Multi-Domain Incremental Learning for Face Presentation Attack Detection

Keyao Wang\textsuperscript{*1}, Guosheng Zhang\textsuperscript{*1}, Haixiao Yue\textsuperscript{*1}, Ajian Liu\textsuperscript{2}, Gang Zhang\textsuperscript{1}, Haocheng Feng\textsuperscript{1}, Junyu Han\textsuperscript{1}, Errui Ding\textsuperscript{1}, Jingdong Wang\textsuperscript{1}

\textsuperscript{1}Department of Computer Vision Technology (VIS), Baidu Inc
\textsuperscript{2}CBSR&MAIS, Institute of Automation, Chinese Academy of Sciences (CASIA)
\{wangkeyao, zhangguosheng, yuehaixiao, zhanggang03, fenghaocheng, hanjunyu, dingerrui\}@baidu.com, ajianliu92@gmail.com, wanglingdong@outlook.com

Abstract

Previous face Presentation Attack Detection (PAD) methods aim to improve the effectiveness of cross-domain tasks. However, in real-world scenarios, the original training data of the pre-trained model is not available due to data privacy or other reasons. Under these constraints, general methods for fine-tuning single-target domain data may lose previously learned knowledge, leading to the issue of catastrophic forgetting. To address these issues, we propose a Multi-Domain Incremental Learning (MDIL) method for PAD, which not only learns knowledge well from the new domain but also maintains the performance of previous domains stably. To this end, we propose an Adaptive Domain-specific Experts (ADE) framework based on the vision transformer to preserve the discriminability of previous domains. Moreover, we present an asymmetric classifier to keep the output distribution of different classifiers consistent, thereby improving the generalization ability. Extensive experiments show that our proposed method achieves state-of-the-art performance compared to prior methods of incremental learning. Excitingly, under more stringent setting conditions, our method approximates or even outperforms DA/DG-based methods.

Introduction

In recent years, face recognition (FR) techniques have been widely exploited in different application scenarios, such as smartphone login, financial payment, access control, etc. While FR systems may suffer from various presentation attacks (PAs), e.g., printed photos, video replay, and 3D masks, which seriously threaten the credibility of facial information and bring great challenges to public security management. To address these issues, various PAD methods have been proposed, including the hand-crafted methods (Freitas Pereira et al. 2012; Patel, Han, and Jain 2016) and the deep learning methods based on auxiliary supervision (Atoum et al. 2017; Liu et al. 2019).

Although these methods have achieved promising results under intra-dataset scenarios, they neglect domain discrepancy across different domains and may encounter performance degradation when adapting to new domains. To mitigate this problem of domain shift, recent studies introduce domain adaptation (DA) (Jia et al. 2021; Wang et al. 2019) and domain generalization (DG) (Zhou et al. 2021; Chen et al. 2021) into the field of PAD. As shown in Figure 1, DA-based methods focus on transferring performance on the unlabeled target domain, but they may affect the performance of the source domain. In contrast, DG-based methods aim to achieve out-of-distribution generalization by using multi-source domain data for model learning simultaneously.

In general, DA/DG-based methods address the domain transfer problem. But in most practical scenarios, part or all of the data in the source domain cannot be accessed by data privacy. MDIL-based methods solve the problem of mitigating the catastrophic forgetting of the original domain data information when only using the target domain data for training. As shown in the third column in Figure 1, MDIL aims to train a single model on sequential non-stationary domains without revisiting the previous domain data. In practical PAD application, since an initially trained model fails to identify novel attack types from unknown domains, a straightforward way to solve this problem is to retrain the model from scratch on both old and new domain data. However, model retraining is computationally costly because it requires storing large amounts of previous domain data.

Figure 1: The comparison of different methods. DA-based methods align the unlabeled target domain with source domains. DG-based methods eliminate dependency on the target domain and test directly on the unseen target domain. MDIL-based methods aim to obtain satisfactory results from previous domain data without revisiting old domain data.
data. Therefore, it is likely that MDIL may be more suitable for PAD scenarios with a potential domain transfer, as it can reduce the data storage requirements of model retraining.

