# Self-supervised Learning of Adversarial Example: Towards Good Generalizations for Deepfake Detection

Liang Chen[1] Yong Zhang[2*] Yibing Song[2] Lingqiao Liu[1*] Jue Wang[2]
[1] The University of Adelaide [2] Tencent AI Lab
{liangchen527, zhangyong201303, yibingsong.cv, arphid}@gmail.com
lingqiao.liu@adelaide.edu.au

## Abstract

*Recent studies in deepfake detection have yielded promising results when the training and testing face forgeries are from the same dataset. However, the problem remains challenging when one tries to generalize the detector to forgeries created by unseen methods in the training dataset. This work addresses the generalizable deepfake detection from a simple principle: a generalizable representation should be sensitive to diverse types of forgeries. Following this principle, we propose to enrich the "diversity" of forgeries by synthesizing augmented forgeries with a pool of forgery configurations and strengthen the "sensitivity" to the forgeries by enforcing the model to predict the forgery configurations. To effectively explore the large forgery augmentation space, we further propose to use the adversarial training strategy to dynamically synthesize the most challenging forgeries to the current model. Through extensive experiments, we show that the proposed strategies are surprisingly effective (see Figure 1), and they could achieve superior performance than the current state-of-the-art methods. Code is available at https://github.com/liangchen527/SLADD.*

## 1. Introduction

The realistic image generation brought by the generative adversarial network (GAN) raises a security issue that human portraits can be easily substituted to provide malicious bioinformatics [11, 45, 15, 44, 51, 56, 40, 54, 48]. This forgery becomes a threat to subject identifications which have been extensively utilized in digital payment, video surveillance, and social media. To reduce these risks, there is an emerging investigation on deepfake detectors to identify face forgeries. Formulated as a binary classification problem, current detectors [26, 3, 2, 32, 39] perform well when training and testing forgeries are synthesized from the same dataset and same forgery methods. However, in practice the testing forgeries are usually from unseen datasets and synthesized by unseen methods. Discrepancies between training and testing data lead to inferior performance of detectors. There is a performance drop when detectors recognize forgeries outside the training dataset, which brings challenges to deepfake detectors for practical usage.

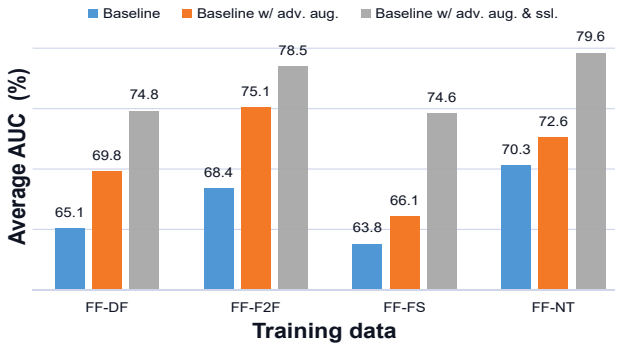



Figure 1. Performance improvements of the proposed strategies for the baseline model (*i.e.* Xception [41]). The models are trained on the four types of data from Faceforensics++ dataset [41] and tested on CelebDF [28], DFDC [12], and DF1.0 [20] datasets.

Attempts have been made in recent arts to improve the generalizations. For example, to overcome dataset bias, some studies [24, 58] suggest data augmentation is an effective tool against poor generalization. These methods augment training data by synthesizing new face forgeries with their empirically designed augmentations. However, their augmentations are with a limited choice of strategy. The lacking of variety may jeopardize the generalization. Meanwhile, intrinsic forgery attributes shared between forgeries, such as detail discrepancies [30], and frequency features [39, 29], are also mined broadly with hand-crafted representations to distinct forgeries. But these attributes mainly rely on imperceptible image patterns, which are sensitive to post-processing steps, such as compression. They vary significantly in different datasets, thus leading to large detection bias and limiting their generalization [60].

*Corresponding authors. This work is done when L. Chen is an intern in Tencent AI Lab.

