{\text{top}} = 20$) in our implementation (sensitivity analyses of r_{top} and r'_{top} are provided in the Appendix). With this kNN, T2PL makes better use of unlabeled samples with relatively high possibility of correct classification 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 $\mathbf{p}^t = \{\boldsymbol{p}_k^t\}_{k=1}^K$ of a certain domain $\mathcal{T}^t$. To achieve cross-domain alignment, prototypes of different domains for the same classes are aggregated and averaged. Then data samples from different domains are optimized to approach these domain-average prototypes. 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 $\boldsymbol{p}_k^*$ of class k for all domains ($\mathcal{S} \cup \mathbf{T}$), and the objective is to move the source prototype $\boldsymbol{p}_k^s$ to the optimal by training on target domains ($\mathbf{T}$) one-by-one. Meanwhile, we also regard that there are optimal prototypes $\boldsymbol{p}_k^t$ 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

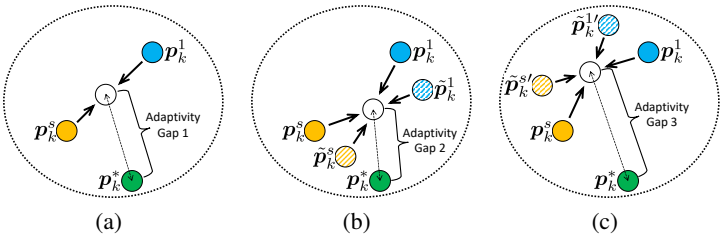


Figure 2: (a): When the model encounters a domain whose prototypes ($\boldsymbol{p}_k^1$) are far from the optimal ones ($\boldsymbol{p}_k^*$), domain alignment with regular prototype learning will enlarge the adaptivity gap. (b): RandMix may be helpful to reduce the gap in many cases of ($\tilde{\boldsymbol{p}}_k^s, \tilde{\boldsymbol{p}}_k^1$). (c): However, in some cases RandMix may produce low-quality data augmentation ($\tilde{\boldsymbol{p}}_k^{s\prime}, \tilde{\boldsymbol{p}}_k^{1\prime}$) that negatively affects the regular prototype learning and enlarges the gap. Here, we have Adaptivity Gap 3 > Adaptivity Gap 1 > Adaptivity Gap 2.

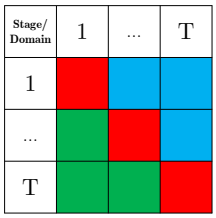


Figure 3: Accuracy Matrix. For each domain, TDG: mean of green elements; TDA: red elements; FA: mean of blue elements.

$\mathcal{T}^1$ as an example, and assume that the distance between $\boldsymbol{p}_k^1$ and $\boldsymbol{p}_k^*$ is larger than the distance between $\boldsymbol{p}_k^s$ and $\boldsymbol{p}_k^*$. 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 can use RandMix to generate more data ($\tilde{\boldsymbol{p}}_k^s$, $\tilde{\boldsymbol{p}}_k^1$) that might be helpful for reducing the adaptivity gap, there is randomness for each use of RandMix and we cannot guarantee such reduction every time (e.g., when the augmentation data is as $\tilde{\boldsymbol{p}}_k^{s\prime}$ and $\tilde{\boldsymbol{p}}_k^{1\prime}$ 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 g_ω consists of a single linear layer and is placed exactly behind the feature extractor f_θ. Thus the dimension of its neuron weights is the same as the hidden dimension of extracted representations. Prototypes $\boldsymbol{p}_k^t$-s are obtained with a CrossEntropy Loss:

$$\mathcal{L}_{\mathrm{CE}} = \mathbb{E}_{\boldsymbol{x}_i \sim \mathcal{P}_X^{\mathcal{T}^t}} \left[\sum_{k=1}^{K} -\log \frac{\mathbb{I}_{\hat{\boldsymbol{y}}_i = k} \exp\left(\boldsymbol{p}_k^{t\top} f_\theta(\boldsymbol{x}_i) + b_k\right)}{\sum_{c=1}^{K} \exp\left(\boldsymbol{p}_c^{t\top} f_\theta(\boldsymbol{x}_i) + b_c\right)} \right], \tag{8}$$

where b_k, b_c are biases of the linear layer, and we set them as zeros during training, and $\mathbb{I}_{(\cdot)}$ is an indicator function that if the subscript condition is true, $\mathbb{I}_{\mathrm{True}} = 1$, otherwise, $\mathbb{I}_{\mathrm{False}} = 0$.. 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:

$$\begin{aligned} \mathcal{L}_{\mathrm{PCA}} &= \mathbb{E}_{\boldsymbol{x}_i \sim \mathcal{P}_X^{\mathcal{T}^t}} \left\{ \sum_{k=1}^{K} -\log \frac{\mathbb{I}_{\hat{\boldsymbol{y}}_i = k} \left[\exp\left(\boldsymbol{p}_k^{t\top} f_\theta(\boldsymbol{x}_i)\right) + \exp\left(\boldsymbol{p}_k^{t-1\top} f_\theta(\boldsymbol{x}_i)\right) \right]}{\Delta} \right\}, \text{where} \\ \Delta &= \sum_{c=1}^{K} \exp\left(\boldsymbol{p}_c^{t\top} f_\theta(\boldsymbol{x}_i)\right) + \sum_{c=1}^{K} \exp\left(\boldsymbol{p}_c^{t-1\top} f_\theta(\boldsymbol{x}_i)\right) + \sum_{\boldsymbol{x}_j \in \mathcal{T}^t, j \neq i} \mathbb{I}_{\hat{\boldsymbol{y}}_i \neq \hat{\boldsymbol{y}}_j} \exp\left(f_\theta(\boldsymbol{x}_i)^\top f_\theta(\boldsymbol{x}_j)\right). \end{aligned} \tag{9}$$

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:

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{PCA}} + \mathcal{L}_{\mathrm{DIS}}, \text{where } \mathcal{L}_{\mathrm{DIS}} = \mathcal{D}_{\mathrm{KL}} \left[g_\omega^{t-1}\left(f_\theta^{t-1}(\boldsymbol{x})\right) \parallel g_\omega^{t}\left(f_\theta^{t}(\boldsymbol{x})\right) \right]_{\boldsymbol{x} \sim \mathcal{P}_X^{\mathcal{T}^t}}. \tag{10}$$

Note that this optimization objective is used in all target domains. As for the source domain, there is no distillation $\mathcal{L}_{\mathrm{DIS}}$ and the nominator of $\mathcal{L}_{\mathrm{PCA}}$ 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 provided in the Supplementary Materials. The datasets, experiment settings and comparison baselines are introduced below, and more details are provided in appendix.

Datasets. **Digits** consists of 5 different domains with 10 classes, including MNIST (MT), SVHN (SN), MNIST-M (MM), SYN-D (SD) and USPS (US). **PACS** contains 4 domains with 7 classes, including Photo (P), Art painting (A), Cartoon (C) and Sketch (S). **DomainNet** is the most challenging cross-domain dataset, including Quickdraw (Qu), Clipart (Cl), Painting (Pa), Infograph (In), Sketch (Sk) and Real (Re). Considering label noise and class imbalance, we follow Xie et al. (2022) to split a subset of DomainNet for our experiments.

Experiment Settings. We apply ResNet-50 as the feature extractor for both PACS and DomainNet, and apply DTN as the feature extractor for Digits (Liang et al., 2020). We randomly 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 DomainNet. 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. `Twofer` is compared with a comprehensive set of state-of-the-art works from Continual DA [CoTTA (Wang et al., 2022), AuCID (Rostami, 2021)], Source-Free DA [SHOT (Liang et al., 2020), GSFDA (Yang et al., 2021), BMD (Qu et al., 2022)], Test-Time/Online DA [Tent (Wang et al., 2020), T3A (Iwasawa & Matsuo, 2021)], Single DG [L2D (Wang et al., 2021), PDEN (Li et al., 2021)], Unified DA&DG [SNR (Jin et al., 2021)], and Multiple DG [PCL (Yao et al., 2022), EFDM (Zhang et al., 2022)].

