Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 32(3): 1973–1997.
DOI: 10.3934/era.2024090
Received: 13 December 2023
Revised: 14 February 2024
Accepted: 22 February 2024
Published: 06 March 2024

Research article

Jointly learning and training: using style diversification to improve domain generalization for deepfake detection

Jicheng Li[1], **Beibei Liu**[1,*], **Hao-Tian Wu**[2], **Yongjian Hu**[1] **and Chang-Tsun Li**[3]

[1] School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, China
[2] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China
[3] School of Information Technology, Deakin University, Geelong, VIC 3216, Australia

* **Correspondence:** Email: eebbliu@scut.edu.cn.

Abstract: Most existing deepfake detection methods often fail to maintain their performance when confronting new test domains. To address this issue, we propose a generalizable deepfake detection system to implement style diversification by alternately learning the domain generalization (DG)-based detector and the stylized fake face synthesizer (SFFS). For the DG-based detector, we first adopt instance normalization- and batch normalization-based structures to extract the local and global image statistics as the style and content features, which are then leveraged to obtain the more diverse feature space. Subsequently, contrastive learning is used to emphasize common style features while suppressing domain-specific ones, and adversarial learning is performed to obtain the domain-invariant features. These optimized features help the DG-based detector to learn generalized classification features and also encourage the SFFS to simulate possibly unseen domain data. In return, the samples generated by the SFFS would contribute to the detector's learning of more generalized features from augmented training data. Such a joint learning and training process enhances both the detector's and the synthesizer's feature representation capability for generalizable deepfake detection. Experimental results demonstrate that our method outperforms the state-of-the-art competitors not only in intra-domain tests but particularly in cross-domain tests.

Keywords: deepfake detection; domain generalization; style diversification; contrastive learning; adversarial learning

1. Introduction

Most deepfake detectors can achieve good performance when applied intra-domain scenarios, but their performance may encounter dramatic degradation when subject to cross-domain settings [1–9].

This phenomenon is largely due to the representation gap between training and testing domains. In other words, the detectors fit the data in the training domain so well that their generalization is likely to be adversely affected. Poor generalization in testing domains with unknown forgery patterns is a non-trivial issue for the industrial use of deepfake detectors. To tackle this issue, many solutions have been proposed.

One common approach to boost generalization capability is to jointly use multiple forensic features from different modalities like spatial modality, transform modality, temporal modality, and compressed modality. For example, Zhou et al. [10] captured both tampering artifact and local noise residual features. Li et al. [11] proposed the Face X-ray method. This generalizable technique succeeds by assuming the existence of blending steps during fake face generation and does not rely on any knowledge of the artifacts associated with specific manipulations. Chen et al. [12] combined features from two color spaces (i.e., RGB and YCbCr). Chen et al. [13] considered that global supervisions are insufficient to learn generalized features and prone to overfitting; thus, they used the correlation between local regions. Zheng et al. [14] detected spatially related manipulation traces (e.g., blending boundary, checkboard, blur artifacts) and the temporal incoherence between frames by employing a fully temporal convolutional network (FTCN) to learn the long-term temporal incoherence. Pang et al. [15] investigated spatial and temporal inconsistency as well as dynamic temporal information, by applying multiple sampling rates. Chen et al. [16] extracted more comprehensive temporal features by using a spatiotemporal attention mechanism and convolutional long short-term memory to enhance spatiotemporal correlations. Liu et al. [17] used a spatial-phase shallow learning network to exploit the sensitivity of the phase spectrum in the presence of up-sampling operation traces in fake face reconstruction. Cao et al. [18] proposed a reconstruction-classification learning framework to enhance the representations of forgery patterns while mining the essential discrepancy between real and fake images. Yu et al. [19] relied on the so-called common forgery feature extractor (CFFE) to explore the commonality of different manipulation traces of deepfakes. Luo et al. [20] proposed a forgery mining framework that utilizes a fine-grained relation learning prototype to mine critical information in forgeries. Considering that videos are usually saved and transmitted in a compression format, some researchers have directly checked forgery traces by using features of compressed modality. For example, Hu et al. [21] detected deepfake videos by learning frame-level features in I frames and temporality-level features in time-dependent residuals. Wang and Chow [22] extracted I frames as key frames to alleviate the information loss and extracted the face-background pairs via the Siamese structure network.

Recently, another promising approach has emerged, where more sophisticated detectors are trained with a vast range of sample data that represent different manipulation artifacts. Different from previous methods focusing on exhaustively exploring new forensic features, these methods also tend to generate self-making data to extend the training space. Using real data from business application scenarios is desirable but not easily achievable due to high cost and other legal issues like privacy. Whereas, generating pseudo training data from lab-based deepfake imitators seems more plausible since those generators are based on common knowledge such as the image inpainting employed in deepfake processes. So far this approach has apparently demonstrated its effectiveness. For example, in their pioneering work [23], Li and Lyu proposed the face warping artifacts (FWAs) method to simulate fake face images by blurring facial regions of real images. Unlike common data augmentation strategies, the FWAs method generates pseudo samples by purposely simulating the manipulation traces left on facial areas in the process of face swapping. Following this idea, Zhao et al. [24] designed an inconsistency image generator to automatically generate annotated training data by using their pairwise self-consistency learning network. Shiohara and Yamasaki [25] proposed a simple yet effective synthetic scheme based on the use of self-blended images (SBIs). SBIs blend pseudo source and target images from single pristine images to reproduce common forgery artifacts. Recently, Chen et al. [26] proposed a self-prediction learning (SPL) method to learn rich hidden representations by predicting masked patches in the pre-training stage. After pre-training, the discriminative model is fine-tuned via multi-task learning, including a deepfake classification task and a forgery mask estimation task. Since face forgery simulation is directly carried out on images in [23–26], we call this pseudo data generation manner image-level data augmentation.

In general, the above two approaches have different focuses. The first approach concentrates on improving feature extraction capability whereas the second one makes efforts to provide a better training environment. Usually, these two approaches work separately. Their connection is exhibited in [25], where SBIs are used to generate pseudo samples and the generated samples are subsequently used to fine-tune the feature extractor. In this work, however, we propose a way to integrate these two approaches into a complete deepfake detection system. They work in a domain generalization (DG) framework and dynamically improve each other. To the best of our knowledge, this is the first work to integrate these two approaches into an end-to-end detector. Our major contributions are summarized as follows.

- For deepfake detection, we experimentally exhibit that style feature distributions vary with different datasets, which exposes the severe challenge of cross-domain detection. Good generalization can be realized unless the feature spaces for different target domains share a common projection. To find such a projection, we concentrate on extracting general features that have good commonality in different target domains.
- A novel deepfake detection system is proposed; it consists of a DG-based deepfake detector and a stylized fake face synthesizer (SFFS). These two parts interact with each other dynamically, focusing on style diversification from feature representation and training data spaces to alleviate overfitting to a specific domain. A two-phase training strategy is adopted in the proposed system. In the first phase, the detector is updated by applying backpropagation, and the SFFS with its weights frozen generates pseudo fake face samples for the updating process of the DG-based detector. In the second phase, the style decoder of the SFFS is trained on samples from simulated unseen domains with style diversification.
- Since style features have a high-frequency property and can better preserve manipulation traces, we use contrastive learning to emphasize the common style features under a DG framework. On the other side, content features usually have low-frequency property and may contain imaging pipeline and storage traces, so we use adversarial learning to emphasize the common content features. The stylized face images are obtained by modifying the style features. This feature-level enrichment drives the fake face synthesizer to produce pseudo training data with more diversified features. The output of the SFFS with image-level generation is then used as new source training data to further train our detection network. Experimental results have demonstrated that this interaction significantly improves the generalization capability of our deepfake detector.

The rest of this paper is organized as follows. Section 2 introduces the problem statement and our methodology. Section 3 explains our proposed detection system in detail. Section 4 gives experimental results and discussions. Section 5 concludes this work.

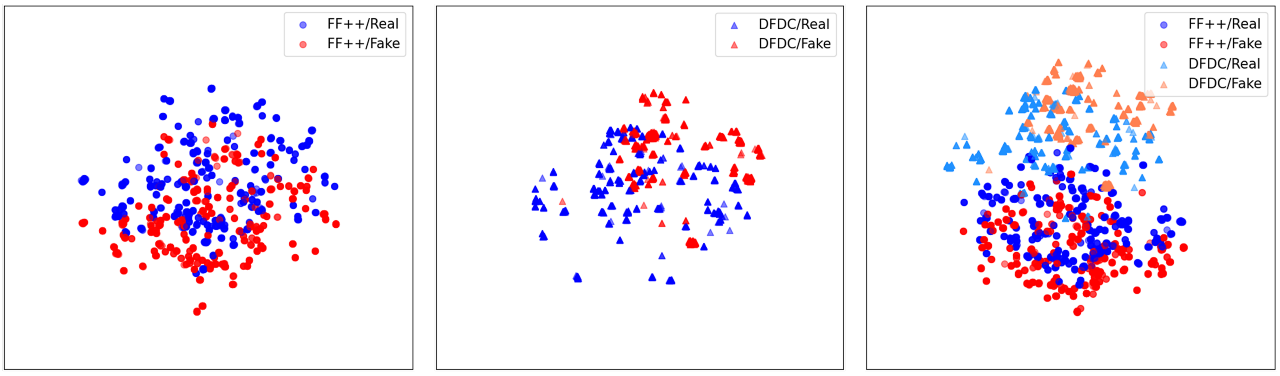
2. Problem statement and our methodology

To better understand the challenge of cross-dataset testing, we choose to use two public datasets, namely FaceForensics++ (FF++) [27] and Deepfake Detection Challenge (DFDC) [28], to visualize their differences in feature spaces by using the t-SNE tool [29]. Similar to [30], we calculate the style feature f_s of each dataset and show its position in the feature space in Figure 1. The details of calculating f_s will be given in the next section. We use red/orange color to represent fake/forged faces and blue/light blue color to represent real faces. The blue points and red points in Figure 1(a) refer to the style features of real and fake faces in the FF++ dataset, respectively. Meanwhile, the blue and orange triangles in Figure 1(b) refer to the style features of real and fake faces in the DFDC dataset, respectively. From the shape of their feature distributions, it could be easily imagined that their classification planes for real and fake faces would be different. Observing the feature distributions of the two datasets in Figure 1(c), we can immediately notice that, regardless of the dataset we used to train the deepfake detector, it is hard to avoid severe performance degradation in the classification stage. This is the so-called lack of generalization for machine learning-based detectors. Moreover, we can calculate the content features of the same two datasets by using a similar method as that used to obtain Figure 2. This situation is similar to what we have seen in Figure 1. This phenomenon is also called the domain shift between the source (training-time) and the target (test-time) domains in cross-dataset testing. The detection performance would degrade more severely with the increase of target domains. Our task in this work is to find a way to alleviate the degradation of detection performance. In other words, we needed to find better feature representations so that the feature distributions from different target domains have more commonality.