In this paper, we address the deepfake detection from a simple heuristic principle: *a generalizable representation should be sensitive to the various types of forgeries.* Training a model by following this principle could potentially avoid the "blind spot" of the model or avoid relying on patterns specific to a dataset, since otherwise the model may not be capable of identifying a variety of forgeries. Pushing this idea to the limit, we propose to enrich the "diversity of forgeries" by synthesizing forgery images from a large pool of configurations [1]. Specifically, given a pristine, we randomly choose a reference image [2] from training data, our synthesizer network (*i.e*. generator) produces forgery configurations that specify the forgery region, the blending type, and the blending ratio. Then based on these configurations, a synthesized forgery is generated (some examples are shown in Figure 2). To enhance the "sensitivity to the forgeries", our detector network (*i.e*. discriminator) is required to predict the configurations of an input in addition to judging if it is a forgery or not. Further, to effectively explore the large space of forgery augmentation, we adopt an adversarial training strategy to dynamically construct the augment that is most challenging to the current detector network. Different from the empirically designed augmentations [24, 58, 6], our adversarial augmentation strategy enjoys a large variety, and it can be dynamically constructed by the performance of the discriminator.

Through our experimental studies, we demonstrate the effectiveness of employing both adversarial augmentation and self-supervised tasks. A significant improvement over the baseline approach is observed, and our method also performs favorably against other state-of-the-art detectors.

## 2. Related Works

**Deepfake detection.** Recent studies have made various attempts for deepfake detection and achieve remarkable success [26, 41, 3, 2, 32, 39, 14, 24, 30, 29, 19, 47, 23, 36, 5]. Several arts discuss low-level differences between pristines and forgeries and suggest using them as classification clues. Li *et al*. [24] assume blending artifacts exist in all pristines and suggest finding the blending boundary beside the detection task; Qian *et al*. [39] and Luo *et al*. [30] use high-frequency details as additional inputs for their models; Liu *et al*. [29] adopt phase spectrums to capture the up-sampling artifacts of face forgery for the task. Despite their effectiveness in many cases, the low-level artifacts are sensitive to post-processing steps which vary in different datasets, thus limiting their generalization [60]. Some works suggest borrowing features from other tasks for deepfake detection.

[1]The term **configuration** means a specific way of synthesizing a forgery image. In the context of our discussion, it can also refer to a set of parameters that control a particular synthesizing process.

[2]We only use the forgery image in the training set as reference.

(a) Pristine (b) Reference (c) Forgery (adv)

Figure 2. Examples of the input pristine, reference images, and their corresponding synthesized adversarial forgeries.

Such as the features from lips reading [19], facial image decomposition [60], and landmark geometric features [43]. Although these features can bring promising improvements, their generalization performance to the deepfake data are questionable. While annotating the deepfake data for these tasks is rather expensive, incorporating these tasks often lead to limited improvements.

**Adversarial learning.** Adversarial training aims to use adversarial examples for augmenting the training samples, which often contains a generator and a discriminator, and they are trained on the principle of defending against adversarial attacks [17, 22, 31, 55]. Goodfellow *et al*. [17] first use the fast gradient sign method to improve the adversarial robustness. Madry *et al*. [22] further propose a multi-step scheme termed as projected gradient descent to enable powerful robustness compared to other works. Moreover, different from previous works that use adversarial training to enlarge datasets through directly synthesizing new images [46, 37, 4, 18, 16], Zhang *et al*. [55] suggest automatically choosing augmentation policies in an adversarial manner, which shows a great reduction in computing cost than the previous method [10]. However, most of these adversarial training strategies are designed for general tasks, such as classification. In contrast, our adversarial example synthesizing process is similar to the deepfake generation procedure, thus is more suitable for the deepfake detection task.