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	CoTTA	AuCID	SHOT	GSFDA	BMD	Tent	T3A	L2D	PDEN	SNR	PCL	EFDM	Ours
MT ↓ MM ↓ SN ↓ SD ↓ US	TDG	MM	52.3	44.9	55.3	55.2	55.3	52.3	52.3	**73.1**	66.6	70.1	51.0	50.5	**86.1**
		SN	24.1	15.8	25.4	28.3	24.9	30.6	26.8	**36.0**	35.4	35.7	23.1	12.1	**45.7**
		SD	34.3	29.3	43.6	30.7	42.5	41.7	34.6	**56.3**	56.0	56.1	35.9	16.3	**67.7**
		US	**90.0**	83.6	69.1	50.5	73.7	84.9	71.0	84.8	83.2	81.7	73.3	35.7	**92.2**
		Avg.	50.2	43.4	48.3	41.2	49.1	52.4	46.2	**62.6**	60.3	60.9	45.8	28.7	**72.9**
	TDA	MT	99.0	99.1	**99.4**	**99.4**	**99.4**	99.0	99.0	**99.3**	99.1	99.2	99.2	99.2	**99.4**
		MM	41.7	41.4	84.6	54.3	80.6	57.5	51.5	**87.4**	84.4	85.0	55.0	14.0	**90.3**
		SN	17.8	18.5	33.4	13.7	34.6	29.4	25.2	**55.3**	50.0	51.1	22.6	9.5	**70.6**
		SD	27.8	30.1	41.9	13.1	45.6	40.2	32.4	**59.9**	50.9	45.2	28.0	11.4	**82.7**
		US	89.9	79.5	75.9	14.8	80.6	90.5	74.5	**91.6**	90.0	91.1	59.2	16.1	**94.9**
		Avg.	55.2	53.7	67.1	39.1	68.2	63.3	56.5	**78.7**	74.9	74.3	52.8	30.0	**87.6**
	FA	MT	93.9	92.6	73.5	33.9	78.9	**98.9**	86.0	92.4	91.5	92.0	83.2	34.4	**96.1**
		MM	33.1	48.5	38.8	10.5	44.0	55.2	44.9	70.7	69.8	**71.1**	38.2	16.2	**82.0**
		SN	17.3	20.4	16.5	9.0	17.4	29.4	16.8	**56.1**	50.3	55.4	20.8	12.1	**70.3**
		SD	27.3	28.8	41.1	10.4	45.9	40.2	27.7	70.0	69.1	**71.3**	28.6	16.3	**81.9**
		Avg.	42.9	47.6	42.5	16.0	46.6	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	37.5	37.2	37.5	36.4	36.4	**61.7**	60.0	61.5	37.1	36.8	**68.2**
		SN	19.2	19.9	18.4	22.5	18.5	26.4	20.4	**52.6**	51.1	52.5	21.9	14.4	**61.5**
		MM	30.9	33.8	25.3	32.6	29.5	40.4	34.6	**63.6**	60.7	60.5	34.9	19.2	**68.0**
		MT	54.2	63.0	37.4	55.6	56.3	77.2	58.9	**84.0**	79.9	81.1	63.0	29.0	**87.0**
		Avg.	35.2	37.6	29.7	37.0	35.4	45.1	37.6	**65.5**	62.9	63.9	39.2	24.9	**71.2**
	TDA	US	98.5	98.6	98.6	98.3	98.6	98.5	98.5	**99.0**	98.8	**98.9**	98.6	98.3	**98.9**
		SD	22.5	24.8	32.8	40.1	30.8	38.5	29.9	**72.5**	71.0	70.8	37.8	9.9	**82.7**
		SN	14.3	18.7	21.5	17.2	23.8	28.3	20.1	**69.1**	65.6	64.0	20.0	10.0	**78.2**
		MM	25.9	32.8	19.3	19.6	29.2	39.5	34.2	**76.0**	75.5	75.0	29.4	10.3	**83.4**
		MT	44.3	55.0	30.3	19.5	54.2	79.5	67.4	**87.0**	85.4	83.9	55.5	19.6	**95.7**
		Avg.	41.1	46.0	40.5	38.9	47.3	56.9	50.0	**80.7**	79.3	78.5	48.3	29.6	**87.8**
	FA	US	95.3	91.9	43.7	49.7	57.6	**98.4**	76.1	93.6	91.0	92.2	82.1	11.3	**96.3**
		SD	19.9	25.8	22.1	22.1	26.1	37.1	23.5	**72.7**	70.4	69.5	27.6	10.7	**82.0**
		SN	13.7	19.7	13.2	8.7	14.3	27.9	12.3	**56.8**	55.9	56.0	18.2	14.1	**68.2**
		MM	26.1	30.1	14.2	9.8	22.8	40.6	16.4	**74.2**	70.1	71.2	29.0	29.0	**85.1**
		Avg.	38.8	41.9	23.3	22.6	30.2	51.0	32.1	**74.3**	71.9	72.2	39.2	16.3	**82.9**

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, as shown in Figure 3. Target domain generalization (TDG) is measured by the mean model performance on a domain *before* its training stage (mean of the green elements in Figure 3), target domain adaptation (TDA) by the performance on a domain *right after* finishing its training stage (the red element), and forgetting alleviation (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 DomainNet (Table 3) in one domain order. Results for more domain orders can be found in the Appendix. We can observe that **`Twofer` achieves significantly higher performance in TDG than all baselines in all cases, and comparable (and in many cases higher) performance in TDA and FA.** Specifically, `Twofer` significantly outperforms the second-best baseline on TDG in all cases and by 3.1∼10.3% 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 in many cases); and 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.

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		CoTTA	AuCID	SHOT	GSFDA	BMD	Tent	T3A	L2D	PDEN	SNR	PCL	EFDM	Ours
P ↓ A ↓ C ↓ S	TDG	A	68.0	66.6	**68.3**	67.3	**68.3**	68.0	68.0	67.2	67.2	67.3	68.1	67.4	**70.0**
		C	48.6	32.1	32.0	47.6	34.9	48.6	34.8	**49.3**	47.2	43.1	40.5	40.4	**51.4**
		S	45.3	39.3	29.1	46.3	31.4	45.5	38.1	**52.6**	52.2	47.7	41.6	42.2	**57.1**
		Avg.	54.0	46.0	43.1	53.7	44.9	54.0	47.0	**56.4**	55.5	52.7	50.1	50.0	**59.5**
	TDA	P	**99.4**	**99.4**	97.0	98.8	97.0	**99.4**	**99.4**	99.1	99.2	99.0	98.8	99.1	**99.7**
		A	73.2	64.9	85.0	**87.6**	86.6	72.9	68.8	72.4	71.1	72.0	77.1	74.9	**87.1**
		C	66.1	56.9	70.9	71.9	**75.0**	66.7	61.2	61.4	60.7	62.0	63.1	35.0	**75.3**
		S	51.0	58.1	62.7	**70.6**	61.6	52.3	61.2	68.2	65.5	66.6	64.6	32.8	**70.0**
		Avg.	72.4	69.8	78.9	**82.2**	80.0	72.8	72.7	75.3	74.1	74.9	75.9	60.5	**83.0**
	FA	P	**98.5**	97.1	95.2	70.8	94.2	**98.5**	**98.6**	94.2	94.0	95.5	74.5	81.4	98.1
		A	73.2	73.3	78.7	41.4	**79.7**	73.2	65.5	67.4	66.0	67.3	60.8	60.1	**87.0**
		C	66.1	**67.8**	66.3	32.0	61.0	67.0	59.1	60.1	58.8	61.3	41.6	49.0	**72.5**
		Avg.	79.3	79.4	**80.1**	48.1	78.3	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	46.5	61.0	46.5	61.6	61.6	64.1	63.1	64.0	50.3	**64.7**	**68.4**
		A	**55.1**	41.6	41.8	47.0	43.8	**55.1**	54.5	51.0	50.4	51.3	35.0	32.2	**66.6**
		P	60.7	52.8	71.0	59.7	**71.1**	60.7	67.2	69.8	70.0	**71.1**	63.7	34.6	**83.3**
		Avg.	59.1	49.4	53.1	55.9	53.8	59.1	61.1	**61.6**	61.2	61.2	49.7	43.8	**72.8**
	TDA	S	95.4	95.5	94.7	**96.2**	94.7	95.4	95.4	**96.3**	96.0	96.1	96.1	95.9	**96.3**
		C	68.2	66.5	**86.8**	79.7	**87.8**	68.0	76.5	79.4	79.0	79.0	75.9	68.0	76.5
		A	61.5	69.3	88.2	86.8	**89.1**	61.5	70.2	69.8	67.9	70.0	56.3	13.7	**89.3**
		P	59.3	74.3	95.1	68.0	**95.6**	59.3	72.5	86.8	85.5	86.7	92.2	13.8	**99.1**
		Avg.	71.1	76.4	**91.2**	82.7	**91.8**	71.1	78.7	83.1	82.1	83.0	80.1	47.9	90.3
	FA	S	**94.0**	71.5	88.5	87.2	87.2	**95.0**	93.9	88.6	88.0	89.3	90.7	81.8	**94.0**
		C	68.2	74.2	**84.7**	63.5	**86.4**	68.0	69.4	78.0	74.9	78.2	75.0	52.9	80.9
		A	61.5	52.8	84.5	81.2	**85.4**	61.7	69.3	71.7	70.4	71.0	60.2	21.2	**87.8**
		Avg.	74.6	66.2	85.9	77.3	**86.3**	74.9	77.5	79.4	77.8	79.5	75.3	52.0	**87.6**

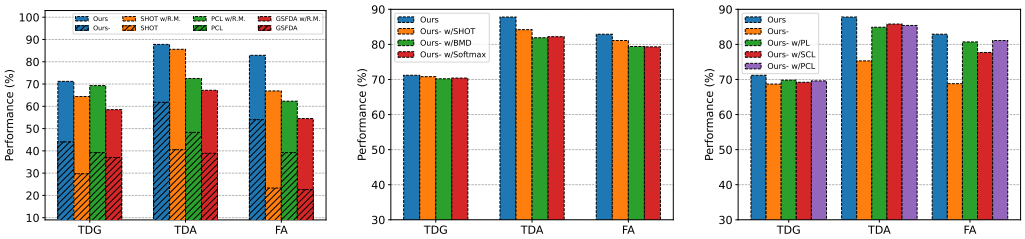
3.2 ABLATION STUDIES

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. The ablation studies on PACS and DomainNet can be found in the Appendix.