However, there are two main challenges with MDIL-PAD. (1) **Domain gap.** Due to the diversity of data collection environments and the ongoing emergence of novel attack types, domain incremental learning task for PAD simultaneously couples the challenges of domain incremental learning (DIL) and class incremental learning (CIL). Larger domain gaps lead to more severe forgetting. (2) **Domain agnostic.** A common and practical constraint in DIL is that domain indexes are not provided for inference. So we can not tackle this problem by training a separate, independent model for each domain. L2P(Wang et al. 2022b) and S-Prompts(Wang, Huang, and Hong 2022) discarded the need for task indexes by designing a query-key mechanism or clustering strategy to automatically select relevant prompts for each instance. However, these strategies involve additional computational overhead. They will be discussed in appendix B.

To this end, we propose a novel multi-domain incremental framework for PAD that leverages the multi-experts to instruct the model adaptively. In our method, the Adaptive Domain-specific Experts (ADE) blocks maintain the isolation of domain-specific parameters and the sharing of domain-invariant parameters, which is beneficial for mitigating catastrophic forgetting. Moreover, we first design a flexible Instance-wise Router (IwR) module for selecting the relevant expert. During inference, the domain-agnostic instances can obtain the associated domain index based on the similarity with the domain centers. Then, the appropriate index guides the instance into the ADE blocks with the corresponding expert branch by gating mechanism. Considering the feature separation of spoof samples among different domains, we consolidate the multi-classifiers into a unified asymmetric classifier. This design helps to keep the consistency of the predicted probability distribution (PPD) of live samples from different classifiers. Furthermore, the proposed method can adapt to the unknown domain and operate for the dynamically added domains in end-to-end training.

- We propose an adaptive domain-specific experts framework for PAD in an incremental update pattern, which can maintain satisfactory results in both previous and new domains. Besides, an innovative IwR is designed to deal with domain-agnostic instances.
- Considering the sparsity and discreteness of spoof samples, an asymmetric classifier is designed to solve the problem of PPD of live samples inconsistent in open append domains.
- Extensive experiments are conducted on widely used benchmark datasets, which demonstrate the effectiveness of the proposed method and also illustrate that our method achieves state-of-the-art performance.

**Related Work**

**Face Presentation Attack Detection**

Face PAD aims to detect spoof attacks of various types and improve the security of face recognition systems. With the development of deep learning, many researchers (Wang et al. 2020b; Zhang et al. 2021) utilize the convolutional neural network to enhance the extraction ability of face representation. Considering the lack of sufficient supervision, some approaches (Liu, Jourabloo, and Liu 2018; Liu et al. 2019) design different auxiliary supervisions to improve the performance of classification, such as depth map (Yu et al. 2021), reflection map (Kim et al. 2019), and PPG signals (Hu et al. 2021). These works achieve promising results with intra-data but neglect the domain gap across different domains.

To achieve better generalized performance in the target data, DA (Jia et al. 2021; Li et al. 2018; Wang et al. 2019, 2020a) and DG (Chen et al. 2021; Jia et al. 2020; Liu et al. 2021a,b; Shao et al. 2019) are introduced into the PAD area. SDA (Wang et al. 2021a) designs a domain adaptor to utilize the unlabeled test domain data at inference. GDA (Zhou et al. 2022b) stylizes the unlabeled target data to the source-domain style via image translation for feature alignment. In contrast, DG aims to learn a generalized representation from multi-source domains, independent of the target domain. SSDG (Jia et al. 2020) leverages asymmetric triplet loss and adversarial learning to regulate live samples and distinguish spoof samples from source domains. SSAN (Wang et al. 2022a) designs style assembly layers to combine indistinguishable content features and domain-specific style features.

Nevertheless, the above methods can effectively narrow the gap between the source domain and the target domain, they only focus on the transfer performance of the target domain and neglect the source domain.