## 3. Proposed Method

We propose to improve the generalizability of deepfake detector with the help of adversarial data augmentation,

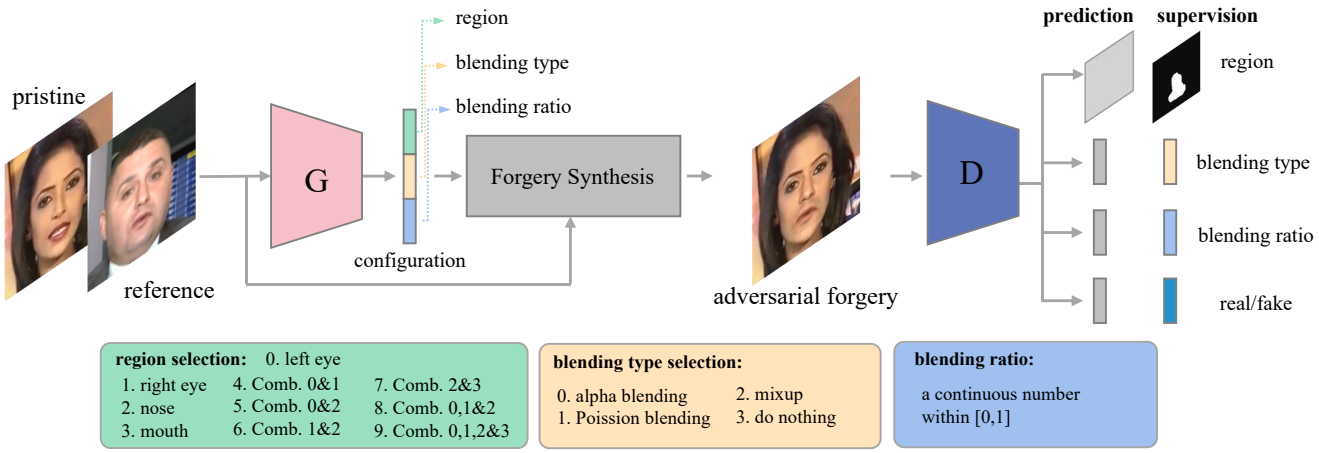



Figure 3. Overview of our model. The synthesizer network (*i.e.* generator) outputs three forgery configurations that are further used to synthesize a new forgery, and these forgery configurations are also used as labels to guide the detector network (*i.e.* discriminator). We train the generator and discriminator in an adversarial manner. Please refer to the text for details.

which enriches the types of forgeries, and self-supervised tasks that enforce sensitivity to forgery configurations. The pipeline of our model is shown in Figure 3. During training, we randomly select a reference image if the input is pristine. These two images are sent to a synthesizer network to produce configurations that specify the facial forgery region, the blending type, and the mixup blending ratio (if mixup blending is selected). Then a new forgery is synthesized based on these two images and the selected configurations. The synthesized forgery is passed to our detector for the real or fake and configurations predictions. Note the input will skip the forgery synthesizing process if it is an original forgery from the training data. We adopt an adversarial training strategy to train the system, in which the synthesizer is regarded as the generator, and the detector is regarded as the discriminator. The training process will train the forgery synthesizer to produce new forgeries to challenge the detector. Once trained, only the detector is used for deepfake detection.

### 3.1. Selecting Space and Synthesizing Forgery

Our synthesizer network $G(\cdot, \theta)$ takes a pristine $\mathbf{I}_p \in \mathbb{R}^{H\times W\times 3}$ and a reference image $\mathbf{I}_f \in \mathbb{R}^{H\times W\times 3}$ as inputs and output three configurations: the forgery region reference index $R_g$, the blending type index $T_g$ (a discrete number), and the mix-up blending ratio $A_g$ (a continuous value). For $A_g$, we directly generate a scalar from the input $\mathbf{I}_p$ and $\mathbf{I}_f$. For $R_g$, $T_g$, we first generate two probability distributions $p(R_g)$ and $p(T_g)$, indicating the chance of selecting a particular index of $R_g$ and $T_g$, and then randomly sample a index according to the probability. For the simplicity of discussion, we use $p_m$ to denote those two probabilities. These configurations, $R_g$, $T_g$ and $A_g$, control how a forgery image is generated. Their definitions are as follows:

$R_g$: The index $R_g \in \{0, \cdots, 9\}$ determines a specific facial region. We divide the facial image into 10 regions based on its landmark, including the left eye, right eye, nose, mouth, and their combinations. Examples of different regions are shown in Figure 4 (c) - (l). We only consider the facial features because most deepfake arts focus on them, and they convey the most information in a facial image;

$T_g$: The reference number $T_g \in \{0, \cdots, 3\}$ determines the blending type. We use three blending techniques, including alpha, Poisson [38], and mixup blending. Without the loss of generality, we include a do-nothing choice among the blending type pool, so that we can use the original pristine from the dataset for further classification.