Effectiveness of `RandMix`. We first remove `RandMix` from `Twofer` and see how the model performance will change. We also try to apply `RandMix` on a few other baselines, including SHOT (Liang et al., 2020), GSFDA (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 perfor-

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

Order	Metric		CoTTA	AuCID	SHOT	GSFDA	BMD	Tent	T3A	L2D	PDEN	SNR	PCL	EFDM	Ours
Qu ↓ Sk ↓ Cl ↓ In ↓ Pa ↓ Re	TDG	Sk	28.2	28.6	23.0	29.7	23.0	28.2	28.2	**31.7**	29.8	29.9	27.9	30.3	**32.8**
		Cl	51.9	42.6	22.0	**52.2**	24.8	52.1	49.4	52.1	50.2	51.0	41.4	28.0	**55.0**
		In	17.1	15.7	12.8	13.6	12.2	**17.2**	16.1	13.2	12.7	11.1	13.2	11.5	**17.5**
		Pa	25.4	23.1	15.3	**26.6**	16.5	25.6	20.8	24.1	23.0	24.3	13.9	13.3	**42.9**
		Re	52.8	41.7	29.5	44.4	33.6	**53.2**	49.2	39.6	37.9	37.0	36.7	18.1	**65.8**
		Avg.	35.1	30.3	20.5	33.3	22.0	**35.3**	32.7	32.3	30.7	30.7	26.6	20.2	**42.8**
	TDA	Qu	**92.7**	**92.7**	92.0	**94.8**	92.0	92.0	92.1	91.8	92.0	91.8	91.5	90.9	92.0
		Sk	36.9	31.1	34.8	**45.5**	37.3	37.0	36.7	37.4	36.6	37.0	31.7	12.3	**47.8**
		Cl	59.4	52.9	34.4	58.9	36.2	**60.1**	59.2	55.1	54.1	55.7	49.3	8.7	**66.7**
		In	19.1	18.5	17.0	**20.9**	13.0	**19.7**	19.6	13.3	13.0	15.1	16.0	9.3	18.5
		Pa	**30.8**	30.2	19.6	28.1	23.5	31.2	31.2	16.0	15.0	20.3	10.6	12.6	**50.6**
		Re	57.7	45.1	34.5	44.3	37.1	**58.9**	58.3	41.9	41.1	42.0	33.6	4.2	**78.2**
		Avg.	49.4	45.1	38.7	48.8	39.8	**49.8**	49.5	42.6	42.0	43.7	38.8	23.0	**59.0**
	FA	Qu	**91.4**	84.4	42.2	75.9	33.7	**91.3**	82.3	79.6	77.9	80.0	80.8	64.2	89.3
		Sk	36.9	35.0	29.2	36.2	29.1	**37.3**	36.6	32.5	31.7	32.8	30.3	15.7	**53.8**
		Cl	59.0	54.3	29.1	42.8	32.1	**60.5**	53.5	44.4	43.3	45.0	42.7	20.0	**66.1**
		In	19.3	18.0	15.4	11.9	13.3	19.7	**20.1**	12.9	11.9	12.0	14.3	11.6	**20.3**
		Pa	**31.7**	31.2	21.4	30.8	22.1	31.0	30.2	15.3	17.3	20.1	8.1	7.5	**54.5**
		Avg.	47.7	44.6	27.4	39.5	26.0	**48.0**	44.5	36.9	36.4	38.0	35.2	23.8	**56.8**



(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 to remove a corresponding module (a, b or c) from `Twofer`, 'Ours- w/' means replacing the corresponding module with a new one.

mance significantly, 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), SHOT (Liang et al., 2020), and BMD (Qu et al., 2022). Figure 4(b) shows that `Twofer` performs the best when it has `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 (SCL), 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`.

4 Conclusion

This paper focuses on improving the model performance *before and during* the training of continually arriving domains, in what we call the *Unfamiliar Period*. To solve this practical and important problem, we propose 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 significantly improve the model performance in the *Unfamiliar Period*, while also achieving good performance in target domain adaptation and forgetting alleviation, in comparison with 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.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019.

Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 295–305, 2022.

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.

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

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.

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.

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.

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

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.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. *arXiv preprint arXiv:2204.02610*, 2022.

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.

Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

Shixiang Tang, Peng Su, Dapeng Chen, and Wanli Ouyang. Gradient regularized contrastive learning for continual domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 2665–2673, 2021.

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.

Jiangwei Xie, Shipeng Yan, and Xuming He. General incremental learning with domain-aware categorical representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14351–14360, 2022.

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 related work comparison, more implementation details of main experiments, additional experiment results and ablation studies, and sensitivity analysis on parameters used in the framework. The Appendix is organized as follows:

- Section A discusses the difference between `Twofer` and related works.
- Section B provides more implementation details, including detailed experiment settings, data splitting strategy, implementation details of the baseline methods, and network structures.
- Section C presents experiment results of additional domain orders on three datasets (Section C.1), ablation studies on PACS and DomainNet (Section C.2), and performance evaluation on forward and backward transfer (Section C.3).
- Section D provides detailed sensitivity analysis of three parameters used in `Twofer`.
- Section E shows additional evaluations of `Twofer` in more settings, including standard data splitting of domain adaptation (Section E.1), and stationary domain adaptation (Section E.2).

A RELATED WORK COMPARISON

Our work is relevant to a number of different topics as we focus on addressing important and practical problems. We show a detailed comparison among these relevant topics in Table 4. Specifically, Fine-Tuning (Tajbakhsh et al., 2016) is usually conducted on a trained model with labeled target domain data. It can achieve certain domain adaptation capability after sufficient training, but the forgetting on previously learned knowledge is often severe. Continual Learning (Lopez-Paz & Ranzato, 2017; Rebuffi et al., 2017), which shares the same settings as Fine-Tuning, can effectively achieve alleviation of catastrophic forgetting. To relax the demand for target labels, a series of domain adaptation (DA) methods are proposed. Unsupervised DA (Ganin & Lempitsky, 2015) can work effectively when the target data is unlabeled, but suffers from poor forgetting alleviation capability in continually changing cases. Continual DA (Wang et al., 2022; Rostami, 2021; Tang et al., 2021) is able to compensate the forgetting on learned knowledge, while achieving good domain adaptation, but it requires the access of source data. Source-Free DA (Yang et al., 2021; Liang et al., 2020; 2021; Qu et al., 2022) is proposed to adapt the model to unlabeled target domains without the need of source data, but the model forgets a lot of learned knowledge during adaptation. Test-Time DA (Wang et al., 2020; Iwasawa & Matsuo, 2021; Chen et al., 2022; Niu et al., 2022) does not need source data and target labels, and can achieve effective domain adaptation without heavy catastrophic forgetting on old knowledge. However, all these DA methods cannot achieve effective target domain generalization on new domains *before and during* the training on them, which is the main focus of our work here.

Different from DA, Domain Generalization (DG), as another cross-domain learning paradigm, can improve model generalization performance even without seeing target domains. According to the composition of the given source domain, DG can be divided into two types, Single Source (Wang et al., 2021; Li et al., 2021) and Multi-Source (Yao et al., 2022; Zhang et al., 2022). Both these two DG-s require labeled source domain data, and can address target domain generalization before any adaptation on target domains. However, they cannot achieve effective target domain adaptation and forgetting alleviation, which are important in continually changing scenarios. Recently, there are some works (Jin et al., 2021) that try to unify DA and DG, but they require significant resources such as source labeled data and target labeled data. Moreover, such Unified DA & DG also suffers from forgetting when being applied in continual learning settings. Compared with these studies, our work focuses on improving model performance on new target domains *before and during* the training on them, in we call the *Unfamiliar Period* in the main text, while also achieving effective target domain adaptation and forgetting alleviation. To ensure the application practicability, we assume that there

Table 4: Characteristics of various problem settings that adapt a model to potentially shifted target domains, especially assuming the domain shift is continually changing.