**Incremental Learning**

In the application of multiple domains, the training data of the original domain are generally inaccessible due to data privacy, and when learning a new domain, the model may catastrophically forget what it learned previously(Kirkpatrick et al. 2017). Incremental Learning (IL) is introduced to alleviate the problem of the long-studied pattern. There are mainly three categories: regularization-based, replay-based, and parameter isolation-based methods. Regularization-based methods (Zenke, Poole, and Ganguli 2017; Aljundi et al. 2018) can consolidate these weights of previous tasks according to their importance. Replay-based techniques store previous experience by implicitly generating replays (Chenshen et al. 2018; Ostapenko et al. 2019) or explicitly displaying original samples (Hou et al. 2019; Wu et al. 2019) to preserve the representation ability of previous domains. As for parameter isolation methods (Mallya and Lazebnik 2018; Rebuffi, Bilen, and Vedaldi 2018), allocating specific model parameters to each task maintains maximal stability by fixing the parameter subsets of previous tasks. LwF (Li and Hoiem 2017) leverages knowledge distillation to maintain the performance of older tasks after adding new tasks. Inspired by the VPT (Jia et al. 2022), some prompting methods (Douillard et al. 2022; Wang, Huang, and Hong 2022) are proposed to alleviate the performance degradation of previous domains by introducing a small number of parameters. L2P (Wang et al. 2022b) utilizes the prompt pool to store encoded knowledge and...
presumes the prompt tokens into the input tokens. In the terms of PAD, the article (Pérez-Cabo et al. 2020) introduces an IL framework into PAD for the first time, which follows the few-shot learning paradigm. For novel face spoof attack types, the method (Rostami et al. 2021) is proposed to treat them as anomalies and correctly classify them via experience replay. (Guo et al. 2022) propose the FAS-wrapper, which employs a regularization-based approach to facilitate knowledge transfer from pre-trained models for MDL. (Cai et al. 2023) propose the rehearsal-free method for Domain Continual Learning of FAS, which deals with catastrophic forgetting and unseen DG problems simultaneously.

Unlike these works, we design a buffer-free dynamic IL framework containing the learnable domain-specific information to mitigate the catastrophic forgetting of previous domains. Furthermore, we propose a multi-expert framework for MDIL, which preserves the parameter independence to learn the corresponding domain knowledge adaptively.

Methodology

In multi-domain incremental learning, \( T \) domains are presented sequentially, defined as \( \mathcal{D} = \{( \mathcal{D}_1, \cdots, \mathcal{D}_T )\} \). Learning follows incremental steps, where each step involves learning an existing model for the current domain. The \( t \)-th input domain is defined as \( \mathcal{D}_t = \{ (x_i^t, y_i^t) \}_{i=1}^{n_t} \), where \( x_i^t \in \mathcal{X} \) represents the input sample, \( y_i^t \in \mathcal{Y} \) is the corresponding label, and \( n_t \) is the number of samples in \( \mathcal{D}_t \). We aim to train a single PAD model \( \mathcal{M}(\mathcal{X}) = \mathcal{Y} \), that predicts \( y = \mathcal{M}(x) \) for any test sample \( x \). At any learning step \( t \), data from previous domains \( \{( \mathcal{D}_1, \cdots, \mathcal{D}_{t-1} )\} \) are not available for training.

Overview

We propose the Adaptive Domain-specific Experts (ADE) framework for coping with the performance degradation in previous domains. As shown in Figure 2, it consists of three key components: the ADE block with multiple experts, the Instance-wise Router (IwR), and an asymmetric classifier.

Specifically, we define an image sample from datasets as \( x \in \mathbb{R}^{H \times W \times C} \), where \( H, W \) represent the length and width of the image, respectively, and \( C \) is the number of channels. The image is first reshaped and split into a sequence of 2D patches \( x_p \in \mathbb{R}^{L \times S^2 \times C} \), where \( S \) is the side length of each image patch and \( L = HW/S^2 \) represents the number of patches. At the start of training, the sequence of embeddings \( x_p \) are first fed into the linear projection to generate the image tokens \( x_{p} \in \mathbb{R}^{L \times S^2 \times C} \rightarrow x_t \in \mathbb{R}^{(L+1) \times D} \), where \( D \) is the embedding dimension, and the extra dimension on \( L \) is the corresponding class token.

Based on ViT (Dosovitskiy et al. 2020), our network consists of total \( N \) transformer blocks. We divide it into the first \( M \) blocks as shared encoder \( f_s \) and the last \( (N - M) \) blocks as expert decoder \( f_e \), while the whole \( f_s \) are designed by ADE blocks. Next, the image tokens \( x_t \) from \( t \)-th domain are sent to the shared encoder \( f_s \) with parameters \( W_{s_1} \):

\[
g_t = f_s(x_t; W_{s_1}),
\]

the first embedding in \( g_t \in \mathbb{R}^{(L+1) \times D} \) is defined as \( g_{cls} \in \mathbb{R}^D \). Then, we propose the IwR module to predict the gate signal to choose the most appropriate domain-specific expert branch for ADE blocks when testing the domain-agnostic instance. And we update ADE blocks \( f_e \) with parameters \( W_{s_2}, W_{e_t} \) of the corresponding expert branch, which keeps the different domains of knowledge independent.