One solution to our problem is domain adaptation (DA), which learns a discriminative classifier or other predictor in the presence of a shift between training and test distributions. The appeal of the DA approaches is their ability to learn a mapping between domains in the situation in which the target domain data are either fully unlabeled (unsupervised domain annotation) or have few labeled samples (semi-supervised DA) [31]. Considering that fake faces have different unknown sources and vary with the rapid development of new technologies, we prefer to regard this problem as a DG issue. The difference between DA and DG is that DA has to access the target domain while DG cannot observe any target domain during training. This makes DG more challenging, but more realistic and favorable than DA in practical applications [32].

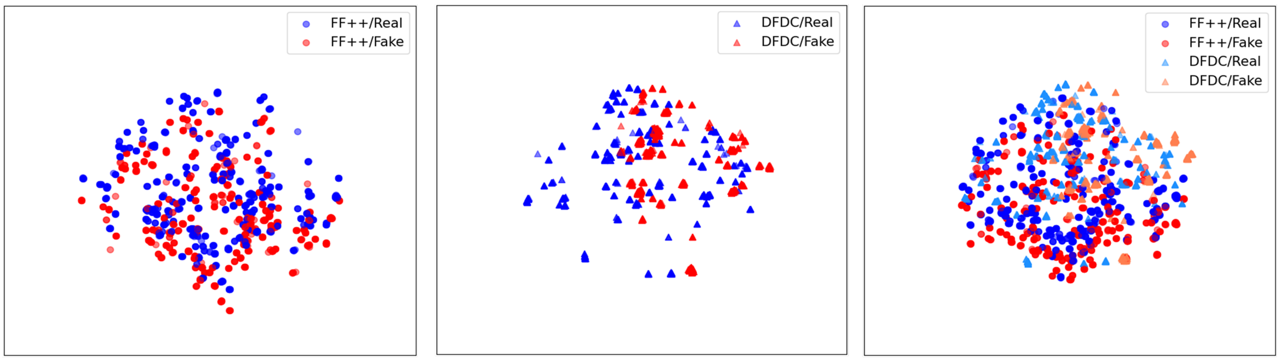
DG aims to first learn a generic model on multiple source domains and then directly generalize to an arbitrary unseen target domain without any additional adaption [32]. Recently, DG for generalizing detection has drawn more and more attention in areas such as person re-identification [33], image recognition [34] and face anti-spoofing (FAS) [30,35]. Wang et al. [30] adjusted the stylized features to enhance liveness-related style information and suppress domain-specific ones in FAS. Style adjustment is widely used in generative frameworks, where content features are taken as global statistics and style features as local statistics. The reassembling of the pair of content and style features can be used to generate pseudo spoof faces. There are different ways to implement style modification in literature. Huang and Belongie [36] proposed adaptive instance normalization (AdaIN) for style transfer and demonstrated that the style information can easily be changed by adjusting the mean and variance of convolutional feature maps. StyleGAN [37] also produced impressive image generation results through the application of AdaIN operations in a generative network. Wang et al. [32] performed random style transformation with extra random noise to generate features with diverse styles and thus enhance the generalization ability.

Motivated by these works, we propose a novel deepfake detection framework as shown in Figure 3. It consists of two parts: the DG-based detector and the SFFS. The detector and the SFFS model work alternately and dynamically update the network parameters.



(a) Original style features in FF++ (b) Original style features in DFDC (c) Original style features in FF++ and DFDC

Figure 1. Distributions of style features with t-SNE visualization.



(a) Original content features in FF++ (b) Original content features in DFDC (c) Original content features in FF++ and DFDC

Figure 2. Distributions of content features with t-SNE visualization.

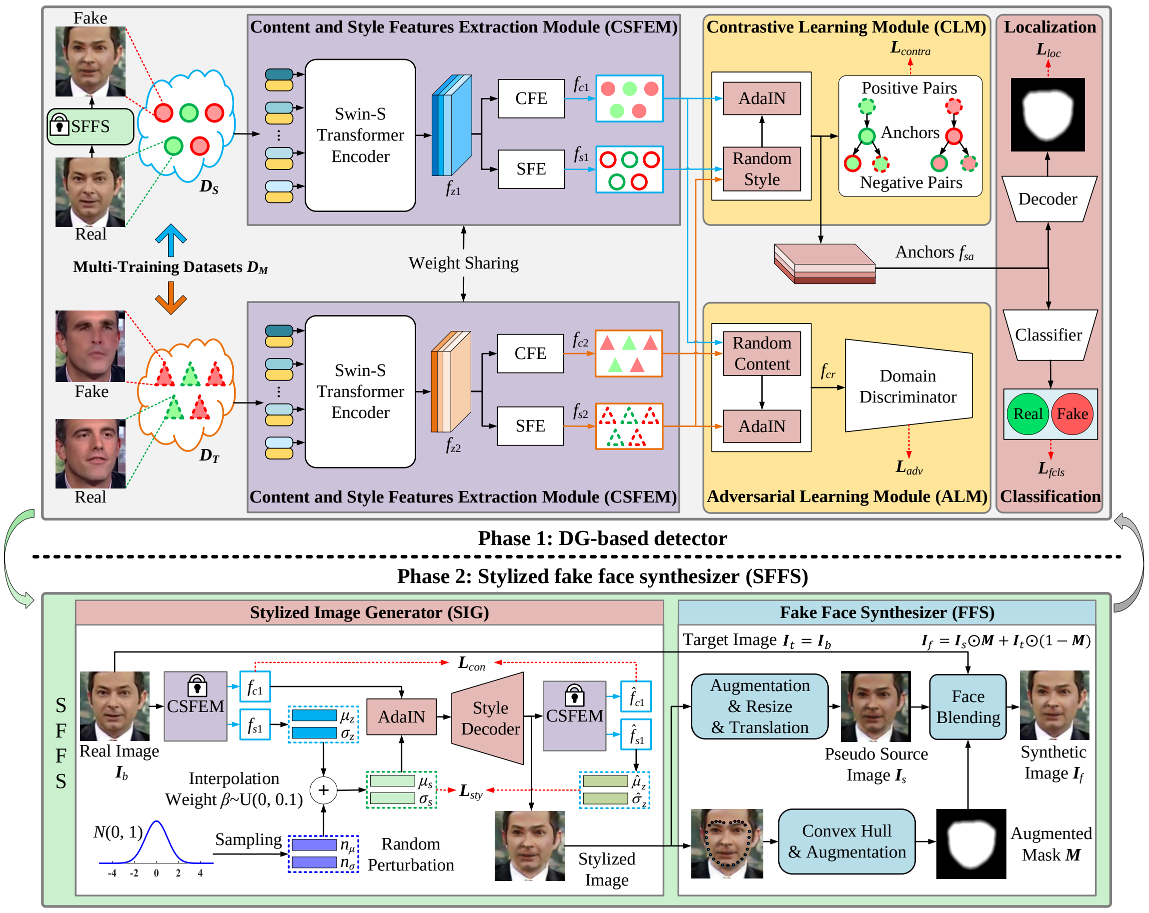

Content and Style Features Extraction Module (CSFEM)
Contrastive Learning Module (CLM)
Localization
Fake
SFFS
Real
Swin-S Transformer Encoder
CFE
SFE
AdaIN
Random Style
Positive Pairs
Anchors
Negative Pairs
Decoder
Anchors f_{sa}
Multi-Training Datasets D_M
Weight Sharing
Random Content
Domain Discriminator
Classifier
Real
Fake
Adversarial Learning Module (ALM)
Classification
Phase 1: DG-based detector
Phase 2: Stylized fake face synthesizer (SFFS)
Stylized Image Generator (SIG)
Fake Face Synthesizer (FFS)
Target Image $I_t = I_b$
$I_f = I_s \odot M + I_t \odot (1 - M)$
CSFEM
Style Decoder
Real Image I_b
Interpolation Weight β~U(0, 0.1)
Sampling
Random Perturbation
Stylized Image
Augmentation & Resize & Translation
Pseudo Source Image I_s
Face Blending
Synthetic Image I_f
Convex Hull & Augmentation
Augmented Mask M

Figure 3. The framework of the proposed method. For the contrastive learning module (CLM), different colors indicate different authenticity information (green = real, red = fake), different shapes represent content information from different domains (round = D_S, triangle = D_T), and different linetypes denote style information from different domains (solid line = D_S, dashed line = D_T).