Topics	Source Data	Target Data	Target Label	Target Domain Generalization	Target Domain Adaptation	Forgetting Alleviation
Fine-Tuning	✗	✓	✓	✗	✓	✗
Continual Learning	✗	✓	✓	✗	✓	✓
Unsupervised DA	✓	✓	✗	✗	✓	✗
Continual DA	✓	✓	✗	✗	✓	✓
Source-Free DA	✗	✓	✗	✗	✓	✗
Test-Time DA	✗	✓	✗	✗	✓	✓
Single Source DG	✓	✗	✗	✓	✗	✗
Multi-Source DG	✓	✗	✗	✓	✗	✗
Unified DA & DG	✓	✓	✓	✓	✓	✗
Twofer	✗	✓	✗	✓	✓	✓

is only unlabeled data from target domains. As shown in the main text, the proposed `Twofer` framework can effectively solve the problem we target under such assumptions.

B Implementation Details

Experiment Settings. To classify the datasets, we apply ResNet-50 as the feature extractor for both PACS and DomainNet. 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 DomainNet. 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 DomainNet. Moreover, we use momentum of 0.9 and weight decay of 0.0005 to schedule the SGD. The exemplar memory is set as 2000 for DomainNet-126 and 200 for all rest 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 DomainNet. The original DomainNet 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. Furthermore, as mentioned in Chen et al. (2022), the original DomainNet contains many noisy labels and a subset called DomainNet-126 is filtered out to avoid the two most challenging domains: Quickdraw and Infograph. Our work focuses on continual domain shifts and we want to evaluate our methods in more challenging domains. Thus, we refer to Xie et al. (2022) to 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 is 32, while the largest is 901. Therefore, this subset is still quite challenging.

Baseline Implementations. For fair comparison, we try our best to extend all baseline methods in our problem settings. AuCID (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 [CoTTA (Wang et al., 2022), Tent (Wang et al., 2020), T3A Iwasawa & Matsuo (2021), AdaCon (Chen et al., 2022), EATA (Niu et al., 2022)], we keep the setting that adapts the model on each target domain only once. For Single DG [L2D (Wang et al., 2021), PDEN (Li et al., 2021)] and Multiple DG [PCL (Yao et al., 2022), EFDM (Zhang et al., 2022)], we use SHOT (Liang et al., 2020) to assign pseudo labels for the optimization on target domains. In EFDM, we use samples from current domain as the content images and randomly select images in the replay memory as the style images. 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 SHOT. We also consider a more advanced version of SHOT for further comparison, i.e., SHOT++ (Liang et al., 2021), which equips an additional semi-supervised traning step using MixMatch (Berthelot et al., 2019). This may not be fair for other methods because all methods can in principle equip with this additional training step.

Thus, we also run SHOT++ without the MixMatch and denote it as SHOT+. 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 DomainNet. 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 DomainNet), a Batch Normalization layer, a ReLU layer and another fully-connected layer (128 units for Digits and 256 units for both PACS and DomainNet). The classifier is a fully-connected layer without the parameters of bias. Both contrastive loss and distillation loss are applied to the representations after the bottleneck.

C Additional Experiment Results

C.1 Additional Domain Orders

Following existing studies of continual domain shifts (Rostami, 2021; Wang et al., 2022; Tang et al., 2021), we have presented experiment results under the most popular domain orders in the main text. To further explore the impact of domain order, which is rarely studied by existing works, we conduct more experiments with additional domain orders for the three datasets. Specifically, for each dataset, we carry out experiments under 10 different domain orders that are randomly shuffled, shown in Table 5. In addition to the orders presented in the main text, we report detailed experiment results for one additional order (Order 1 in Table 5) in Table 6 (Digits), Table 7 (PACS), Table 8 (DomainNet), and present the average performance for all orders in Table 9 (Digits), Table 10 (PACS), Table 11 (DomainNet). Note that for the average performance in Tables 9, 10 and 11, we select a subset of the baseline methods that perform relatively better than the rest, including a few additional baselines (i.e., SHOT+ (Liang et al., 2021), SHOT++ (Liang et al., 2021), AdaCon (Chen et al., 2022), EATA (Niu et al., 2022), L2D (Wang et al., 2021), and PDEN (Li et al., 2021)), to carry out experiments for saving time and space. Under these additional domain orders, `Twofer` still achieves the best average TDG on all three datasets, and comparable (and in many cases better) performance in TDA and FA. Combining these results with the ones shown in the main text, we can clearly see that `Twofer` provides significant improvement on TDG over previous state-of-the-art methods, while achieving comparable performance in TDA and FA.

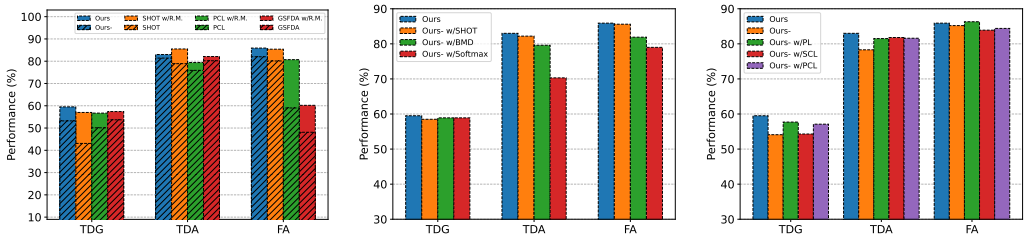
Table 5: Different domain orders used to assess their impact on various approaches' performance. The results are reported in Tables 6, 7, and 8 for Order 1, and in Tables 9, 10, and 11 for the average performance of all orders.

Order Datasets	Digits	PACS	DomainNet
Order 1	SN→MT→MM→SD→US	A→C→P→S	Re→Pa→In→Cl→Sk→Qu
Order 2	SN→SD→MT→US→MM	A→C→S→P	Cl→In→Pa→Qu→Re→Sk
Order 3	MM→US→MT→SD→SN	A→P→C→S	Cl→Re→In→Qu→Sk→Pa
Order 4	MT→MM→SN→SD→US	C→A→S→P	In→Qu→Cl→Pa→Re→Sk
Order 5	MT→MM→US→SN→SD	C→S→P→A	Pa→Sk→Qu→In→Re→Cl
Order 6	SD→MM→SN→MT→US	P→A→C→S	Qu→Re→Cl→Pa→In→Sk
Order 7	SD→SN→US→MM→MT	P→S→A→C	Qu→Sk→Cl→In→Pa→Re
Order 8	SD→US→MM→SN→MT	P→S→C→A	Sk→In→Pa→Cl→Re→Qu
Order 9	US→MT→SN→MM→SD	S→C→A→P	Sk→Re→Pa→Cl→Qu→In
Order 10	US→SD→SN→MM→MT	S→P→C→A	Sk→Re→Qu→Pa→In→Cl

C.2 Additional Ablation Studies

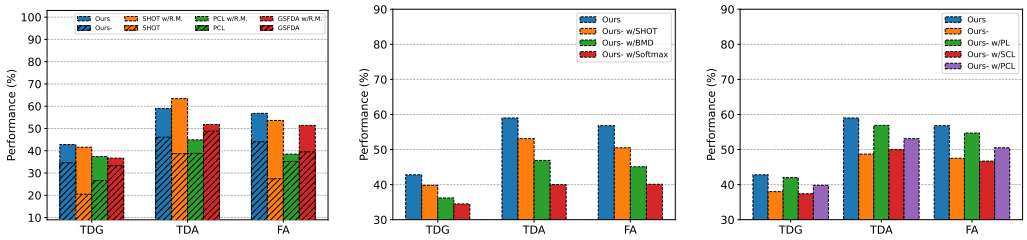
In addition to the ablation studies in the main text for Digits, we also conduct ablation studies here on PACS and DomainNet. Specifically, we choose the order P→A→C→S for PACS and Qu→Sk→Cl→In→Pa→Re for DomainNet to carry out experiments. For the ablation study of each module, `RandMix`, `T2PL`, and `PCA`, we follow the same setting as the main text (Section 3.2). Figure 5 shows the ablation study results for PACS, and Figure 6 for DomainNet. We can con-

clude similar observations as the ablation studies of Digits (Section 3.2), i.e., all three modules, RandMix, T2PL, and PCA, play essential roles in enabling Twofer to perform optimally. In addition, although each individual module can help enhance the performance when being applied in other baseline methods, those modified baselines still perform worse than Twofer. This shows the advantages of the overall design of Twofer.