\[
z_t = f_e(g_t, t; W_{s_2}, W_{e_t}),
\]

where \( z_t \in \mathbb{R}^{(L+1) \times D} \) is the output feature of the \( t \)-th expert branch. After getting the feature \( z_t \), we split the first embedding of \( z_t \) as \( z_{cls} \in \mathbb{R}^D \). The final result \( p_t \) is generated by the proposed asymmetric classifier network \( f_c \) with

Figure 2: The overall framework of our proposed method. Our network is divided into the first \( M \) standard vision transformer blocks as shared encoder \( f_s \) and the last \( (N - M) \) ADE blocks as expert decoder \( f_e \). The processing steps are as follows: 1) Images from \( t \)-th domain are first embedded to tokens and then processed by the \( f_s \) for generating \( g_t \). 2) The first embedding of \( g_{cls} \) is fed to IwR for learning the domain center \( c_t \) in the training phase and predicting the gate signal to choose the appropriate expert branch in ADE blocks at test time. 3) The result is predicted by the proposed asymmetric classifier network \( f_c \).
Compared to the original ViT blocks, we introduce the Adaptive Domain Expert (ADE) Block. The advantage of the proposed framework is described in Algorithm 1.

Inference stage. In this section, we describe the designed ADE blocks and the instance-wise router in detail. The whole process of the proposed framework is described in Algorithm 1.

### Adaptive Domain-Specific Experts

To mitigate the performance degradation of previous domains, we design the domain-specific experts framework. This framework can keep the parameter independent among different domains while sharing the knowledge between similar domains. Besides, the proposed framework can adaptively deal with the domain-agnostic sample in the inference stage. In this section, we describe the designed ADE blocks and the instance-wise router in detail. The whole process of the proposed framework is described in Algorithm 1.

**ADE Block.** Compared to the original ViT blocks, we dynamically extend the original MLPs to multi-domain-specific experts, which have the same architecture $E_d(x) = MLP(x)$. Specifically, we define the sharing of parameters and the $t$-th expert parameters in the ADE block as $W_{s_2}, W_{s_t}$, respectively. This design decouples the network parameters into a set of domain-invariant and domain-specific parameters. Thus, they can learn domain-specific knowledge separately without interference while leveraging domain-invariant knowledge to consolidate the generalization capability and alleviate catastrophic forgetting. Equipped with the learnable gating mechanism, the proposed ADE block can adaptively choose the domain-related expert in inference. The output $z_t$ is obtained by Eq. 2.

When learning a new domain $D_{t+1}$ in step $t+1$, we keep the previously trained parameters $W_s (1 \leq j \leq t)$ frozen and update the sharing of parameters $W_{s_2}$ and the newly added domain parameters $W_{s_{t+1}}$ according to the process:

$$z_{t+1} = f_e(g_{t+1}, t + 1; W_{s_2}, W_{s_{t+1}}).$$

Note that the domain index (gate signal) $t$ is known during the training phase, while it is unknown at test time.

**Instance-Wise Router.** As mentioned above, we employ isolated experts to maintain knowledge independence across different domains, which brings up a critical question: how to automatically choose domain-related experts during the inference phase? To deal with this problem, we design a learnable gating mechanism, Instance-wise Router (IwR), to assign each domain-agnostic instance to a domain-related expert branch.

During the training phase on the current incremental domain $D_t$, we aim to find a domain center that aligns with the latent feature $g_{cls}$ in $\mathbb{R}^D$. We assume that images from the same domain have a similar distribution on the high-level feature space projected by a well-pretrained network. Specifically, we denote the domain center as $c_t \in \mathbb{R}^D$ and the instance-wise router network as $f_r$. As shown in Figure 2, the corresponding domain center $c_t$ is expected to be close to the feature center of the current domain. Thus, we achieve this goal by minimizing a cross-entropy loss:

$$L_{gate} = -\frac{1}{n_t} \sum_{i=1}^{n_t} \log \frac{e^{sim(g_{cls}, c_t)}/\tau}{1 - e^{sim(g_{cls}, c_t)}/\tau},$$

where $n_t$ is the number of $t$-th domain images, and $\tau$ is a temperature coefficient. Besides, $sim(a, b) = \frac{a \cdot b}{\|a\| \|b\|}$ is the cosine similarity calculation.