3. Proposed deepfake detection system

The DG-based detection model consists of four modules: the content and style features extraction module (CSFEM), the adversarial learning module (ALM), the contrastive learning module (CLM), and the face forgery classification and localization module (FFCLM). For the given multi-training datasets D_M, we first randomly split it into D_T and D_S with the same number of samples and non-domain overlapping, and apply the SFFS to create pseudo training data to expand D_S. Subsequently, we utilize the CSFEM to extract two sets of style and content features corresponding to D_S and D_T. Then, we apply AdaIN to randomly reassemble the above two sets of style and content features to construct the content randomization and style randomization feature spaces. The content randomization feature space is used for the ALM to obtain a domain-invariant representation, and the style randomization feature space is for the CLM to emphasize common style features as well as suppress the domain-specific ones. Finally, the learned generalized representation, assembled by domain-invariant content features and common style features, is fed to the FFCLM for forgery classification and localization.

The SFFS model consists of two modules: the stylized image generator (SIG) and fake face synthesizer (FFS). The SIG uses style perturbation and style transfer to enrich the style diversity. Then, the FFS is utilized to generate the fake face images that represent different manipulation artifacts. The complete method is illustrated in Algorithm 1. We will describe our method module by module below.

Algorithm 1 Optimization process for proposed framework

```
Require: The multiple training datasets $D_M$.
Ensure: Final optimized model parameters $\mathbf{\Phi}$ of DG-based detector and $\mathbf{\Theta}$ of SFFS.
1: Initialize model parameters $\mathbf{\Phi}$ and $\mathbf{\Theta}$.
2: while not end of iteration do
3:     Randomly split $D_M$ into $D_S$ and $D_T$ without domain overlapping.
4:     Input the real samples from a randomly selected data domain in $D_S$ to SFFS to generate the
       synthesized samples, which are used to create pseudo training data with a new domain label.
5:     // Phase 1: Train the DG-based detector
6:     Sample a batch of data $\{x_S, y_S\}$ from $D_S$.
7:     Input $x_S$ to the CSFEM to obtain the content feature $f_{c1}$ and style feature $f_{s1}$.
8:     Sample a batch of data $\{x_T, y_T\}$ from $D_T$.
9:     Input $x_T$ to the CSFEM to obtain the content feature $f_{c2}$ and style feature $f_{s2}$.
10:    Utilize $f_{c1}$, $f_{s1}$, and $f_{s2}$ to get the self-assembly features $f_{sa}$ and random-assembly features $f_{ra}$.
11:    Compute the contrastive loss $\boldsymbol{L}_{contra}$ with Eq (3.3).
12:    Utilize $f_{c1}$, $f_{c2}$, and $f_{s2}$ to get the content randomization features $f_{cr}$.
13:    Compute the adversarial loss $\boldsymbol{L}_{adv}$ with Eq (3.7).
14:    Compute the classification loss $\boldsymbol{L}_{fcls}$ with Eq (3.8).
15:    Compute the localization loss $\boldsymbol{L}_{loc}$ with Eq (3.9).
16:    Compute the overall detection loss $\boldsymbol{L}_d$ with Eq (3.10).
17:    Fix the SFFS and then minimize $\boldsymbol{L}_d$ and update $\mathbf{\Phi}$.
18:    // Phase 2: Train the SFFS
19:    Get the channel-wise mean $\mu_z$ and standard deviation $\sigma_z$ of $f_{s1}$.
20:    Sample two perturbation variables $\mu_n$ and $\sigma_n \sim N(0, 1)$ with the same dimension as $\mu_z$ and $\sigma_z$.
21:    Generate the mixed style $\mu_s$ and $\sigma_s$ by using Eqs (3.11) and (3.12).
22:    Get the new augmented feature $f_g$ by using AdaIN.
23:    Get the stylized image $Dec(f_g)$ by using the style decoder $Dec$.
24:    Input the stylized image to the CSFEM to obtain the content feature $\hat{f}_{c1}$ and style feature $\hat{f}_{s1}$.
25:    Get the channel-wise mean $\hat{\mu}_z$ and standard deviation $\hat{\sigma}_z$ of $\hat{f}_{s1}$.
26:    Compute the content loss $\boldsymbol{L}_{con}$ with Eq (3.14).
27:    Compute the style loss $\boldsymbol{L}_{sty}$ with Eq (3.15).
28:    Compute the overall generation loss $\boldsymbol{L}_{SFFS}$ with Eq (3.13).
29:    Fix the CSFEM and then minimize $\boldsymbol{L}_{SFFS}$ and update $\mathbf{\Theta}$.
30: end while
```

3.1. Content and style features extraction module

For deepfake detection, we are not interested in meaningful image content, rather, we focus on forgery traces to answer the question of whether a test face image is real. For this reason, many traditional methods often carry out forensic investigation into high-frequency image components that directly reflect image details like texture and edge, while ignoring the useful forensic information hidden in image content. Image content mainly exists in low-pass components, reflecting the approximation of the image. Although the existence of image content would interfere with the accurate extraction of manipulation artifacts, we think that image content is the carrier of image processing history and can provide useful information for image content-related traces like image origin, compression standards, compression ratios, storage format, and other low-pass imaging pipeline processing operations. Based on this consideration, we propose to suppress meaningful image content and extract high-level semantic features that contain content and style features. The CSFEM has been designed to output the content feature f_{c1} and the style feature f_{s1}. In the first procedure, the hierarchical vision transformer using shifted windows (Swin-S) [38] is applied as an embedding network to learn high-level semantic representations. In the second procedure, the content features extraction (CFE) and style features extraction (SFE) branches obtain the outputs of f_{c1} and f_{s1}.

Training a Swin-S with good representation ability relies on a huge dataset to learn the prior information. Since the unsupervised representation learning strategy called SimMIM [39] has demonstrated that masked image modeling (MIM) pre-training can yield good generalization results on various downstream tasks, we adopt the SimMIM strategy to pre-train the Swin-S on publicly available deepfake datasets. After pre-training, the Swin-S is optimized together with the overall network. The content features and the style features are directly learned via CFE and SFE. Their network structures are shown in Table 1. Like [30], batch normalization-based structures are used to summarize global image statistics while instance normalization-based structures focus on the distinctive characteristics of specific samples.

Table 1. Parameters for the proposed detection network. k means kernel size, c_{out} represents the output channels, s is the stride, p denotes the operation of padding zeros, B is the batch size, and H_0 and W_0 indicate the height and width of input image.

Phase	Layers	Name	Parameters	Output feature size

Phase 1: DG-based detector	Encoder in CSFEM	$x_1 = \text{Swin-S}(x_0)$	embed dim = 96
			depths = [2, 2, 18, 2]
			heads = [3, 6, 12, 24]
			window size = 7
				$B \times 768 \times \lceil \frac{H_0}{32} \rceil \times \lceil \frac{W_0}{32} \rceil$

	CFE in CSFEM	$x_2 = \begin{bmatrix} \text{Conv2D}(x_1) \\ \text{BatchNorm2D}(x_1) \\ \text{ReLU}(x_1) \end{bmatrix}_{\times 3}$	c_{out} = [768, 768, 768] $k = 3 \times 3$ $s = 1$, p = 'same'	$B \times 768 \times \lceil \frac{H_0}{32} \rceil \times \lceil \frac{W_0}{32} \rceil$
	SFE in CSFEM	$x_3 = \begin{bmatrix} \text{Conv2D}(x_1) \\ \text{InstanceNorm2D}(x_1) \\ \text{ReLU}(x_1) \end{bmatrix}_{\times 3}$	c_{out} = [768, 768, 768] $k = 3 \times 3$ $s = 1$, p = 'same'	$B \times 768 \times \lceil \frac{H_0}{32} \rceil \times \lceil \frac{W_0}{32} \rceil$
	AdaIN	$x_4 = \text{AdaIN}(x_2, x_3)$	$c_{out} = 768$	$B \times 768 \times \lceil \frac{H_0}{32} \rceil \times \lceil \frac{W_0}{32} \rceil$
	Discriminator	$x_5 = \begin{bmatrix} \text{GRL}(x_4) \\ \text{Linear}(x_4) \\ \text{ReLU}(x_4) \\ \text{Dropout}(0.5) \\ \text{Linear}(x_4) \end{bmatrix}$	c_{out} = [256, 2]	$B \times 2$
	Classifier	$x_6 = \text{Linear}(x_4)$	$c_{out} = 2$	$B \times 2$
	Decoder	$x_7 = \begin{bmatrix} \text{Upsample}(x_4) \\ \text{Conv2D}(x_4) \\ \text{BatchNorm2D}(x_4) \\ \text{ReLU}(x_4) \end{bmatrix}_{\times 5}$	scale = 2 c_{out} = [512, 256, 128, 64, 2] $k = 3 \times 3$ $s = 1$, p = 'same' scale = 2	$B \times 2 \times H_0 \times W_0$
Phase 2: SFFS	Style Decoder in SIG	$x_7 = \begin{bmatrix} \text{Upsample}(x_4) \\ \text{Conv2D}(x_4) \\ \text{ReLU}(x_4) \\ \text{Conv2D}(x_4) \\ \text{ReLU}(x_4) \end{bmatrix}_{\times 5}$	scale = 2 c_{out} = [512, 256, 128, 64, 3] $k = 3 \times 3$ $s = 1$, p = 'same' scale = 2	$B \times 3 \times H_0 \times W_0$

3.2. Contrastive learning module

Fake faces often undergo an affine transform (i.e., scaling, rotation and shearing) to match the poses of the target faces that they will replace [23]. Compared with real faces, fake faces generated by common deepfake tools would lose high-frequency textures and details. The difference between these two faces can be better reflected by their style features. From the view of style features, a major obstacle to generalization is that domain-specific style features may conceal common ones in cross-domain scenarios, which may result in misjudgment for unseen data. To solve this problem, we propose a contrastive learning approach to emphasize common style features while suppressing domain-specific ones.