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

Figure 5: Ablation studies of RandMix, T2PL, and PCA on PACS. Average performance of all domains for three metrics (TDG, TDA, FA) are shown. "Ours" here denotes the full framework of Twofer, while "Ours-" represents to remove a corresponding module (a, b or c) from Twofer, "Ours- w/" means replacing the corresponding module with a new one.



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

Figure 6: Ablation studies of RandMix, T2PL, and PCA on DomainNet. Average performance of all domains for three metrics (TDG, TDA, FA) are shown. "Ours" here denotes the full framework of Twofer, while "Ours-" represents to remove a corresponding module (a, b or c) from Twofer, "Ours- w/" means replacing the corresponding module with a new one.

C.3 Performance Evaluation on Forward and Backward Transfer

In this section, we use "forward transfer" (FT) and "backward transfer" (BT) to evaluate the performance of different approaches on main experiments. FT and BT are usually used in standard continual learning (Lopez-Paz & Ranzato, 2017), and they measure the relative performance change over time. To our best knowledge, BT and FT are not commonly used metrics for CDSL. Our definitions of TDA and FA are perhaps more suitable for evaluating the model performance in CDSL since they capture the "absolute" performance rather than the relative performance change measured by FT and BT. In spite of this, we also try to use FT and BT to evaluate the model performance in CDSL, shown in Table 12. According to these results, we can observe that Twofer can achieve best FT in most cases, while performing very well in BT.

D Sensitivity Analysis

We conduct sensitive analysis of the confidence threshold r_{con} in splitting RandMix, and r_{top} and r'_{top} in T2PL. All sensitive analysis experiments are conducted on Digits with the domain order of US→SD→SN→MM→MT. For splitting RandMix, r_{con} controls the proportion of unlabeled data that needs to be augmented. As shown in Figure 7(a), different r_{con}-s correspond to different model performance, and we can obtain the best performance when $r_{\text{con}} = 0.8$, which is adopted in our experiments. As for the sensitivity analysis of the other two parameters, according to the results shown in Figures 7(b) and 7(c), we can observe that a larger r_{top} that corresponds to a

Table 6: 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	CoTTA	AuCID	SHOT	GSFDA	BMD	Tent	T3A	L2D	PDEN	SNR	PCL	EFDM	Ours
SN → MT → MM → SD → US	TDG	MT	71.2	70.6	76.1	75.2	76.1	71.2	71.2	70.1	70.0	71.0	71.8	71.9	73.8
		MM	43.2	49.0	56.3	41.7	57.3	52.6	51.0	59.7	55.5	57.9	57.6	43.8	67.6
		SD	90.2	80.3	71.0	77.2	71.7	90.8	64.6	80.1	76.6	77.0	82.7	44.3	84.8
		US	61.9	74.8	83.1	79.0	81.6	82.1	67.7	72.2	70.1	73.8	85.8	39.8	88.1
		Avg.	66.6	68.7	71.6	68.3	71.7	74.2	63.6	70.5	68.1	69.9	**74.5**	50.0	**78.6**
	TDA	SN	92.8	92.6	94.1	93.6	94.1	92.8	92.8	92.8	93.0	92.9	93.4	92.9	92.1
		MT	47.8	73.4	98.8	97.9	98.8	74.6	69.4	78.8	76.5	79.0	86.6	62.2	90.2
		MM	34.5	45.4	72.9	59.0	73.9	52.4	36.4	67.7	65.4	67.0	65.0	15.6	85.7
		SD	88.5	68.9	65.0	65.2	67.0	90.1	45.7	82.0	79.8	81.3	76.6	12.2	88.5
		US	53.3	77.7	77.5	95.6	90.7	81.6	60.5	88.9	88.0	88.5	87.1	12.6	92.7
		Avg.	63.4	71.6	81.7	82.3	**84.9**	78.3	61.0	82.0	80.5	81.7	81.7	39.1	**89.8**
	FA	SN	88.5	63.5	33.6	90.1	37.8	92.0	40.8	82.3	81.0	81.4	68.9	16.0	75.6
		MT	42.1	73.6	81.8	48.4	89.5	72.4	52.5	70.9	65.1	68.7	85.4	38.6	91.3
		MM	33.2	47.4	46.9	63.4	56.7	52.8	35.2	57.9	55.3	58.0	62.6	20.7	80.3
		SD	88.5	67.9	61.4	79.0	68.9	90.1	49.7	71.1	70.0	72.6	75.9	17.8	87.4
		Avg.	63.1	63.1	55.9	70.2	63.2	**76.8**	44.6	70.6	67.9	70.2	73.2	23.3	**83.7**

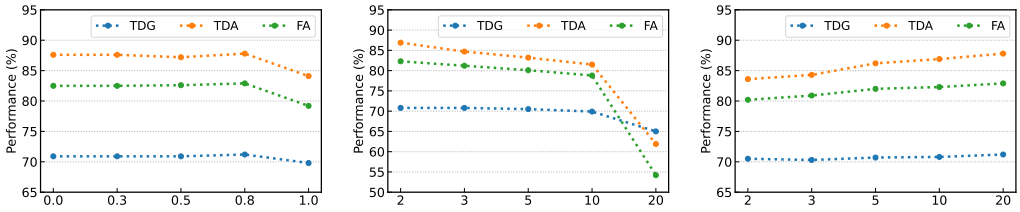
Table 7: Performance comparisons between ours and other methods on PACS in TDG, TDA, and FA under an additional domain order (besides the ones in Table 2 of the main text). We blue and bold **the best** average performance, and bold **the second best**.

Order		Metric	CoTTA	AuCID	SHOT	GSFDA	BMD	Tent	T3A	L2D	PDEN	SNR	PCL	EFDM	Ours
A → C → P → S	TDG	C	65.0	64.0	60.0	67.0	60.0	65.0	65.0	63.7	63.0	63.5	63.1	69.9	70.2
		P	97.3	93.7	97.1	96.6	96.8	97.3	97.3	97.3	97.0	97.0	96.4	95.7	96.7
		S	62.3	57.7	50.7	63.2	51.4	62.4	58.5	60.1	58.6	57.1	53.3	61.3	70.0
		Avg.	74.9	71.8	69.3	**75.6**	69.4	74.9	73.6	73.7	72.9	72.5	70.9	**75.6**	**79.0**
	TDA	A	96.1	96.3	92.7	97.3	92.7	96.1	96.1	96.0	95.9	96.0	96.3	94.9	95.9
		C	74.8	67.0	86.8	85.3	85.0	74.8	70.1	71.5	70.7	71.0	64.6	79.3	79.5
		P	97.0	93.1	99.0	98.2	98.8	97.0	97.3	98.0	97.5	98.0	99.1	95.5	99.1
		S	64.5	62.1	55.9	77.5	65.2	64.8	71.6	72.3	70.1	70.9	77.9	55.9	77.4
		Avg.	83.1	79.6	83.6	**89.6**	85.4	83.2	83.8	84.5	83.6	84.0	84.5	81.4	**88.0**
	FA	A	94.6	87.6	78.2	66.9	79.4	94.7	92.8	91.9	91.0	91.5	90.7	83.2	93.0
		C	74.8	74.7	71.8	52.2	69.0	74.8	68.7	70.9	70.3	71.0	66.4	71.1	74.9
		P	97.0	93.7	78.9	20.7	79.7	96.7	94.6	95.0	94.4	95.0	95.2	84.4	97.9
		Avg.	**88.8**	85.3	76.3	46.6	76.0	88.7	85.4	85.9	85.2	85.8	84.1	79.6	**88.9**