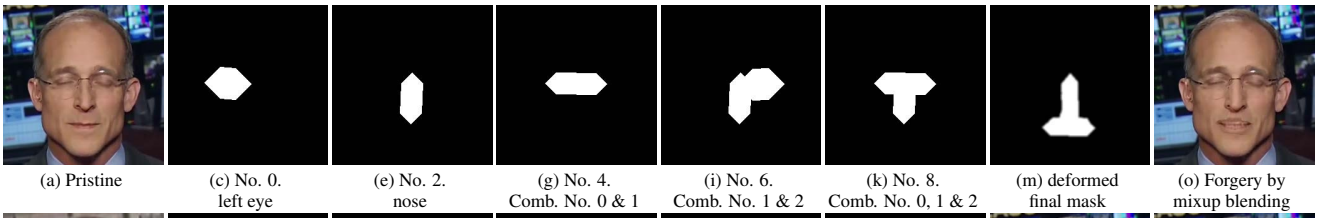
$A_g$: The continuous value $A_g \in [0, 1]$ is the blending ratio which is only effective when mixup blending is selected.

We adopt a random shape deformation operation and apply Gaussian blur with random kernel size to the selected forgery region before the synthesizing step. An example of the final mask is shown in Figure 4 (m). Then, to synthesize the adversarial forgery $\mathbf{I}_a$, we crop the facial parts determined by $R_g$ from $\mathbf{I}_f$, and blend it into $\mathbf{I}_p$ using the configurations indicated by the synthesizer network. Specifically, we use the official implementations from OpenCV for alpha and Poisson blendings to synthesize $\mathbf{I}_a$, and for mixup blending, $\mathbf{I}_a$ can be obtained by:

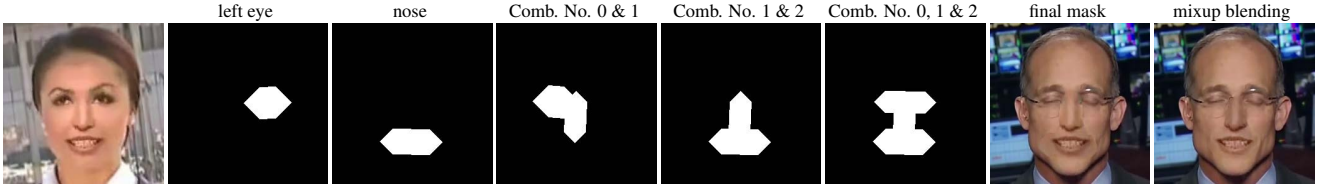
$$\mathbf{I}_a = A_g \times \mathbf{M}_d * (\mathbf{I}_f - \mathbf{I}_p) + \mathbf{I}_p, \tag{1}$$

where $\mathbf{M}_d$ is the deformed final mask, and $*$ is the Hadamard product.

Like existing deepfake methods, the cropped facial parts are applied with color transfer (aligning the mean of the RGB channels, respectively) and face alignment before the alpha and Poisson blending steps, which can avoid obvious artifacts in the newly synthesized forgery. Examples of newly synthesized forgeries with different blending types are shown in Figure 4 (n) - (p).



(a) Pristine (c) No. 0. left eye (e) No. 2. nose (g) No. 4. Comb. No. 0 & 1 (i) No. 6. Comb. No. 1 & 2 (k) No. 8. Comb. No. 0, 1 & 2 (m) deformed final mask (o) Forgery by mixup blending



(b) Random reference (d) No. 1. right eye (f) No. 3. mouth (h) No. 5. Comb. No. 0 & 2 (j) No. 7. Comb. No. 2 & 3 (l) No. 9. Comb. No. 0, 1, 2 & 3 (n) Forgery by alpha blending (p) Forgery by Poisson blending

Figure 4. An example of the pristine and randomly chosen reference images (*i.e.* (a) and (b)), the corresponding selecting space of the manipulate regions (*i.e.* (c) - (m)), and the adversarial forgeries synthesized by different blending types (*i.e.* (n) - (p)) under the instruction of the deformed manipulate region map (*i.e.* (m)). The selected region number is 7. in this example, and the blending ratio for (o) is 0.5.