We fix the source domain f_{c1} when we use the contrastive learning strategy. Two types of features are introduced: self-assembly features $f_{sa}(f_{c1,i}, f_{s1,i})$ as anchor features and random-assembly features $f_{ra}(f_{c1,i}, f_{s*,i})$ as styled features. An adaptive style transfer method, namely AdaIN [36] is used to assemble a content feature and a style feature as follows.

$$f_{sa}(f_{c1,i}, f_{s1,i}) = \sigma(f_{s1}) \cdot \left(\frac{f_{c1,i} - \mu(f_{c1})}{\sigma(f_{c1})} \right) + \mu(f_{s1}), \tag{3.1}$$

$$f_{ra}(f_{c1,i}, f_{s*,i}) = \sigma(f_{s*}) \cdot \left(\frac{f_{c1,i} - \mu(f_{c1})}{\sigma(f_{c1})} \right) + \mu(f_{s*}), \tag{3.2}$$

where i is the index and f_{s*} is randomly picked from $\{f_{s1}, f_{s2}\}$. $\mu(\cdot)$ and $\sigma(\cdot)$ are the respective channel-wise mean and standard deviation of the feature. In essence, AdaIN realizes the style transfer by replacing the mean and standard deviation in Eq (3.1) with those from shuffled style features. We implement Eqs (3.1) and (3.2) batch by batch.

To avoid overfitting, a stop-gradient operation is implemented on anchor features to fix their position in the feature space. We carry out the contrastive learning by measuring the cosine similarity between $f_{sa}(f_{c1,i}, f_{s1,i})$ and $f_{ra}(f_{c1,i}, f_{s*,i})$. $f_{ra}(f_{c1,i}, f_{s*,i})$ is drawn closer to or pushed away from their corresponding $f_{sa}(f_{c1,i}, f_{s1,i})$. The batch-based contrastive loss $\boldsymbol{L}_{contra}$ is defined below:

$$\boldsymbol{L}_{contra} = \frac{1}{B} \sum_{i=1}^{B} Sign(f_{sa}(f_{c1,i}, f_{s1,i}), f_{ra}(f_{c1,i}, f_{s*,i})) \cdot Dist(f_{sa}(f_{c1,i}, f_{s1,i}), f_{ra}(f_{c1,i}, f_{s*,i})), \tag{3.3}$$

where B is the batch size. $Sign(\cdot)$ checks the consistency of feature labels. $Dist(\cdot)$ is the negative cosine similarity. We calculate $Sign(\cdot)$ and $Dist(\cdot)$ as follows. Here $\|\cdot\|_2$ denotes the ℓ_2-norm.

$$Sign(f_{sa}(f_{c1,i}, f_{s1,i}), f_{ra}(f_{c1,i}, f_{s*,i})) = \begin{cases} +1, & label(f_{c1,i}, f_{s1,i}) = label(f_{SR,i}) \\ -1, & otherwise \end{cases}, \tag{3.4}$$

$$Dist(f_{sa}(f_{c1,i}, f_{s1,i}), f_{ra}(f_{c1,i}, f_{s*,i})) = -\frac{f_{sa}(f_{c1,i}, f_{s1,i})}{\left\| f_{sa}(f_{c1,i}, f_{ra}(f_{c1,i}, f_{s*,i})) \right\|_2} \cdot \frac{f_{SR,i}}{\left\| f_{ra}(f_{c1,i}, f_{s*,i}) \right\|_2}. \tag{3.5}$$

3.3. Adversarial learning module

Content features are useful for sharing common low-pass information in cross-domain scenarios. They mainly record some global semantic features and physical attributes; thus, a shared feature distribution is easily acquired by using adversarial learning [30]. Based on this consideration, we choose to use AdaIN in a different way that we generate the content randomization features f_{cr} with unchanged style but shuffled content features. In Eq (3.6), f_{c*} is randomly selected from $\{f_{c1}, f_{c2}\}$.

$$f_{cr}(f_{c*,i}, f_{s2,i}) = \sigma(f_{s2}) \cdot \left(\frac{f_{c*,i} - \mu(f_{c*})}{\sigma(f_{c*})} \right) + \mu(f_{s2}). \tag{3.6}$$

The domain discriminator is used to make the f_{cr} indistinguishable for different domains. Just like [30], the parameters of the content random feature extractor G_C are optimized by maximizing the adversarial loss function while the parameters of the domain discriminator D are optimized in the opposite direction. The process is formulated as follows:

$$\min_{G_C}\max_{D} \boldsymbol{L}_{adv} = -\mathbb{E}_{(x,y)\in(X,Y_D)} \sum_{i=1}^{N_D} \mathbb{1}[i = y] log D(G_C(x)), \tag{3.7}$$

where Y_D is the set of domain labels and N_D is the number of different data domains. $\mathbb{1}[i = y]$ indicates that the value is 1 when $i = y$; otherwise, it is 0. To optimize G_C and D simultaneously, the gradient reversal layer [31] is used to reverse the gradient by multiplying it by a negative scalar during the backward propagation. Table 1 shows the details of discriminator D.

3.4. Face forgery classification and localization module

The anchor $f_{sa}(f_{c1,i}, f_{s1,i})$ is sent to the FFCLM for binary cross entropy calculation and localization comparison. Unlike traditional deepfake detectors with only classification metric, we use both classification loss and localization loss to supervise the training process. The newly introduced localization information comes from the ground-truth face mask in training datasets, which is a kind of side information. The cross entropy loss $\boldsymbol{L}_{fcls}$ and the pixel-level cross entropy localization loss $\boldsymbol{L}_{loc}$ are given as follows.

$$\boldsymbol{L}_{fcls} = -\sum_{i=1}^{N_C} y_i log(p_i), \tag{3.8}$$

$$\boldsymbol{L}_{loc} = -\frac{1}{H_0 W_0} \sum_{h=1}^{H_0} \sum_{w=1}^{W_0} \sum_{i=1}^{N_C} \boldsymbol{M}_{ihw} log(\hat{\boldsymbol{M}}_{ihw}), \tag{3.9}$$

where N_C indicates number of classes (i.e., $N_C = 2$), p_i and y_i are the softmax prediction probabilities of the classifier and the ground-truth label, respectively. $\boldsymbol{M}$ is the ground-truth forgery mask with 1 indicating the manipulated pixels and 0 otherwise. The decoder consists of up-sampling and convolutional layers to estimate a forgery mask $\hat{\boldsymbol{M}} \in \mathbb{R}^{N_C \times H_0 \times W_0}$.

To sum up, the overall detection loss function is formulated as in Eq (3.10), where α_1, α_2, and α_3 are the weights used to balance the four loss terms. The complete network structure and parameters for our DG-based detector are shown in Table 1.

$$\boldsymbol{L}_d = \boldsymbol{L}_{fcls} + \alpha_1 \cdot \boldsymbol{L}_{loc} + \alpha_2 \cdot \boldsymbol{L}_{contra} + \alpha_3 \cdot \boldsymbol{L}_{adv}. \tag{3.10}$$

3.5. Stylized fake face synthesizer

We develop a pseudo fake face generator called the SFFS to enrich the style diversity of training data. The SFFS consists of two parts: the SIG and FFS. The SIG uses feature-level modifications, more exactly, style perturbation and style transfer to enrich the diversity of the style features. It is an independent module which can generate stylized images. Meanwhile, the FFS is an independent image-level-augmentation pseudo face generator. For simplicity, we directly apply the SBIs as in [25] and FWAs as in [23] to accomplish this task. When the SIG and FFS work together, they can generate augmented training samples, i.e., stylized fake faces, by using the output of the SIG. The structures of the SIG and FFS are respectively illustrated in Figure 3. We give more detailed descriptions below.

The SIG has an encoder-decoder structure with the encoder CSFEM and the style decoder *Dec*. The architecture and parameters of the SIG are shown in Table 1. Given a real image $\boldsymbol{I}_b$, the content feature f_{c1} and style feature f_{s1} are extracted by the CSFEM. The style transfer is realized by integrating the perturbation information into the original style in a hidden space. Exactly, we modify the channel-wise mean μ_z and standard deviation σ_z of f_{s1} with two random perturbation variables n_μ and n_σ as in Eqs (3.11) and (3.12). The new channel-wise mean μ_s and standard deviation σ_s are obtained as follows [32].

$$\mu_s = \beta \times n_\mu + (1 - \beta) \times \mu_z, \tag{3.11}$$

$$\sigma_s = \beta \times n_\sigma + (1 - \beta) \times \ \sigma_z, \tag{3.12}$$

where n_μ and $n_\sigma \sim\ N(0,\ 1)$ are two perturbations with the same dimension as μ_z and σ_z. β is an interpolation weight sampled from uniform distribution, denoted as $\beta \sim \mathrm{U}(0, 0.1)$. We use AdaIN, as defined in Eq (3.2), to reassemble f_{c1} with μ_s and σ_s to get the new augmented feature f_g. The stylized image $Dec(f_g)$ is obtained from the decoder by using f_g as an input. In the training of the SFFS, we fix the parameters of the CSFEM and train the decoder with the overall generation loss $\boldsymbol{L}_{SFFS}$ as follows.

$$\boldsymbol{L}_{SFFS} = \boldsymbol{L}_{con} + \alpha_4 \cdot \boldsymbol{L}_{sty}, \tag{3.13}$$

$$\boldsymbol{L}_{con} = \left\|\hat{f}_{c1} - f_{c1}\right\|_2, \tag{3.14}$$

$$\boldsymbol{L}_{sty} = \|\hat{\mu}_z - \mu_s\|_2 + \|\hat{\sigma}_z - \sigma_s\|_2, \tag{3.15}$$

where α_4 is the weight use to balance the content loss $\boldsymbol{L}_{con}$ and the style loss $\boldsymbol{L}_{sty}$. The content feature $\hat{f}_{c1}$ and style feature $\hat{f}_{s1}$ of $Dec(f_g)$ are also extracted by the CSFEM, as shown in Figure 3. $\hat{\mu}_z$ and $\hat{\sigma}_z$ denote the channel-wise mean and standard deviation of $\hat{f}_{s1}$.