Table 8: 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		CoTTA	AuCID	SHOT	GSFDA	BMD	Tent	T3A	L2D	PDEN	SNR	PCL	EFDM	Ours
Re → Pa → In → Cl → Sk → Qu	TDG	Pa	78.2	77.4	75.4	80.0	75.4	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	24.1	30.7	25.1	26.1	25.0	25.0	26.0	24.0	29.7
		Cl	67.7	57.1	54.4	64.2	54.5	67.8	67.6	68.0	66.6	67.0	55.2	50.2	68.6
		Sk	64.1	60.5	57.8	58.5	53.9	64.3	62.6	63.3	62.5	62.8	55.7	42.0	64.7
		Qu	22.1	24.5	16.0	24.9	13.8	22.1	20.7	22.9	21.9	22.3	15.5	13.7	24.6
		Avg.	**52.6**	49.1	45.4	50.3	44.3	**52.6**	50.8	51.9	50.8	51.0	45.9	41.6	**53.5**
	TDA	Re	98.4	98.4	99.0	98.6	99.0	98.4	98.4	98.4	98.3	98.3	98.0	99.0	98.2
		Pa	73.3	77.4	80.3	82.4	79.1	73.3	80.7	76.6	75.4	75.5	70.9	64.4	75.0
		In	33.4	30.5	34.9	27.8	34.3	33.7	34.8	30.1	31.0	30.1	25.5	11.2	29.7
		Cl	68.8	61.1	67.0	67.2	58.1	69.1	72.5	70.7	69.1	70.0	56.0	34.1	70.3
		Sk	64.6	62.8	69.4	58.8	58.5	65.1	67.5	65.5	65.0	65.2	53.1	23.6	66.7
		Qu	22.8	31.4	35.0	29.6	38.4	23.7	26.7	27.0	30.1	28.8	21.5	14.0	69.0
		Avg.	60.2	60.3	**64.3**	60.7	61.2	60.6	63.4	61.4	61.5	61.3	54.2	41.1	**68.2**
	FA	Re	97.8	94.3	86.9	75.6	84.1	97.6	95.0	94.1	92.2	93.0	91.4	54.9	97.5
		Pa	73.4	78.2	74.5	53.3	74.8	73.5	75.7	71.3	70.0	71.1	64.0	37.8	74.3
		In	33.4	31.4	26.1	14.0	26.2	33.8	28.4	28.0	27.1	27.5	23.4	14.9	29.0
		Cl	68.8	61.0	56.3	37.4	50.1	69.6	64.5	62.8	61.1	61.9	54.1	28.9	71.5
		Sk	64.6	60.7	45.4	24.1	38.6	64.9	53.1	60.3	59.7	60.7	49.5	29.2	68.2
		Avg.	67.6	65.1	57.8	40.9	54.8	**67.9**	63.3	63.3	62.0	62.8	56.5	33.1	**68.1**

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



(a) r_{con} (b) r_{top} (c) r'_{top}

Figure 7: Sensitivity analysis of confidence threshold r_{con} in `RandMix`; r_{top} and r'_{top} in `T2PL`.

E Evaluations on More Settings

E.1 Standard Data Splitting of Domain Adaptation

We believe that using the entire set of training data to test the adaptation performance in target domains is not practical enough as the model may be adapted to highly overfit the training data and perform badly on unseen testing data. Thus, we choose to randomly split training and testing sets in the above experiments. To test whether `Twofer` can still work effectively in non-splitting settings, we also conduct experiments for further comparison with existing domain adaptation works. Moreover, we also use the evaluation protocol in Chen et al. (2022) to filter out a DomainNet-126 dataset, instead of using the full set of DomainNet in this section. The detailed experimental results are shown in Table 13 for Digits, Table 14 for PACS, and Table 15 for DomainNet-126. As shown in these tables, `Twofer` can still outperform other baselines in TDG, while achieving comparable

Table 9: Performance comparisons between ours and other baseline methods on Digits in average TDG, TDA, and FA under 10 different domain orders. We blue and bold **the best** performance, and bold **the second best**.

Metric	& Orders	SHOT+	SHOT++	Tent	AdaCon	EATA	L2D	PDEN	Ours
TDG	Order 1	67.9	70.9	**74.2**	71.0	72.2	70.5	68.1	**78.6**
	Order 2	79.5	79.6	74.1	75.5	72.7	**80.0**	78.8	**80.5**
	Order 3	67.5	67.7	70.9	68.1	71.4	**73.5**	72.0	**76.9**
	Order 4	48.6	47.0	52.4	51.8	55.0	**62.6**	60.3	**72.9**
	Order 5	53.6	54.9	52.5	53.0	55.2	63.5	**63.7**	**73.5**
	Order 6	73.0	74.7	77.5	76.4	78.0	**78.6**	78.0	**81.7**
	Order 7	76.4	75.9	77.8	75.0	78.0	**79.0**	78.5	**81.4**
	Order 8	74.4	72.7	78.1	76.6	**78.6**	77.7	76.6	**81.3**
	Order 9	36.1	38.6	42.4	41.5	44.9	63.2	**64.0**	**72.8**
	Order 10	33.1	32.7	45.1	44.0	44.3	**65.5**	62.9	**71.2**
	Avg.	61.0	61.5	64.5	63.3	65.0	**71.4**	70.3	**77.0**
TDA	Order 1	83.1	**89.4**	78.3	75.8	77.3	82.0	80.5	**89.8**
	Order 2	**93.9**	**95.6**	78.7	77.0	77.7	91.2	89.2	91.4
	Order 3	80.8	80.7	72.4	76.7	78.3	**87.6**	86.1	**89.0**
	Order 4	64.4	64.6	63.3	64.5	66.8	**78.7**	74.9	**87.6**
	Order 5	80.4	**84.8**	58.9	67.0	66.8	80.0	80.0	**87.1**
	Order 6	89.7	**93.2**	81.4	81.2	82.7	90.0	88.8	**91.0**
	Order 7	90.5	**92.6**	82.4	82.0	82.6	89.4	86.0	**91.6**
	Order 8	89.5	**92.1**	81.5	82.4	82.7	88.5	85.0	**90.3**
	Order 9	48.0	48.8	46.5	55.9	58.1	**74.1**	72.7	**86.7**
	Order 10	48.0	47.7	56.9	58.0	57.8	**80.7**	79.3	**87.8**
	Avg.	76.8	79.0	70.0	72.1	73.1	**84.2**	82.3	**89.2**
FA	Order 1	62.9	69.2	**76.8**	72.5	76.7	70.6	67.9	**83.7**
	Order 2	75.0	77.4	82.8	83.0	**83.3**	81.3	80.5	**87.9**
	Order 3	71.4	71.8	73.0	80.7	85.6	**85.7**	83.3	**90.5**
	Order 4	47.4	52.3	55.9	58.9	60.4	**72.3**	70.2	**82.6**
	Order 5	76.9	**78.8**	63.1	71.6	72.2	77.1	75.4	**85.3**
	Order 6	61.0	68.5	80.5	81.0	**81.1**	77.4	75.1	**84.9**
	Order 7	63.4	72.6	80.4	**81.5**	81.3	78.9	77.6	**85.6**
	Order 8	57.8	65.5	80.5	**81.7**	81.4	75.3	74.0	**84.2**
	Order 9	24.7	31.7	33.9	57.6	62.2	**72.6**	70.9	**85.4**
	Order 10	29.9	32.9	51.0	55.5	52.8	**74.3**	71.9	**82.9**
	Avg.	57.0	62.1	67.8	72.4	73.7	**76.6**	74.7	**85.3**

Table 10: Performance comparisons between ours and other baseline methods on PACS in average TDG, TDA, and FA under 10 different domain orders. We blue and bold **the best** performance, and bold **the second best**.

Metric & Orders		SHOT+	SHOT++	Tent	AdaCon	EATA	L2D	PDEN	Ours
TDG	Order 1	70.4	70.0	**74.9**	74.5	74.8	73.7	72.9	**79.0**
	Order 2	67.8	68.5	**75.1**	75.0	74.4	75.0	72.9	**77.4**
	Order 3	67.8	65.9	**77.0**	76.1	**76.9**	75.1	74.3	**76.9**
	Order 4	69.5	67.2	**79.4**	77.3	78.2	**79.4**	77.9	**80.5**
	Order 5	61.9	62.9	75.6	75.5	**79.7**	76.1	75.0	**78.2**
	Order 6	47.0	50.7	54.0	55.1	**57.3**	56.4	55.5	**59.5**
	Order 7	38.2	43.7	53.8	53.7	**54.3**	54.0	52.1	**56.0**
	Order 8	37.7	40.7	52.2	52.0	52.5	**52.7**	52.0	**54.0**
	Order 9	52.9	54.4	59.1	55.5	49.8	**61.6**	61.2	**72.8**
	Order 10	41.3	45.1	**60.5**	51.9	50.5	57.8	55.0	**69.3**
	Avg.	55.5	56.9	**66.2**	64.7	64.8	**66.2**	64.9	**70.4**
TDA	Order 1	85.1	**89.7**	83.2	85.1	86.4	84.5	83.6	**88.0**
	Order 2	**89.6**	**90.3**	83.7	83.0	84.1	84.9	82.7	88.2
	Order 3	**88.8**	**90.2**	83.8	82.7	83.6	83.4	80.9	86.6
	Order 4	**91.6**	**92.5**	87.3	88.5	89.6	84.1	84.5	89.0
	Order 5	85.2	85.8	**87.0**	**87.0**	86.7	81.2	81.1	**86.0**
	Order 6	81.9	**84.9**	72.8	75.3	79.2	75.3	74.1	**83.0**
	Order 7	68.4	71.0	**75.2**	75.0	74.3	73.0	72.5	**76.3**
	Order 8	65.8	69.4	**75.2**	73.9	74.3	70.3	70.0	**75.9**
	Order 9	**91.8**	**92.5**	71.1	71.0	72.7	83.1	82.1	90.3
	Order 10	76.4	**83.2**	70.0	71.3	70.3	74.9	72.5	**91.0**
	Avg.	82.5	**85.0**	78.9	79.3	80.1	79.5	78.4	**85.3**
FA	Order 1	73.6	77.7	88.7	87.9	**91.5**	85.9	85.2	**88.9**
	Order 2	73.7	**83.5**	79.4	80.6	**82.6**	80.0	78.7	79.9
	Order 3	81.7	77.9	**90.2**	**88.8**	86.6	84.1	79.4	87.6
	Order 4	78.4	78.0	84.3	**85.3**	**88.0**	79.7	76.8	85.1
	Order 5	83.3	88.1	88.5	**89.1**	**91.0**	78.3	77.7	84.9
	Order 6	82.2	**85.2**	79.6	80.7	83.4	73.9	72.9	**85.9**
	Order 7	65.9	73.8	**78.4**	77.8	76.3	76.2	75.7	**79.3**
	Order 8	59.4	74.0	75.1	**75.3**	74.8	72.7	71.9	**75.9**
	Order 9	84.9	**85.8**	74.9	75.0	75.8	79.4	77.8	**87.6**
	Order 10	73.1	**90.2**	73.3	74.5	74.8	74.3	72.5	**90.3**
	Avg.	75.6	**81.4**	81.2	81.5	82.5	78.5	76.9	**84.5**