### 3.2. Joint Training with Self-Supervised Tasks

Self-supervised tasks have been shown effective for improving generalization in previous works [7, 50]. However, those attempts often consider self-supervised learning tasks proven effective for image classification tasks. In contrast to them, this paper employs an auxiliary self-supervised task specifically designed for the deepfake detection problem. By doing so, we expect that the auxiliary task aligns better with the target task and thus could lead to better performance. Our idea follows a multi-task scheme that allows the model to simultaneously learn from two tasks, the main task and a set of auxiliary self-supervised task. Specifically, for the auxiliary tasks, we consider forgery region estimation, blending type estimation and blending ratio estimation. We elaborate on each task and the loss in the following.

**Main task loss** $\mathcal{L}_{Main}$. The main task is a binary classification task that predicts whether a given input image is pristine or forgery. Similar to previous work [30], we adopt the AM-Softmax Loss [49] to compute $\mathcal{L}_{Main}$ since it allows for smaller intra-class variations and larger inter-class differences than the regular cross-entropy loss.

**Forgery region estimation loss** $\mathcal{L}_R$. We also create a forgery region estimation task. Specifically, a forgery region prediction head is attached to the detector network and generates forgery region mask with the size of $[H/16 \times W/16]$. The ground-truth forgery mask $\mathbf{M}_{gt}$ is created depending on categories of the input images, which is divided into three categories. Specifically, if the input image is the adversarial forgery, the corresponding $\mathbf{M}_{gt}$ is the resized deformed final mask (*i.e.* $\mathbf{M}_{gt} = \mathbf{M}_d$); As most datasets provide the ground truth forgery region, we can directly use them as $\mathbf{M}_{gt}$ if the input image is an original forgery from the training dataset; If the input image is an original pristine from the training dataset, the $\mathbf{M}_{gt}$ is an all-zero matrix (*i.e.* $\mathbf{M}_{gt} = \mathbf{0}$), indicating there is no forgery region in the input image. An $L_1$ loss is applied for the task:

$$\mathcal{L}_R = \frac{\|\mathbf{M}_{gt} - \mathbf{M}_e\|_1}{H/16 \times W/16}, \tag{2}$$

where $\mathbf{M}_e$ is the estimated region map from our detector network, and $\|\cdot\|_1$ denotes the $L_1$ norm.

**Blending type estimation loss** $\mathcal{L}_T$. We then introduce a blending type estimation task that aims to estimate the blending type of input images. Similar to the forgery region estimation task, the ground-truth of the prediction target, i.e., ground-truth blending type $T_{gt}$, for this task varies accordingly to the category of the input data. When the input is a synthesized adversarial forgery, $T_{gt}$ can be directly borrowed from the output of the generator (*i.e.* $T_{gt} = T_g$, where $T_g \in \{0, 1, 2\}$ according to the selected blending type); If the input is the original pristine, we set $T_{gt} = 3$, implying there is no blending operation in the image; As the blending types are often unavailable in the existing datasets, we thus set $T_{gt} = 4$ if the input image is the original forgeries from the training dataset, indicating a blending type outside of our selecting space. Similarly, we adopt the AM-Softmax Loss [49] to compute $\mathcal{L}_T$.

**Blending ratio estimation loss** $\mathcal{L}_A$. To take full advantage of the blending information provided in the forgery synthesizing process, we further construct a task to estimate the blending ratio output by the synthesizer network. Because the blending ratio is only effective when the synthesizer network selects the mixup blending, we can set the prediction ground-truth $A_{gt} = A_g$ when the input image is a synthesized adversarial forgery using mixup blending. Correspondingly, this loss is not computed when the input image is either the original images in the training dataset or adversarial forgery synthesized with other blending types. An $L_1$ loss is adopted to compute $\mathcal{L}_A$:

$$\mathcal{L}_A = \tau \times \|A_{gt} - A_e\|_1, \tag{3}$$

where $A_e$ is the estimated blending ratio from the detector network, and $\tau$ is a binary value, which s.t. $\tau = 1$ if $T_{gt} = 2$ (*i.e.* the blending type is mixup blending) and $\tau = 0$ otherwise, thus ensuring $\mathcal{L}_A$ is only effective when adopting the mixup blending in the forgery synthesizing process.