### Asymmetric Classifier

Considering the design of the classifier for multi-branch network architecture, one way to avoid catastrophic forgetting is to construct multiple separate, independent classifiers $\{[\theta_{t_1}, \theta_{s_1}], \ldots, [\theta_{t_F}, \theta_{s_F}]\}$, and each classifier corresponds to a domain-related expert branch. However, outputs of the PAD model are commonly expected to be predicted probability of live $p_l = f_c(z_{cls}, t; \theta_{t})$, differentiated by a threshold. However, this approach makes it hard to ensure consistency of the predicted probability distribution (PPD) across different classifiers. Thus, the setting of using the same threshold for multiple classifiers easily causes poor performance. Another alternative approach is to design a shared single classifier. Although it gets rid of inconsistencies of PPD, it still suffers from catastrophic forgetting.

Inspired by (Jia et al. 2020), the feature distribution of the live samples is compact while that of the spoof samples present is dispersed across domains. It suggests that the domain gap is mainly reflected in the spoof samples. Therefore, we consolidate the multi-classifiers into a unified asymmetric classifier. Specifically, we use a shared class center $\theta_{t}$ for live samples across different domains, while the spoof samples from different domains are regarded as an individual category. As shown in Figure 3, the asymmetric classifier $\{[\theta_{t_1}, \theta_{s_1}], \ldots, [\theta_{t_F}, \theta_{s_F}]\}$ contains $T + 1$ categories. The predicted probability of live $p_l = f_c(z_{cls}; \theta_{t})$ is domain-independent, and the PPD significantly consistent due to the unified decision boundaries. For training data with domain index $t$, the output can be formulated by Eq. 3.

---

**Algorithm 1: The Procedure of MDIL-PAD.**

<table>
<thead>
<tr>
<th>Require: Sequential domain dataset $D_t = {(x^i_t, y^i_t)}_{i=1}^{n_t}$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: [ \text{// Training:} ]</td>
</tr>
<tr>
<td>2: for $t = 1, \ldots, T$ do</td>
</tr>
<tr>
<td>3: \quad Reload $W_{s_1}, W_{s_2}, \theta_t$ from $t - 1$ step. Random initialize $W_{s_2}, \theta_{s_2}, \text{Freeze } W_{s_1}$.</td>
</tr>
<tr>
<td>4: for $i = 1, \ldots, \text{MaxEpoches}$ do</td>
</tr>
<tr>
<td>5: \quad Forward pass $x_t$ via $W_{s_1}$, Eq. 1</td>
</tr>
<tr>
<td>6: \quad Compute domain-specific loss $L_{gate}$ by Eq. 5</td>
</tr>
<tr>
<td>7: \quad Forward pass $g_t$ via $W_{s_2}, W_{s_2}$, Eq. 2</td>
</tr>
<tr>
<td>8: \quad Forward pass $x_{cls}$ via $\theta_t, \theta_{s_2}$, Eq. 3</td>
</tr>
<tr>
<td>9: \quad Compute classifier loss $L_{cls}$ by Eq. 6</td>
</tr>
<tr>
<td>10: \quad Compute total loss via $L_{all}$ Eq. 7</td>
</tr>
<tr>
<td>11: end for</td>
</tr>
<tr>
<td>12: Discard training data $D_t$.</td>
</tr>
<tr>
<td>13: end for</td>
</tr>
<tr>
<td>14: [ \text{// Inference:} ]</td>
</tr>
<tr>
<td>15: Forward pass $x_t$ via $W_{s_1}$, Eq. 1</td>
</tr>
<tr>
<td>16: Forward pass $g_{cls}$ via trained Router $f_r$, Eq. 8</td>
</tr>
<tr>
<td>17: Forward pass $g_t$ via $W_{s_2}, W_{s_2}$, Eq. 2</td>
</tr>
<tr>
<td>18: Forward pass $z_{cls}$ via $\theta_t, \theta_{s_2}$, Eq. 3</td>
</tr>
</tbody>
</table>

Note that when training on different domains, we keep the shared encoder $f_s$ frozen, only fine-tuning the expert decoder $f_e$ and the asymmetric classifier network $f_c$. 

$\theta_{t_1}, \theta_{s_1}$: 

$$p_t = f_e(z_{cls}, t; \theta_{t_1}, \theta_{s_1}).$$

(3)
which results in great distribution discrepancies. These datasets contain face samples from different capture devices, illumination, background, and spoof attack types, (I for short), MSU-MFSD (Wen, Han, and Jain 2015) (M for short), CASIA-MFSD (Zhang et al. 2012) (C for short), Idiap ReplayAttack (Chingovska, Anjos, and Marcel 2012) (I for short), and SiW (Liu, Jourabloo, and Liu 2018) (S for short). (S for short).