We use the self-blending strategy in [25] to explain the function of the FFS. Let $\boldsymbol{I}_s = Dec(f_g)$ and $\boldsymbol{I}_t = \boldsymbol{I}_b$ represent the source and target images, respectively. All augmentations and transformations applied to $\boldsymbol{I}_s$ are similar to those in [25]. Dlib [40] is used to predict a facial region and the mask $\boldsymbol{M}$ is initialized by calculating the convex hull from predicted facial landmarks. Then $\boldsymbol{M}$ is augmented through elastic deformation and smoothing to increase the diversity of the blending mask and consequently obtain $\tilde{\boldsymbol{M}}$. The pseudo fake face image $\boldsymbol{I}_f$ is obtained by using Eq (3.16). Here $\odot$ denotes element-wise multiplication.

$$\boldsymbol{I}_f = \boldsymbol{I}_s \odot \tilde{\boldsymbol{M}} + \boldsymbol{I}_t \odot (1 - \tilde{\boldsymbol{M}}) \tag{3.16}$$

The training process for the SFFS is illustrated in Algorithm 1. In the training phase of the SFFS, we first fix the model parameters of CSFEM and then update the model parameters of the style decoder *Dec* by minimizing the generation loss $\boldsymbol{L}_{SFFS}$. The well-trained SFFS generates diverse training data for the next training phase of the DG-based detector. This alternating updating process is repeated epoch by epoch.

Similarly, FWAs [23] can also be incorporated within the SFFS. We provide some resulting fake face images in Figure 4. Compared to DeepFakes (DF) in FF++ [27], the SFFS can generate more diverse and natural-looking fake faces. The feature distribution of those generated images and its analysis are given in Section 4.6.3. Furthermore, the contributions of the SFFS and its key components are studied in Section 4.5.

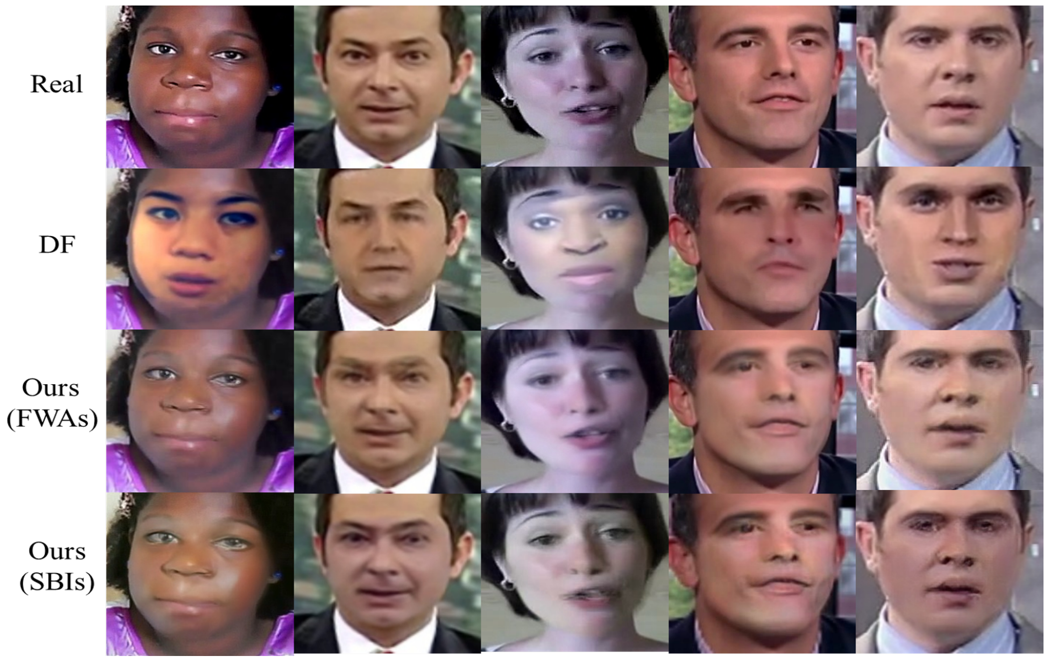


Figure 4. Visualization of real images (row 1), DeepFakes (DF) in FF++ (row 2), and the synthesized images of the SFFS by using FWAs (row 3) and SBIs (row 4) to act as the FFS module, respectively.

4. Experiments

To demonstrate the effectiveness of our method, we compare the proposed detector with 7 typical methods, namely the baseline method Xception [1], published in 2017, FTCN [14], published in 2021, SimMIM [39], SBIs [25], and CFFE [19], published in 2022, AdapGRnet [41] and SPL [26], published in 2023. For a fair comparison, we simulate the compared methods with their officially provided source codes under the same conditions and environment.

4.1. Datasets

We perform experiments on five widely used benchmark datasets, i.e., FF++ [27], DeepfakeDetection (DFD) [42], DFDC [28], Celeb-DF (CDF) [43], and DeeperForensics-1.0 (Deeper) [44]. **FF++** contains 1000 original videos collected from YouTube and 4000 fake videos forged by four manipulation methods, i.e., DeepFakes (DF), Face2Face (F2F), FaceSwap (FS), and NeuralTextures (NT). The videos in FF++ have three kinds of video compression qualities: raw (c0), high-quality (c23) and low-quality (c40). **DFD** provides 363 real videos and 3068 faked videos and contains a variety of life scenes and facial expression changes. Similarly, it also includes three subsets of c0, c23, and c40 videos **DFDC** is a large-scale dataset in which subjects in complex scenes are manipulated by using various unknown methods to develop deepfake detection algorithms in the real scenarios. **CDF** is a high-quality face swapping dataset which contains 590 real videos and 5639 fake videos. It applies a more advanced DF method to generate the fake videos. **Deeper** is one of the largest datasets; it which modifies the pristine videos in FF++ to have new face IDs by using a new DF manipulation method.

Like [27, 28, 43, 44], we split the datasets into training, validation, and testing sets at the video level. Table 2 shows the details of our used datasets. We use the OpenCV toolkit [45] to split each video into frames and then evenly sample 100 frames from each video. For each frame, we then use Dlib toolkit [40] to detect the bounding box of the face, which is then enlarged by 30% to contain more image content around the face. Notably, during the pre-training stage of Swin-S, we utilize the unlabeled mixed training set named DF-Mix, taken from [26], as the pre-training dataset.

Table 2. The detailed information of the used datasets.

Datasets	Real Videos/Fake Videos		
	Training	Validation	Testing
FF++(c23) (DF/FS/F2F/NT)	720/720	140/140	140/140
DFD(c23)	-	-	363/3068
DFDC	-	-	2500/2500
CDF	-	-	178/340
Deeper	-	-	2000/2000

4.2. Experimental setup

The Swin-S transformer encoder is pre-trained by using the regular settings of SimMIM, as detailed in [39]. To train our detection model, the input images are resized to 224 × 224 (i.e., $H_0 = W_0 = 224$) and the AdamW optimizer is employed with a cosine learning rate scheduler. The training hyperparameters are set as follows: the mini-batch size $B = 32$, an initial learning rate of 0.001, weight decay of 0.05, $\beta_1 = 0.9, \beta_1 = 0.999$, and a warm-up of 5 epochs. The weight factors α_1, α_2, α_3, and α_4 of the loss function are set as 1.0, 0.5, 0.5, and 1.0, respectively. For each mini-batch, the number of real and fake samples is half and half. However, for the fake samples, half of the fake images come from the training dataset, and the rest come from SFFS. This training stage lasted for 50 epochs.

We use the area under the receiver operating characteristic curve (AUC) as the evaluation metric. The AUC value can reflect the overall performance under different detection thresholds, which is independent of the threshold values. A high AUC value usually indicates a good detection accuracy and a low detection error rate. The AUC values are calculated based on a frame-level evaluation in this work. From the viewpoint of generalization, the AUC values in cross-domain testing are expected to remain the same as in intra-dataset testing. From the viewpoint of robustness, the AUC values are expected to be the same for different dataset testings.

4.3. Experiment on generalization to unseen datasets

To evaluate performance in cross-domain scenarios, we use the complete FF++(c23) collection as the training dataset, including four types of face forgery manipulations: DF, F2F, FS and NT, and we use the following four datasets as unseen datasets: DFD(c23), DFDC, CDF, and Deeper. The results are listed in Table 3, where the best result is bolded and the second best result is underlined. For comparison purposes, we also list the AUC values of the compared methods, namely Xception, FTCN, SimMIM, SBIs, CFFE, AdapGRnet, SPL, and our method on the FF++(c23) in intra-dataset testing. In the third column, we can see that all AUC values are greater than 96% though our method has the highest AUC value, exactly 99.60%.

Let us examine the generalization capability of the baseline algorithm Xception first. Table 3 shows that, compared with 96.30% in the intra-dataset testing, it decreases by 13.14%, 28,40%, 36.84%, and 26.49% for DFD, DFDC, CDF and Deeper, respectively. Its average decrease in AUC values is 26.22%, indicating that the earliest detector severely lacks generalization capability. For the FTCN, SimMIM, CFFE and AdaGRnet, compared to the intra-dataset testing results, their average decrease in AUC values have been increased to -18.52%, -19.78%, -18.71%, and -21.46%, respectively. For the state-of-the-art methods like SBIs and SPL, with the help of self-made training data, their performance has been further improved, increasing to -15.31% and -15.12%, respectively. Our method achieves the best result and the average decrease is -10.74%, over 4% better than the second best method SPL, and over 15% better than the baseline method Xception.

Table 3. Cross-dataset evaluation. Our proposed method is compared with 7 methods in terms of AUC values (%). The third column lists the AUC values for intra-dataset testing. “-” denotes the decrease in AUC value relative to that in the third column.