### 3.3. Adversarial Training

To better leveraging the forgery augmentation space, we adopt adversarial training to dynamically construct the most challenging auxiliary task. Specifically, we use the synthesizer network $G(\cdot, \theta)$ as the generator, which maximizes the training loss of the target discriminator (*i.e.* the detector network $D(\cdot, w)$) through adversarial learning. The optimization process can be presented as,

$$\min_{w} \max_{\theta} \mathcal{L}, \ \text{ s.t. } \mathcal{L}(\theta, w) = \mathcal{L}_{Main} + \alpha\mathcal{L}_R + \mu\mathcal{L}_T + \gamma\mathcal{L}_A, \tag{4}$$

where $\alpha$, $\mu$, and $\gamma$ are weight hyper-parameters. Recall that $\theta$ denotes all the parameters in the generation process. Following the common practice of adversarial training, the above optimization problem can be approximately solved by iteratively updating the discriminator and the generator.

We first present the learning process of the discriminator. Eq. (4) is a minimization problem regarding $w$. By fixing the current generator parameter $\theta$, with the learning rate of $\eta$ and batch size of $N$, we perform gradient descent update:

$$w^{t+1} = w^t - \eta \frac{1}{N} \sum_{n=1}^{N} \nabla_{w^t} \mathcal{L}_n(\theta^t, w^t), \tag{5}$$

where $\mathcal{L}_n$ is the loss for the $n$-th sample in a mini-batch.

The generator is designed to increase the training loss of the target discriminator by synthesizing more challenging samples than the original benign examples, thus encouraging the discriminator to learn more generalizable features. It plays a zero-sum game in an adversarial framework with the discriminator. Mathematically, we can formulate the object as a maximizing problem according to Eq. (4),

$$\theta^{t+1} = arg \max_{\theta^t} \mathcal{L}(w^{t+1}, \theta^t). \tag{6}$$

However, directly updating $\theta$ by solving Eq. (6) could be problematic because there exist non-differentiable sampling operations that will break gradient flow from $D(\cdot, w)$ to $G(\cdot, \theta)$. To handle that, we apply REINFORCE algorithm [52] to approximate the gradient calculation for $\theta$:

$$\begin{aligned} \theta^{t+1} &= \theta^t + \epsilon \nabla_{\theta^t} \mathcal{L}_b \\ &\approx \theta^t + \epsilon \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}_b \nabla_{\theta^t} \log p_m, \end{aligned} \tag{7}$$

where $\mathcal{L}_b = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}_n(w^{t+1}, \theta^t)$, $M$ denotes the number of selected configuration series in a batch. $p_m$ represents the probabilities of generating $R_g$ and $T_g$, which is estimated from the sythesizer network $G(\cdot, \theta)$.

## 4. Experiments

### 4.1. Settings

**Training datasets**. Following recent deepfake detection methods [39, 24, 30, 29, 47, 23], we train our model in the Faceforencis++ (FF++) dataset [41]. It contains 1,000 original videos, where 720 videos are used for training, 140 videos are reserved for testing and the rest for validation. All videos undergo four state-of-the-art deepfake methods, which includes Deepfakes (DF) [11], Face2Face (F2F) [45], FaceSwap (FS) [15], and NeuralTextures (NT) [44]. Final outputs are generated with different compression levels including RAW, High Quality (HQ) and Low Quality (LQ), respectively. We use the HQ and LQ versions in our experiments, and the HQ version is adopted by default unless otherwise stated.