Table 11: Performance comparisons between ours and other baseline methods on DomainNet in average TDG, TDA, and FA under 10 different domain orders. We blue and bold **the best** performance, and bold **the second best**.

Metric & Orders		SHOT+	SHOT++	Tent	AdaCon	EATA	L2D	PDEN	Ours
TDG	Order 1	47.3	47.0	**52.6**	51.8	52.3	51.9	50.8	**53.5**
	Order 2	52.6	51.1	32.2	52.7	**57.8**	**55.0**	54.1	**57.8**
	Order 3	53.2	54.3	56.3	56.0	**59.3**	54.0	52.6	**57.8**
	Order 4	40.7	42.8	**52.6**	52.2	51.6	49.6	48.9	**56.2**
	Order 5	49.2	51.7	**53.8**	53.5	52.6	51.7	51.5	**54.8**
	Order 6	34.8	35.5	34.0	35.0	34.4	**36.0**	35.7	**42.1**
	Order 7	24.2	27.0	35.3	33.8	**36.0**	32.1	30.7	**42.8**
	Order 8	59.9	60.0	62.1	61.5	62.8	**63.1**	62.7	**64.7**
	Order 9	58.5	60.1	63.4	62.6	**63.9**	61.9	60.8	**65.8**
	Order 10	55.4	59.9	42.1	59.0	**60.4**	58.8	57.0	**61.9**
	Avg.	47.6	48.9	48.4	51.8	**53.1**	51.4	50.5	**55.7**
TDA	Order 1	59.8	**71.9**	60.6	61.7	62.2	61.4	61.5	**68.2**
	Order 2	**69.7**	66.0	31.3	67.1	67.2	56.9	55.3	**72.5**
	Order 3	**73.4**	**73.2**	67.2	68.0	69.9	55.8	55.0	66.2
	Order 4	51.9	55.8	54.5	52.9	53.4	**58.2**	56.2	**61.2**
	Order 5	62.3	**71.2**	59.8	60.1	59.8	58.6	57.5	**64.4**
	Order 6	55.8	**56.9**	51.8	53.4	55.0	51.0	50.1	**56.0**
	Order 7	48.6	49.5	49.8	51.0	**53.8**	42.6	42.0	**59.0**
	Order 8	70.2	70.9	68.5	**71.2**	**72.0**	65.3	63.7	68.0
	Order 9	71.2	**72.7**	68.9	70.5	71.2	63.3	62.1	**78.7**
	Order 10	63.9	64.6	30.2	66.9	**70.1**	62.8	61.4	**67.5**
	Avg.	62.7	**65.3**	54.3	62.3	63.5	57.6	56.5	**66.2**
FA	Order 1	61.8	67.5	**68.6**	67.8	64.7	63.3	62.0	**68.1**
	Order 2	65.5	**69.9**	33.6	64.7	66.4	48.1	46.9	**70.8**
	Order 3	64.4	**68.7**	67.1	68.0	**69.1**	45.4	45.0	65.2
	Order 4	44.7	**58.2**	55.6	53.5	53.9	41.9	41.5	**56.8**
	Order 5	60.0	**65.2**	58.6	57.7	57.6	52.2	52.2	**61.7**
	Order 6	53.0	**58.3**	53.9	52.8	**56.5**	48.9	45.8	55.0
	Order 7	43.5	**53.9**	48.0	47.9	49.9	36.9	36.4	**56.8**
	Order 8	62.6	68.0	**73.7**	73.5	**73.9**	59.0	57.7	66.0
	Order 9	67.4	69.1	**76.5**	75.9	72.6	66.7	66.0	**76.4**
	Order 10	62.8	61.6	32.1	**66.2**	62.8	62.3	61.9	**67.7**
	Avg.	58.6	**64.0**	56.8	62.8	62.7	52.5	51.5	**64.5**

Table 12: Performance comparisons between ours and other methods on Digits, PACS, and DomainNet in FT and BT under default orders. We blue and bold **the best**, and bold **the second best**.

Order	Metric	CoTTA	AuCID	SHOT	GSFDA	BMD	Tent	T3A	L2D	PDEN	SNR	PCL	EFDM	Ours
Digits:	FT	39.2	34.6	35.5	18.0	39.5	45.4	35.1	**57.2**	51.3	54.7	31.1	12.1	**67.1**
Order 4	BT	-3.8	**0.0**	-19.1	-33.6	-16.4	**-0.6**	-16.1	-6.9	-5.2	-3.9	-9.9	-7.8	-3.7
Digits:	FT	21.2	22.6	8.2	18.7	19.0	35.8	24.7	**52.1**	48.6	49.5	25.2	7.0	**64.0**
Order 10	BT	-2.4	**-2.1**	-7.0	-28.8	5.3	**-0.2**	-15.4	-5.7	-7.8	-6.3	-11.2	-17.2	-4.4
Digits:	FT	53.8	52.5	53.3	45.5	53.1	**65.7**	46.7	51.1	47.2	48.5	64.9	25.8	**70.7**
Order 1	BT	**-3.2**	-8.2	-31.5	-21.4	-18.7	**-0.6**	-14.3	-14.3	-17.3	-13.2	-8.0	-19.3	-5.9
PACS:	FT	47.8	36.6	32.6	**52.1**	35.0	47.9	35.7	50.4	46.9	43.5	44.0	40.0	**53.6**
Order 6	BT	**-0.3**	5.4	-8.6	-68.5	-12.5	**-0.2**	-4.0	-8.9	-6.8	-7.0	-4.2	-6.5	-3.1
PACS:	FT	44.2	43.7	52.6	49.2	**53.9**	44.2	47.9	48.8	47.9	47.8	41.3	15.0	**61.9**
Order 9	BT	-0.1	**1.6**	-5.5	-14.7	-5.6	**0.0**	-4.6	-3.9	-5.7	-5.4	-1.5	-9.6	-0.2
PACS:	FT	61.2	61.4	52.7	58.7	54.5	61.3	59.2	**62.0**	61.5	58.8	57.0	60.4	**64.2**
Order 1	BT	**-0.5**	**-0.5**	-20.5	-67.1	-19.0	**-0.5**	-3.7	-4.5	-5.0	-5.2	-3.5	-19.7	**-3.2**
DomainNet:	FT	25.5	22.7	7.9	**26.7**	11.1	25.7	23.1	22.5	21.0	21.1	13.4	4.3	**32.0**
Order 7	BT	0.0	-0.8	-14.1	-15.7	-20.0	**0.1**	-5.0	-5.7	-5.8	-5.5	-7.0	-6.5	**2.4**
DomainNet:	FT	40.7	40.3	36.0	**41.3**	32.2	**40.9**	38.8	39.5	38.3	38.7	31.2	24.4	39.9
Order 1	BT	**-0.1**	-1.0	-24.7	-51.0	-19.8	**0.0**	-12.1	-8.3	-10.0	-9.1	-6.9	-15.2	-3.3

results in TDA and FA. Such results demonstrate that `Twofer` can work effectively in different data-splitting settings.

Table 13: Performance comparisons between ours and other baseline methods on Digits in TDG, average TDA and FA under standard data splitting of domain adaptation. We blue and bold **the best** average performance, and bold **the second best**.