Methods	Training set	Testing set FF++(c23)	DFD(c23)	DFDC	CDF	Deeper	Average decrease
Xception [1]		96.30	-13.14	-28.40	-36.84	-26.49	-26.22
FTCN [14]		99.26	-8.74	-19.39	-19.41	-26.64	-18.52
SimMIM [39]		96.21	-12.76	-21.86	-23.65	-20.86	-19.78
SBIs [25]	FF++(c23)	99.35	-5.79	-26.93	-9.23	-19.27	-15.31
CFFE [19]		97.63	-6.37	-25.54	-23.43	-19.48	-18.71
AdapGRnet [41]		99.48	-9.88	-23.29	-26.12	-26.54	-21.46
SPL [26]		99.50	-12.03	-19.32	-16.26	-12.87	-15.12
Ours		**99.60**	**-4.50**	**-16.54**	**-9.05**	**-12.85**	**-10.74**

Table 4. Further cross-dataset evaluation. Our proposed method is compared with 7 methods in terms of AUC values (%). Mixed⋆ in the third column denotes the mean of 4 intra-dataset evaluations because the mixed dataset is different for each cross-dataset evaluation. “-” denotes the decrease in AUC value compared to that in the third column.

Methods	Training set	Testing set Mixed⋆	DFD(c23)	DFDC	CDF	Deeper	Average decrease
Xception [1]		95.45	-10.82	-20.31	-27.33	-9.53	-17.00
FTCN [14]		98.13	-3.29	-11.61	-12.97	-10.89	-9.69
SimMIM [39]		98.08	-6.76	-12.83	-13.22	-9.74	-10.64
SBIs [25]	Mixed dataset	98.17	-2.45	-15.61	-5.10	-7.49	-7.66
CFFE [19]		97.60	-2.47	-9.13	-10.27	-7.77	-7.41
AdapGRnet [41]		98.20	-3.98	-12.67	-14.12	-11.10	-10.47
SPL [26]		98.23	-4.98	-10.93	-9.56	-4.01	-7.37
Ours		**98.56**	**-1.81**	**-8.74**	**-4.33**	**-3.46**	**-4.59**

4.4. Further experiment on generalization to unseen datasets

Using data from different datasets can allow one to better take advantage of the DG strategy. To further demonstrate the effectiveness of the proposed detection system, we conduct cross-dataset test by using the following strategy: we choose three of the four datasets {DFD, DFDC, CDF, Deeper} to form a large mixed training dataset with FF++(c23) and test the compared methods on the remaining dataset. Table 4 illustrates the AUC values of all the methods. The effect can be easily observed from the results. We first examine the results in intra-dataset testing. Although the AUC values for all 8 methods including ours, decrease by around 1% relative to Table 3, all 8 methods have exhibited significant improvement in cross-dataset tests. Compared to the results from the intra-dataset testing, the corresponding decreases are -17%, -9.69%, -10.64%, -7.66%, -10.47%, and -7.37% for Xception, FTCN, SimMIM, SBIs, CFFE, AdapGRnet, and SPL, respectively. Our method still performs best. It only decreases by 4.59% . Moreover, from the standard deviation of the decrease values {-1.81, -8.74, -4.33, -3.46}, we can easily notice that the performance is quite stable, indicating strong robustness against different datasets.

4.5. Ablation study

In this subsection, we investigate the contribution made by each module in detail so that we can better understand their roles in the framework. For comparison with the above results, we still use FF++(c23) as the training dataset and use DFD(c23), DFDC, CDF, and Deeper as the testing datasets. As shown in Table 5, we design 7 module combinations, namely {Co1, Co2, ..., Co7}, and we provide the corresponding experimental results. Co1, Co2, and Co3 are used to investigate the effect of the ALM, CLM, and FFCLM, which are the key parts of our detector. Co4 and Co5 are used to investigate the effect of the style perturbation operation, the SIG of our SFFS synthesizer, respectively. Co6 is utilized to study the effect of the complete SFFS. Co7 refers to our method with the complete setting.

Regarding the results for Co6, the SFFS makes the most significant contribution in terms of AUC values, which verifies that our alternating sample generation and model training can greatly extend the capability of our detector to recognize unseen data. Other obvious contributions are made by the CLM, SIG, ALM, and style perturbation, in order. Regarding the results for Co2, Co5, Co1, and Co4, there is similar impact to the improvement on generalization capability. In particular, the ALM and CLM contribute much to feature extraction. The style perturbation and SIG provide better training samples. Regarding the results for Co3, multi-task learning in the FFCLM can also improve the generalization capability to some degree.

Table 5. Ablation study for 6 key components in our method. Each configuration is trained on FF++(c23) and tested on DFD(c23), DFDC, CDF and Deeper. The AUC values (%) are reported.

Types	Modules	Training set	Testing set DFD(c23)	DFDC	CDF	Deeper	Average
Co1	w/o ALM		91.90	80.52	84.73	80.71	84.49
Co2	w/o CLM		91.68	80.15	84.70	80.66	84.30
Co3	w/o FFCLM		94.12	82.10	87.35	84.23	86.95
Co4	w/o SP	FF++(c23)	93.85	79.74	86.80	83.42	85.95
Co5	w/o SIG		92.20	78.55	85.32	81.62	84.42
Co6	w/o SFFS		91.36	76.69	83.38	78.80	82.56
Co7	Our Complete		**95.10**	**83.06**	**90.55**	**86.75**	**88.87**

4.6. *Further discussion*

4.6.1. Face forgery localization

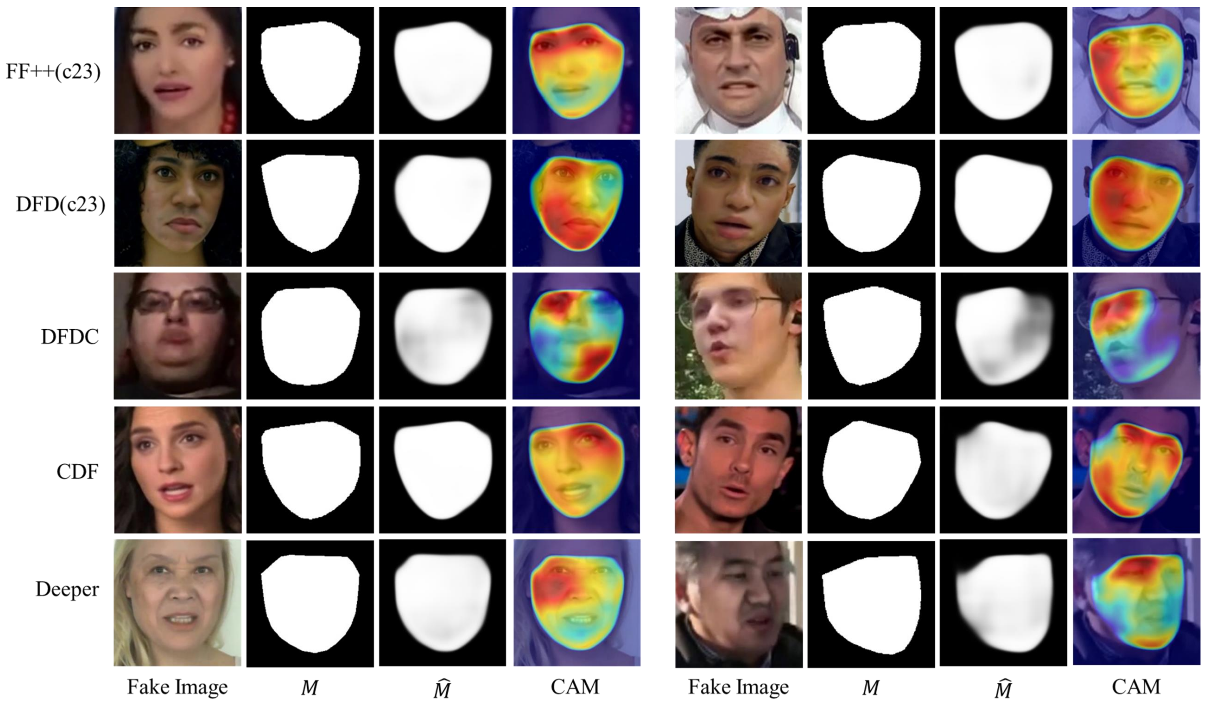


Figure 5. Illustrations of the estimated forgery masks and class activation maps (CAMs). Images are overlayed with the important attention regions highlighted with Grad-CAM [46].

In the FFCLM, we use Eq (3.9) to provide auxiliary supervision to improve detection accuracy. In this subsection, we visually demonstrate the effectiveness of our face forgery mask estimation and forgery localization method. In Figure 5, we first show the original forged face images from five public deepfake datasets employed in this work. We then list the ground-truth forgery masks $\boldsymbol{M}$ provided by datasets and our estimated masks $\hat{\boldsymbol{M}}$, respectively. Comparing $\hat{\boldsymbol{M}}$ with $\boldsymbol{M}$, we can easily find that our mask estimation method can work well. Even in profile face images, our method still achieves a relatively accurate forged localization result. We further give the class activation maps (CAMs) using Grad-CAM [46]. It can be observed that the forged region around the face is highlighted in all fake images, and that the regions with more obvious visual artifacts have higher attention values. These observations further verify that our method can accurately capture facial anomalies in forged face images.