We evaluate the effectiveness of our method on the following observations. (1) Compared with other general methods, the proposed approach effectively mitigates the performance degradation of previous domains. Based on FT, our method obtains significant improvements of 15.40% and 18.83% on dataset M in step 2 and step 3, respectively. In addition, the newly supplemented domain in different steps also maintains optimal performance. (2) Compared with other IL-based methods, our method achieves the minimum average drop $\Delta m\%$ on previous domains under various steps. And in longer steps like steps 3 and 4, our method outperforms other methods, showing that our proposed method can retain different domains of knowledge more persistently by constructing domain-specific experts.

### Results of 4-Domain Incremental Settings
In this section, we first construct a model on dataset M in step 1. Then the same model is optimized by dataset C in step 2. Subsequently, we fine-tune the model on new datasets with the domain shift I and O in steps 3 and 4 and evaluate the results from previous domains. As shown in Table 1, we make the following observations. (1) Compared with other IL-based methods, our method achieves the minimum average drop $\Delta m\%$ on previous domains under various steps. And in longer steps like steps 3 and 4, our method outperforms other methods, showing that our proposed method can retain different domains of knowledge more persistently by constructing domain-specific experts.

### Results of Cross-Attack-Type Incremental Settings
In this section, we focus on the larger domain gap caused by the absence of overlapping attack types in sequence domains. Specifically, we exclude Print/Replay type attacks from SiW.

#### Implementation Details
We train our method using the Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9, an initial learning rate of 0.01, and a batch size of 48. Input images are resized to 224x224. To compare fairly, we adopt the same network ViT-B/16 (Dosovitskiy et al. 2020) across all methods in Table 1. Specifically, we reimplement LwF (Li and Hoiem 2017) by utilizing the ViT. For DyTox (Douillard et al. 2022) and S-iPr (Wang, Huang, and Hong 2022), we use their official implementations by tuning their block number to 12 since the original paper set it to 6, and we maintain the same settings for all experiments.

#### Evaluation Metrics
Following the work of (Jia et al. 2020), we utilize the Half Total Error Rate (HTER) and Area Under Curve (AUC) to evaluate the performance. To quantify the overall performance, similar to (Kanakis et al. 2020), we define the average decrease in HTER of each task $t$ with respect to the multi-domain performance $b$ as $\Delta m\%$. For the model $q_t$, the $\Delta m\%$ is calculated by:

$$\Delta m\% = \frac{1}{T} \sum_{t=1}^{T} HTER_{b,t} - HTER_{q,t}. \quad (9)$$

Lower $\Delta m\%$ indicates less performance drop in previous domain.

### Comparison to the State-of-the-Art Methods
To validate the effectiveness of our proposed approach, we compare it with other IL baselines. Joint training (JT) is trained with all domain datasets and considered an upper-bound performance. Fine-tuning (FT) is optimized on the new domain without any explicit effort to mitigate forgetting, which is considered a lower-bound performance. Feature extraction (FE) freezes all backbone parameters and just trains the fully connected (FC) layer of a new domain.

### Datasets
We evaluate the effectiveness of our method on five PAD datasets: OULU-NPU (Boulkenafet et al. 2017) (O for short), CASIA-MFSD (Zhang et al. 2012) (C for short), Idiap ReplayAttack (Chingovska, Anjos, and Marcel 2012) (I for short), MSU-MFSD (Wen, Han, and Jain 2015) (M for short), and SiW (Liu, Jourabloo, and Liu 2018) (S for short). These datasets contain face samples from different capture devices, illumination, background, and spoof attack types, which results in great distribution discrepancies.

### Experiment

**Datasets.** We evaluate the effectiveness of our method on five PAD datasets: OULU-NPU (Boulkenafet et al. 2017) (O for short), CASIA-MFSD (Zhang et al. 2012) (C for short), Idiap ReplayAttack (Chingovska, Anjos, and Marcel 2012) (I for short), MSU-MFSD (Wen, Han, and Jain 2015) (M for short), and SiW (Liu, Jourabloo, and Liu 2018) (S for short). These datasets contain face samples from different capture devices, illumination, background, and spoof attack types, which results in great distribution discrepancies.