Order	Metric		SHOT+	SHOT++	Tent	AdaCon	EATA	L2D	PDEN	Ours
		MM	53.3	53.3	53.1	53.2	53.3	73.6	68.4	86.8
MT		SN	21.4	22.1	29.7	30.0	29.3	35.4	35.6	43.7
MM	TDG	SD	34.4	33.5	38.0	41.7	42.5	52.6	54.1	67.5
SN		US	79.3	74.6	87.8	88.0	89.7	87.7	83.5	90.1
SD		Avg.	47.1	45.9	52.1	53.2	53.7	**62.3**	60.4	**72.0**
US	TDA	Avg.	63.8	62.8	59.9	64.9	65.0	**77.6**	75.3	**86.8**
	FA	Avg.	46.5	51.0	50.8	59.0	58.8	**72.3**	70.0	**82.3**
		SD	30.5	30.5	37.5	37.5	36.3	61.3	60.2	68.6
US		SN	19.5	19.6	27.2	27.5	26.4	52.7	51.5	62.0
SD	TDG	MM	27.5	27.4	42.1	41.8	40.2	64.5	60.8	68.7
SN		MT	49.0	49.5	76.7	78.2	73.7	83.4	82.7	87.3
MM		Avg.	31.6	31.7	47.4	46.3	44.2	**65.5**	63.8	**71.7**
MT	TDA	Avg.	45.2	42.9	57.3	58.3	57.0	**82.9**	80.3	**87.1**
	FA	Avg.	26.7	32.5	51.5	52.0	52.4	**74.1**	72.0	**82.8**
		MT	75.7	70.7	70.6	70.4	66.9	70.0	70.0	72.7
SN		MM	48.4	56.2	52.2	53.3	50.7	65.6	59.2	69.2
MT	TDG	SD	59.3	63.6	88.0	88.0	89.6	77.6	76.6	84.2
MM		US	81.5	82.6	77.7	78.5	78.2	76.1	71.6	87.8
SD		Avg.	66.2	68.3	72.1	**72.6**	71.3	72.3	69.4	**78.5**
US	TDA	Avg.	84.0	**87.5**	74.5	77.4	76.8	85.9	81.9	**89.7**
	FA	Avg.	60.0	67.1	72.8	75.2	**76.2**	75.2	71.4	**83.8**

E.2 Stationary Domain Adaptation

While our work primarily focuses on continual domain shift, here we provide results of stationary domain adaptation for further comparison. In this setting, we consider the combination of one single source domain and one single target domain, and consider the generalization performance on

Table 14: Performance comparisons between ours and other baseline methods on PACS in TDG, average TDA and FA under standard data splitting of domain adaptation. We blue and bold **the best** average performance, and bold **the second best**.

Order	Metric		SHOT+	SHOT++	Tent	AdaCon	EATA	L2D	PDEN	Ours
P A C S	TDG	A	67.5	67.5	71.6	71.0	71.4	67.5	67.0	73.8
		C	39.2	40.4	48.5	50.6	50.6	49.2	47.8	53.0
		S	38.9	42.1	48.6	50.0	50.2	52.7	52.5	63.5
		Avg.	48.5	50.0	56.2	57.2	**57.4**	56.5	55.8	**63.4**
	TDA	Avg.	83.1	**85.3**	75.7	78.6	79.2	75.5	74.0	**86.2**
	FA	Avg.	79.6	**84.3**	81.4	80.7	83.5	71.0	70.9	**86.4**
S C A P	TDG	C	36.3	36.3	58.3	59.2	46.9	63.7	63.1	69.6
		A	47.0	46.6	50.4	51.7	43.8	52.0	50.8	69.7
		P	76.1	75.1	54.2	61.1	53.8	70.2	68.9	80.7
		Avg.	53.1	52.7	54.3	57.3	48.2	**62.0**	60.9	**73.3**
	TDA	Avg.	**91.5**	**92.2**	69.6	77.8	73.0	83.0	82.5	89.6
	FA	Avg.	86.5	**88.8**	74.1	79.0	76.6	78.0	78.3	**87.7**
A C P S	TDG	C	59.6	59.6	64.8	64.5	63.8	63.7	63.3	66.1
		P	95.1	95.8	97.6	97.1	97.3	97.3	97.3	97.3
		S	53.4	55.9	64.2	64.0	64.1	61.0	60.5	67.0
		Avg.	69.4	70.4	**75.5**	75.2	75.1	74.0	73.7	**76.8**
	TDA	Avg.	86.7	**90.4**	84.0	85.8	86.7	84.5	83.7	**89.5**
	FA	Avg.	70.0	78.6	89.5	**90.7**	**91.4**	85.6	85.2	87.8

Table 15: Performance comparisons between ours and other baseline methods on DomainNet-126 in TDG, average TDA and FA under standard data splitting of domain adaptation. We blue and bold **the best** average performance, and bold **the second best**.

Order	Metric		SHOT+	SHOT++	Tent	AdaCon	EATA	L2D	PDEN	Ours
Sk Cl Pa Re	TDG	Cl	58.2	58.2	60.5	57.8	51.9	52.0	51.3	62.5
		Pa	54.0	54.1	53.5	58.5	59.9	53.2	51.5	61.0
		Re	62.6	63.0	58.6	59.0	56.5	52.7	53.6	63.8
		Avg.	58.3	**58.4**	57.6	**58.4**	56.1	52.6	52.1	**62.4**
	TDA	Avg.	**69.2**	**69.2**	64.5	65.3	61.3	57.2	55.9	**69.9**
	FA	Avg.	61.7	**65.0**	62.8	64.0	55.8	50.9	50.5	**65.8**
Re Pa Cl Sk	TDG	Pa	63.7	63.7	60.5	58.3	61.4	57.7	56.0	62.3
		Cl	54.5	54.9	56.6	57.0	57.4	55.4	54.0	57.0
		Sk	50.1	50.6	53.4	51.5	52.8	53.2	51.8	56.0
		Avg.	56.1	56.4	56.9	55.6	**57.2**	55.4	53.9	**58.5**
	TDA	Avg.	69.0	**70.5**	66.4	67.7	67.3	63.4	63.3	**70.0**
	FA	Avg.	64.7	**69.5**	**68.4**	66.4	65.7	60.7	58.9	**69.5**

the target domain accuracy after adaptation. In such situation, we remove the replay memory, the distillation loss term (Eq. 10) and the second term in the nominator of $\mathcal{L}_{\text{PCA}}$ (Eq. 9), as they force the model to achieve forgetting alleviation for previous domains and may impede the adaptation to the target domain. We conduct our stationary domain adaptation in PACS and DomainNet-126, and the results after adaptation are shown in Tables 16 and 17. For PACS, `Twofer` performs the best in most cases and achieves 4.4% higher average accuracy than the second best. As for DomainNet-126, `Twofer` also shows competitive results when compared with other baselines.

Table 16: Comparison of domain adaptation for stationary source-target pairs on PACS. We blue and bold **the best** average performance, and bold **the second best**.

Method	P→A	P→C	P→S	A→P	A→C	A→S	C→P	C→A	C→S	S→P	S→A	S→C	Avg.
SHOT	87.2	85.4	33.8	99.3	87.8	44.4	99.0	91.9	64.9	53.5	84.9	89.8	76.8
SHOT+	90.4	87.9	38.8	99.5	88.8	41.9	98.9	92.6	67.2	52.2	84.3	89.5	77.7
SHOT++	93.1	89.1	38.8	99.6	90.1	41.3	99.1	93.6	68.2	53.1	87.2	91.7	78.4
Tent	77.0	64.9	55.5	97.4	76.7	63.7	95.4	82.8	73.5	55.3	59.1	68.0	72.4
AdaCon	88.2	85.3	56.6	99.5	86.8	67.7	97.7	91.5	75.0	61.3	82.7	75.3	**80.6**
EATA	79.2	72.7	59.0	97.2	80.0	72.7	95.7	84.8	79.7	55.3	60.0	68.7	75.4
Ours	92.5	83.2	62.4	99.5	88.0	59.0	98.9	92.6	64.8	99.2	92.8	86.7	**85.0**

Table 17: Comparison of domain adaptation for stationary source-target pairs on DomainNet-126. We blue and bold **the best** average performance, and bold **the second best**.

Method	Re→Cl	Re→Pa	Pa→Cl	Cl→Sk	Sk→Pa	Re→Sk	Pa→Re	Avg.
SHOT	68.2	67.2	68.7	58.5	62.6	59.4	80.1	66.4
SHOT+	68.7	67.9	69.2	59.5	63.1	60.8	79.3	66.9
SHOT++	70.4	69.5	70.8	61.1	65.1	62.2	80.7	**68.5**
Tent	58.2	62.7	58.3	57.7	58.4	56.9	70.1	60.3
AdaCon	69.0	68.7	68.6	57.5	64.3	61.5	80.0	67.1
EATA	60.1	65.1	57.8	53.3	56.0	59.7	70.0	60.3
Ours	69.4	68.4	69.0	60.5	62.5	62.0	79.9	**67.4**