4.6.2. Effect of β on generalization performance

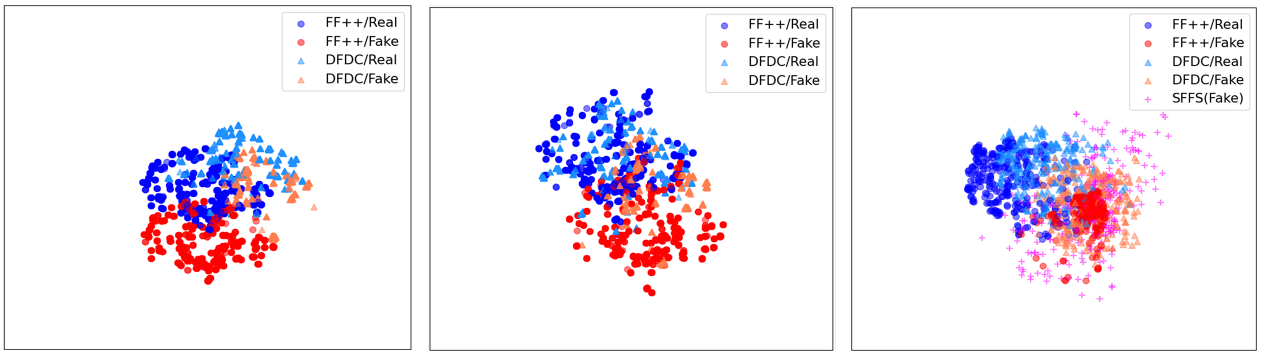
The interpolation weight β is a critical parameter to control style perturbation in the SFFS. Since the output of the SFFS is used as training data (see Figure 3) and dynamically takes part in the model training, β would affect the performance of our detector. We list the relationship between β and the performance of the detector in Table 6. We limit the range of β to [0, 0.3] to prevent the mixed style from being overly affected by random perturbation variables, which may alter the image content. It can be observed that when the range of β is increased from U(0, 0.05) to U(0, 0.10), the generalization performance improved. This is because better style diversity of samples is realized as β increases. However, as the range of β increases from U(0, 0.10) to U(0, 0.30), the detection performance decreases. The reason for this is that too large of a β value makes the variation in image styles overly dramatic, thereby introducing too much interference. To balance the style diversity and the interference in image content, we set β to U(0, 0.10).

Table 6. Evaluation with different interpolation weights β in terms of AUC values (%). The training dataset is FF++(c23), and the testing datasets are DFD(c23), DFDC, CDF, and Deeper.

Interpolation weight β	Training set	Testing set				Average
		DFD(c23)	DFDC	CDF	Deeper	
U(0, 0.05)	FF++(c23)	94.10	81.22	89.55	82.62	86.87
U(0, 0.10)		**95.10**	**83.06**	**90.55**	**86.75**	**88.87**
U(0, 0.15)		93.92	81.67	88.96	81.96	86.63
U(0, 0.20)		91.55	79.88	85.76	78.90	84.02
U(0, 0.25)		89.86	77.94	83.52	76.72	82.01
U(0, 0.30)		88.90	75.66	82.74	75.93	80.81

4.6.3. Learned feature spaces

As stated at the end of Section 2, we expect to find a good projection space so that the features from different target domains have more commonality. Now we shall go back to this topic and explain why the proposed detector can achieve better performance in different target domains. We still use the same two datasets as an example and shall visually illustrate the change in feature distributions. We first investigate the change in the distributions of style features. By comparing Figure 6(a) with Figure 1(c), we can easily find that the red and orange points representing fake faces have become closer, as with the blue and light blue points representing real faces, indicating that the intra-class distance decreases. In the meantime, the read and orange points are more widely separated from the blue and light blue points, indicating inter-distance increases. Obviously, the learned features have more expected distributions, which benefits the generalization. As for the content features, when we compare Figure 6(b) with Figure 2(c), we can obtain a conclusion similar to that for the style features. An astonishing improvement can be found in Figure 6(c), where the points representing real and fake faces are further separated. In addition, the pink "+" points that represent the features of our generated pseudo samples cover a larger area, which indicates that our detector can recognize more different unseen features from cross-domain scenarios. It is worth mentioning that the blue and light blue points representing real faces contract more compactly in this case.



(a) Learned style features in FF++ and DFDC (b) Learned content features in FF++ and DFDC (c) Learned classification features in FF++ and DFDC

Figure 6. Distributions of learned features with t-SNE visualization.

4.6.4. Effect of different video compression qualities of training data

The quality of training data may affect the performance of detectors. To evaluate the ability of our method to minimize this influence, we train our detector on the training data in three video compression protocols, i.e., raw format c0, good quality format c23, and poor quality format c40. We then test it on the CDF dataset. To emphasize the superiority of our method, we compare it with the same 7 literature methods as in Tables 3 and 4. Table 7 gives experimental results in terms of AUC values. It can be seen that, among the three protocols, each of the 8 methods achieve the highest AUC values in the format c23. We imagine that the video quality of CDF data is close to that of the format c23 since real videos in CDF are chosen from publicly available YouTube videos [43]. Videos shared on social media platforms like YouTube tend to be slightly compressed for the sake of storage and transmission. When trained on data in the format c0 and tested on CDF videos, all 8 methods have slightly lower AUC values than their counterparts on the format c23. This decrease is attributable to the quality difference between the training and testing datasets. The obvious loss of AUC values for all 8 methods take place when using the training data in the format c40. The reason is that the strong video compression operation in this protocol removes many useful forensic traces in the high frequency components of images. However, our method keeps the highest AUC values among the 8 methods for each protocol, indicating its good robustness against the change of compression quality of training data.

Table 7. Cross-dataset evaluation with different qualities of training data. The AUC values (%) are reported.

Methods	Testing on CDF Training on FF++(c0)	Training on FF++(c23)	Training on FF++(c40)
Xception [1]	58.32	59.46	54.60
FTCN [14]	78.69	79.85	74.82
SimMIM [39]	74.30	75.56	71.23
SBIs [25]	89.18	90.12	85.15
CFFE [19]	73.96	74.20	73.22
AdapGRnet [41]	72.25	73.36	73.57
SPL [26]	82.74	83.24	80.26
Ours	**89.93**	**90.55**	**86.65**

4.6.5. Computational complexity analysis

We analyze the computational complexity by comparing our method with the above 7 methods. The computational load is closely associated with the FLOPs and model parameters, so we make our comparison based on these metrics. The end-to-end detectors include our method and [1, 14, 19, 39, 41]. Table 8 shows that the detector using the Vision Transformer (ViT) as the backbone, as in [39], significantly increases the computational complexity relative to the convolutional neural networks (CNN)-based detectors, including those in [1, 14, 19, 41]. Our method uses the Swin-S (an improved ViT) network as the backbone and thus requires more parameters and FLOPs than [1, 14, 19, 41] but less than [39]. The SBIs [25] and SPL [26] are not end-to-end detectors in that their data generation and detection processes are separately carried out. Therefore, we are only concerned about the complexity of the detection process. We can observe that SPL (i.e., the second best method in cross-dataset evaluation) has the highest computational complexity. This is because SPL uses the large ViT model Swin-L [38]. Our method has less computational complexity than Swin-L based SPL but more than CNNs-based SBIs.

Table 8. Computational complexity comparison for different detection methods.

Methods	Input Image Size	Parameters (Megabytes)	FLOPs (Gigabytes)
Xception [1]	$1 \times 3 \times 256 \times 256$	20.81	6.01
FTCN [14]		14.75	11.99
SimMIM [39]		197.00	48.95
SBIs [25]		19.00	3.12
CFFE [19]		26.50	10.83
AdapGRnet [41]		11.19	5.72
SPL [26]		198.50	52.90
Ours		66.31	15.53

4.6.6. Effect of using different encoders in the CSFEM

We further investigate the effect of using different encoders in the CSFEM. Exactly, we use four typical detection networks, namely, Xception [1], ResNet-50 [47], EfficientNet-B4 [48], and Swin-S [38], and apply their feature extraction layers as the encoder one by one. The features extracted with different encoders are reshaped to have the same dimensions, i.e., $B \times 768 \times \left\lceil \frac{H_0}{32} \right\rceil \times \left\lceil \frac{W_0}{32} \right\rceil$, and then fed to the CFE and SFE branches separately. The cross-dataset evaluation results are listed in Table 9. It can be seen that all four encoding networks realize good generalization performance, indicating the effectiveness of our proposed framework. Among them, the use of Swin-S results in the highest AUC values.

Table 9. Further investigation of using 4 different encoders in our CSFEM. The AUC values (%) are reported.

Encoder in CSFEM	Training set	Testing set				Average
		DFD(c23)	DFDC	CDF	Deeper	
Xception	FF++(c23)	93.03	79.58	85.60	83.26	85.37
ResNet-50		92.42	80.26	87.15	83.51	85.84
EfficientNet-B4		93.36	81.84	86.67	84.06	86.48
Swin-S (Our used)		**95.10**	**83.06**	**90.55**	**86.75**	**88.87**

5. Conclusions

In this paper, we have proposed a new way to design deepfake detectors so as to make them generalize well to cross-domain scenarios. Instead of using the aimless relation between the deepfake detector and pseudo training data generator, our deepfake detector and the training data generator work together and alternately update the network parameters. For better detection, we construct a more diverse feature space by reassembling different content and style features. Furthermore, the contrastive learning strategy has been employed to emphasize generic style information while the adversarial learning strategy has been used to obtain a domain-invariant content representation. For pseudo training data generation, a feature-level data augmentation method has been devised to enrich style diversity and prevent the detector from over-relying on specific styles in the training data. Extensive experiments have shown that our method outperforms other related state-of-the-art methods in both the intra-dataset testing and cross-domain scenarios. The generalization capability has been significantly improved due to the jointly learning and training mechanism.

Use of AI tools declaration

The authors declare they have not used artificial intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the Science and Technology Foundation of Guangzhou Huangpu Development District (Grant No. 2022GH15), and the Open Foundation of Henan Key Laboratory of Cyberspace Situation Awareness (Grant No. HNTS2022017).

Conflict of interest

The authors declare that there are no conflicts of interest.