**The first category refers to the live face, and the other categories are face spoof samples of various domains. The unified classifier $f_c$ is optimized by the cross-entropy loss function:**

$$L_{\text{cls}} = -\frac{1}{n_t} \sum_{i=1}^{n_t} y_i^t \log(p_i^t) + (1 - y_i^t) \log(1 - p_i^t). \quad (6)$$

The total training loss function contains two parts:

$$L_{\text{all}} = L_{\text{cls}} + \alpha L_{\text{gate}}, \quad (7)$$

where $\alpha$ is a scaling factor and $L_{\text{gate}}$ set to 1.0 for all experiments. The proposed algorithm is optimized by continuously fine-tuning new domain data.

**Inference Phase**

The proposed method can adaptively solve the domain-agnostic incremental problem. Furthermore, it can be performed for the open appending domains in the end-to-end network. For a query domain-agnostic instance $x$, we obtain the feature $g_{cls} \in \mathbb{R}^D$ from pretrained shared encoder $f_s$. Then we calculate the feature similarity between $g_{cls}$ and well-trained domain centers $c_t$ in IwR. The domain index with the highest similarity is selected as the gate signal:

$$t = \arg\max_{i \leq [1, T]} f_r(g_{cls}, c_i). \quad (8)$$

According to the predicted domain index $t$, we perform the corresponding expert branch on embedding to generate the final output of face prediction.
# Results of Cross-Domain in Incremental Settings.

In this section, we conduct cross-dataset testing in common Leave-One-Out (LOO) settings of PAD domains. The comparison PAD methods include DA-based and DG-based methods. In contrast, IL settings have more stringent conditions that the model has to be trained on three datasets domain by domain, while DA-based and DG-based methods give the joint training performance, where the model is trained simultaneously on three datasets. It should be noted that sequential training will theoretically affect the ability of the model to learn a generalizable representation due to catastrophic forgetting. Nevertheless, as shown in Table 3, our approach outperforms IL-based methods and can achieve comparable performance with DA-based and DG-based methods in cross-domain settings, which demonstrates the effectiveness of alleviating catastrophic forgetting and preserving generalization capability.

## Ablation Studies

### Ablations of Different Positions of IwR

We insert the IwR module into the ViT network, which is divided into the first $M$ blocks and the last $N - M$ blocks, named shared encoder $f_s$ and expert decoder $f_e$, respectively. Here we aim to evaluate the influence of value $M$ with different classifiers. Figure 4 illustrates the results in step 4. We can observe that the best performance is achieved by inserting IwR into the 8th block. The main reason is that the deeper shared encoder $f_s$ brings more compact features within each domain for IwR, while deeper expert decoder $f_e$ with ADE blocks provides a stronger potential for learning domain-specific knowledge separately and domain-invariant knowledge sequentially. Therefore, we trade off the performance of each module and set $M$ to 8 for all experiments.
<table>
<thead>
<tr>
<th>Methods</th>
<th>O&amp;C&amp;I to M</th>
<th>O&amp;M&amp;I to C</th>
<th>O&amp;C&amp;M to I</th>
<th>I&amp;C&amp;M to O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HTER(%)</td>
<td>AUC(%)</td>
<td>HTER(%)</td>
<td>AUC(%)</td>
</tr>
<tr>
<td>DA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDA (Wang et al. 2021a)</td>
<td>15.40</td>
<td>91.80</td>
<td>24.50</td>
<td>84.40</td>
</tr>
<tr>
<td>VLAD (Wang et al. 2021b)</td>
<td>11.43</td>
<td>96.44</td>
<td>20.79</td>
<td>86.32</td>
</tr>
<tr>
<td>GDA (Zhou et al. 2022b)</td>
<td>9.20</td>
<td>98.0</td>
<td>12.20</td>
<td>93.00</td>
</tr>
<tr>
<td>DG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MADDG (Shao et al. 2019)</td>
<td>17.69</td>
<td>88.06</td>
<td>24.50</td>
<td>84.51</td>
</tr>
<tr>
<td>D2AM (Chen et al. 2021)</td>
<td>12.70</td>
<td>95.66</td>
<td>20.98</td>
<td>85.58</td>
</tr>
<tr>
<td>FGHV (Liu et al. 2022)</td>
<td>9.17</td>
<td>96.92</td>
<td>12.47</td>
<td>93.47</td>
</tr>
<tr>
<td>AMEL (Zhou et al. 2022a)</td>
<td>10.23</td>
<td>96.62</td>
<td>11.88</td>
<td>94.39</td>
</tr>
<tr>
<td>SSDG-R (Jia et al. 2020)</td>
<td>7.38</td>
<td>97.17</td>
<td>10.44</td>
<td>95.94</td>
</tr>
<tr>
<td>SSAN-R (Wang et al. 2022a)</td>
<td>6.67</td>
<td>98.75</td>
<td>10.00</td>
<td>96.67</td>
</tr>
<tr>
<td>IL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LwF (Li and Hoiem 2017)</td>
<td>17.14</td>
<td>98.18</td>
<td>33.94</td>
<td>69.56</td>
</tr>
<tr>
<td>L2P (Wang et al. 2022b)</td>
<td>13.57</td>
<td>93.23</td>
<td>22.28</td>
<td>83.45</td>
</tr>
<tr>
<td>Ours</td>
<td>5.71</td>
<td>98.19</td>
<td>13.22</td>
<td>91.94</td>
</tr>
</tbody>
</table>