References

1. F. Chollet, Xception: Deep learning with depthwise separable convolutions, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2017), 1800–1807. https://doi.org/10.1109/CVPR.2017.195
2. B. Bayar, M. C. Stamm, Constrained convolutional neural networks: a new approach towards general purpose image manipulation detection, *IEEE Trans. Inf. Forensics Secur.*, **13** (2018), 2691–2706. https://doi.org/10.1109/TIFS.2018.2825953
3. R. Durall, M. Keuper, J. Keuper, Watch your up-convolution: CNN based generative deep neural networks are failing to reproduce spectral distributions, in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2020), 7887–7896. https://doi.org/10.1109/CVPR42600.2020.00791
4. Y. Qian, G. Yin, L. Sheng, Z. Chen, J. Shao, Thinking in frequency: face forgery detection by mining frequency-aware clues, in *Computer Vision – ECCV 2020*, Springer, (2020), 86–103. https://doi.org/10.1007/978-3-030-58610-2_6
5. H. Zhao, W. Zhou, D. Chen, T. Wei, W. Zhang, N. Yu, Multi-attentional deepfake detection, in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2021), 2185–2194. https://doi.org/10.1109/CVPR46437.2021.00222
6. Y. Luo, Y. Zhang, J. Yan, W. Liu, Generalizing face forgery detection with high-frequency features, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2021), 16317–16326.
7. J. Yang, A. Li, S. Xiao, W. Lu, X. Gao, Mtd-net: learning to detect deepfakes images by multi-scale texture difference, *IEEE Trans. Inf. Forensics Secur.*, **16** (2021), 4234–4245. https://doi.org/10.1109/TIFS.2021.3102487
8. B. Chen, W. Tan, Y. Wang, G. Zhao, Distinguishing between natural and GAN-generated face images by combining global and local features, *Chin. J. Electron.*, **31** (2022), 59–67. https://doi.org/10.1049/cje.2020.00.372
9. G. Li, X. Zhao, Y. Cao, Forensic symmetry for deepfakes, *IEEE Trans. Inf. Forensics Secur.*, **18** (2023), 1095–1110. https://doi.org/10.1109/TIFS.2023.3235579
10. P. Zhou, X. Han, V. I. Morariu, L. S. Davis, Two-stream neural networks for tampered face detection, in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, (2017), 1831–1839. https://doi.org/10.1109/CVPRW.2017.229

11. L. Li, J. Bao, T. Zhang, H. Yang, D. Chen, F. Wen, et al., Face x-ray for more general face forgery detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2020), 5001–5010.

12. B. Chen, X. Liu, Y. Zheng, G. Zhao, Y. Q. Shi, A robust GAN-generated face detection method based on dual-color spaces and an improved xception, *IEEE Trans. Circuits Syst. Video Technol.*, **32** (2022), 3527–3538. https://doi.org/10.1109/TCSVT.2021.3116679

13. S. Chen, T. Yao, Y. Chen, S. Ding, J. Li, R. Ji, Local relation learning for face forgery detection, in *AAAI Technical Track on Computer Vision I*, **35** (2021), 1081–1088. https://doi.org/10.1609/aaai.v35i2.16193

14. Y. Zheng, J. Bao, D. Chen, M. Zeng, F. Wen, Exploring temporal coherence for more general video face forgery detection, in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, (2021), 15024–15034. https://doi.org/10.1109/ICCV48922.2021.01477

15. G. Pang, B. Zhang, Z. Teng, Z. Qi, J. Fan, Mre-net: Multi-rate excitation network for deepfake video detection, *IEEE Trans. Circuits Syst. Video Technol.*, **33** (2023), 3663–3676. https://doi.org/10.1109/TCSVT.2023.3239607

16. B. Chen, T. Li, W. Ding, Detecting deepfake videos based on spatiotemporal attention and convolutional LSTM, *Inf. Sci.*, **601** (2022), 58–70. https://doi.org/10.1016/j.ins.2022.04.014

17. H. Liu, X. Li, W. Zhou, Y. Chen, Y. He, H. Xue, et al., Spatial-phase shallow learning: rethinking face forgery detection in frequency domain, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2021), 772–781.

18. J. Cao, C. Ma, T. Yao, S. Chen, S. Ding, X. Yang, End-to-end reconstruction-classification learning for face forgery detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2022), 4113–4122.

19. P. Yu, J. Fei, Z. Xia, Z. Zhou, J. Weng, Improving generalization by commonality learning in face forgery detection, *IEEE Trans. Inf. Forensics Secur.*, **17** (2022), 547–558. https://doi.org/10.1109/TIFS.2022.3146781

20. A. Luo, C. Kong, J. Huang, Y. Hu, X. Kang, A. C. Kot, Beyond the prior forgery knowledge: mining critical clues for general face forgery detection, *IEEE Trans. Inf. Forensics Secur.*, **19** (2024), 1168–1182. https://doi.org/10.1109/TIFS.2023.3332218

21. J. Hu, X. Liao, W. Wang, Z. Qin, Detecting compressed deepfake videos in social networks using frame-temporality two-stream convolutional network, *IEEE Trans. Circuits Syst. Video Technol.*, **32** (2022), 1089–1102. https://doi.org/10.1109/TCSVT.2021.3074259

22. T. Wang, K. P. Chow, Noise based deepfake detection via multi-head relative-interaction, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **37** (2023), 14548–14556. https://doi.org/10.1609/aaai.v37i12.26701

23. Y. Li, S. Lyu, Exposing deepfake videos by detecting face warping artifacts, in *2019 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, (2019), 46–52.

24. T. Zhao, X. Xu, M. Xu, H. Ding, Y. Xiong, W. Xia, Learning self-consistency for deepfake detection, in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, (2021), 15003–15013. https://doi.org/10.1109/ICCV48922.2021.01475

25. K. Shiohara, T. Yamasaki, Detecting deepfakes with self-blended images, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2022), 18699–18708. https://doi.org/10.1109/CVPR52688.2022.01816

26. H. Chen, Y. Lin, B. Li, S. Tan, Learning features of intra-consistency and inter-diversity: keys toward generalizable deepfake detection, *IEEE Trans. Circuits Syst. Video Technol.*, **33** (2023), 1468–1480. https://doi.org/10.1109/TCSVT.2022.3209336

27. A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, M. Niessner Faceforensics++: Learning to detect manipulated facial images, in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, (2019), 1–11. https://doi.org/10.1109/ICCV.2019.00009

28. B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, et al., The deepfake detection challenge (DFDC) dataset, preprint, arXiv:2006.07397.

29. L. V. der Maaten, G. Hinton, Visualizing data using t-SNE, *J. Mach. Learn. Res.*, **9** (2008), 2579–2605. Available from: https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf?fbcl.

30. Z. Wang, Z. Wang, Z. Yu, W. Deng, J. Li, T. Gao, Domain generalization via shuffled style assembly for face anti-spoofing, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2022), 4113–4123. https://doi.org/10.1109/CVPR52688.2022.00409

31. Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in *Proceedings of the 32nd International Conference on Machine Learning*, **37** (2015), 1180–1189. Available from: http://proceedings.mlr.press/v37/ganin15.html.

32. Y. Wang, L. Qi, Y. Shi, Y. Gao, Feature-based style randomization for domain generalization, *IEEE Trans. Circuits Syst. Video Technol.*, **32** (2022), 5495–5509. https://doi.org/10.1109/TCSVT.2022.3152615

33. S. Lin, C. T. Li, A. C. Kot, Multi-domain adversarial feature generalization for person re-identification, *IEEE Trans. Image Process.*, **30** (2021), 1596–1607. https://doi.org/10.1109/TIP.2020.3046864

34. H. Nam, H. Lee, J. Park, W. Yoon, D. Yoo, Reducing domain gap by reducing style bias, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2021), 8690–8699.

35. Q. Zhou, K. Y. Zhang, T. Yao, X. Lu, R. Yi, S. Ding, et al., Instance-aware domain generalization for face anti-spoofing, in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2023), 20453–20463. https://doi.org/10.1109/CVPR52729.2023.01959

36. X. Huang, S. Belongie, Arbitrary style transfer in real-time with adaptive instance normalization, in *2017 IEEE International Conference on Computer Vision (ICCV)*, (2017), 1510–1519. https://doi.org/10.1109/ICCV.2017.167

37. T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2019), 4396–4405. https://doi.org/10.1109/CVPR.2019.00453

38. Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, et al., Swin transformer: hierarchical vision transformer using shifted windows, in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, (2021), 10012–10022.

39. Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, et al., Simmim: a simple framework for masked image modeling, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2022), 9653–9663.

40. D. E. King, Dlib-ml: a machine learning toolkit, *J. Mach. Learn. Res.*, **10** (2009), 1755–1758. Available from: https://www.jmlr.org/papers/volume10/king09a/king09a.pdf.

41. Z. Guo, G. Yang, J. Chen, X. Sun, Exposing deepfake face forgeries with guided residuals, *IEEE Trans. Multimedia*, **25** (2023), 8458–8470. https://doi.org/10.1109/TMM.2023.3237169

42. M. Schroepfer, Creating a data set and a challenge for deepfakes, in *Facebook Artificial Intelligence*, **5** (2019). Available from: https://ai.facebook.com/blog/deepfake-detection-challenge.

43. Y. Li, X. Yang, P. Sun, H. Qi, S. Lyu, Celeb-df: a large-scale challenging dataset for deepfake forensics, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2020), 3207–3216.

44. L. Jiang, R. Li, W. Wu, C. Qian, C. C. Loy, Deeperforensics-1.0: a large-scale dataset for real-world face forgery detection, in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2020), 2886–2895. https://doi.org/10.1109/CVPR42600.2020.00296

45. G. Bradski, The opencv library, in *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, **25** (2000), 120–123.

46. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: visual explanations from deep networks via gradient-based localization, in *2017 IEEE International Conference on Computer Vision (ICCV)*, (2017), 618–626. https://doi.org/10.1109/ICCV.2017.74

47. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2016), 770–778.

48. M. Tan, Q. Le, Efficientnet: rethinking model scaling for convolutional neural networks, in *Proceedings of the 36th International Conference on Machine Learning*, (2019), 6105–6114.