Table 3: Comparison to SOTA PAD methods on four cross-dataset benchmarks in different learning settings: DA, DG, and IL. Note that IL methods train on three datasets sequentially, while DA and DG methods train on three datasets together.

<table>
<thead>
<tr>
<th>Expert Branch</th>
<th>Multi classifiers</th>
<th>Asymmetric classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Aaware Agnostic</td>
<td>Aware Agnostic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_1$</td>
<td>9.68</td>
<td>6.33</td>
</tr>
<tr>
<td>$E_2$</td>
<td>20.21</td>
<td>7.14</td>
</tr>
<tr>
<td>$E_3$</td>
<td>24.28</td>
<td>10.82</td>
</tr>
<tr>
<td>$E_4$</td>
<td>25.07</td>
<td>8.52</td>
</tr>
</tbody>
</table>

Table 4: Comparison to different classifiers in different settings in step 4. Results are reported in HTER(%).

Ablations of the Different Classifiers. We compare the performance of different classifiers. Figure 4 shows that the asymmetric classifier has the smallest performance drop in various IwR position settings. Single-classifier strategies suffer from catastrophic forgetting, while multi-classifier has lower performance due to inconsistencies of PPD. We further conduct quantitative analysis between multi-classifiers and asymmetric classifier in Table 4. We train four datasets incrementally and test samples from the previous domain M in domain-aware/agnostic forms. Domain-aware form means that samples are forwarded via designated expert branch, while the domain-agnostic form means that samples are gated by the IwR mechanism. For multi-classifiers, individual expert branches with corresponding classifiers excel in domain-aware form but face notable performance decline when aggregating outputs due to PPD inconsistencies. In contrast, the asymmetric classifier keeps the class center $\theta_l$ shared across different expert branches, achieving superior performance in domain-agnostic form.

Visualization and Analysis

Visualization of Output Distribution. We conduct a statistical analysis of predicted scores from different classifiers. Figure 5 presents the boxplot of distributions based on predicted scores. We observe that in multi-classifiers method, the decision boundary of live and spoof samples varies greatly among different expert branch. A uniform threshold cannot achieve optimal results for all classifier outputs. For asymmetric classifier, the decision boundary is similar among different expert branch. A uniform threshold can maintain satisfactory results in different domains.

Visualization of Grad-CAM. As shown in Figure 6, we utilize the Grad-CAM (Zhou et al. 2016) to illustrate class activation maps. We randomly select some samples from dataset C in step 4 and compare the activation map learned by our method and fine-tuning method. The activation map from fine-tuning is shown in the first row. It has a serious problem of catastrophic forgetting (HTER: 1.38% → 33.94%), which makes activation not look conspicuous for live samples and attention to wrong areas for spoof samples. In contrast, the second row shows that our method preserves activations in facial areas for live samples and highlights spoof cue areas of print attack and replay attack.

Conclusion

In this paper, we propose a novel MDIL framework of PAD to mitigate catastrophic forgetting in previous domains. Specifically, we design the ADE blocks equipped with learnable IwR to learn domain-specific knowledge separately without interference. Furthermore, an asymmetric classifier is designed to address the problem of PPD of live samples inconsistent in open appending domains. Extensive experiments with detailed analysis demonstrate the effectiveness.
References