**Testing datasets**. To evaluate the generalizability of our method, we use the following benchmark datasets: CelebDF [28], which contains 408 real videos and 795 synthesized videos that are generated by the improved deepfake technology; Deepfake Detection Challenge (DFDC) [12], which includes over 1,000 real and over 4,000 fake videos manipulated by multiple Deepfake, GANbased, and non-learned methods; DeeperForensics-1.0 (DF1.0) [20], which consists of over 11,000 fake videos generated by their DF-VAE [20] method.

We use DLIB [42] for face extraction and alignment, and we resize the aligned faces to $256 \times 256$ for all the samples in training and test datasets.

**Implemenation details**. We modify Xception [8] as the backbones for our synthesizer and detector networks, and their parameters are initialized by Xception pre-trained on ImageNet. The hyper-parameter used in (4) are $\alpha = 0.1$, $\mu = 0.05$, and $\gamma = 0.1$. We use the Adam optimizer [21] for both the two networks with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and the batch size is fixed as 32. The learning rates are set to $2 \times 10^{-4}$ and $5 \times 10^{-5}$ for the detector and synthesizer networks, respectively.

### 4.2. Generalizability Comparisons

To comprehensively evaluate the generalizability of our method, we compare several state-of-the-art methods including Xception [41], Face X-ray [24], F3Net [39], RFM [47], and SRM [30]. To ensure fair comparisons, we use the provided codes of Xception [41], RFM [47], and SRM [30] from the authors, and we reimplement Face X-ray [24] and F3Net [39] rigorously following the companion paper's instructions and train these models under the same settings.

**Generalizations under different datasets**. In these experiments, we train the compared models on each of the

| Method | DF | | | F2F | | | FS | | | NT | | | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | |
| Xception [41] | 0.654 | 0.681 | 0.617 | 0.708 | 0.598 | 0.745 | 0.708 | 0.601 | 0.605 | 0.646 | 0.625 | 0.838 | 0.669 |
| Face X-ray [24] | 0.609 | 0.554 | 0.668 | 0.633 | 0.684 | 0.766 | 0.646 | 0.697 | **0.795** | 0.613 | 0.703 | 0.866 | 0.686 |
| F3Net [39] | 0.682 | 0.664 | 0.658 | 0.679 | 0.654 | 0.761 | 0.679 | 0.636 | 0.651 | 0.672 | 0.689 | 0.932 | 0.696 |
| RFM [47] | 0.758 | 0.723 | 0.717 | 0.736 | 0.663 | 0.732 | 0.714 | 0.591 | 0.714 | 0.726 | 0.600 | 0.846 | 0.710 |
| SRM [30] | 0.679 | 0.650 | 0.720 | 0.687 | 0.693 | 0.775 | 0.671 | 0.643 | 0.771 | 0.656 | 0.651 | **0.936** | 0.711 |
| Ours | **0.772** | **0.730** | **0.742** | **0.787** | **0.781** | **0.786** | **0.742** | **0.800** | 0.695 | **0.741** | **0.759** | 0.889 | **0.769** |

Table 1. Generalizability comparisons with state-of-the-art methods in the term of AUC. The best results are in bold. The first row denotes the training data, and the second row shows the corresponding test dataset. Our method performs favorably among the models compared.

| Training set | Method | Test set | | | |
|---|---|---|---|---|---|
| | | LQ | | HQ | |
| | | DF | FS | DF | FS |
| NT | Xception [41] | 0.587 | 0.517 | 0.770 | 0.718 |
| | Face X-ray [24] | 0.571 | 0.510 | 0.585 | 0.779 |
| | F3Net [39] | 0.583 | 0.519 | 0.805 | 0.612 |
| | RFM [47] | 0.558 | 0.516 | 0.798 | 0.639 |
| | SRM [30] | 0.555 | 0.529 | 0.838 | **0.795** |
| | Ours | **0.628** | **0.568** | **0.846** | 0.721 |

Table 2. Generalizability comparisons across different compression levels in the term of AUC. Our method achieves comparable performance against existing methods.

four methods in FF++ [41], and evaluate it on the benchmark datasets including CelebDF [28], DFDC [12], and DF1.0 [20]. This setting is rather challenging because both pristines and forgeries in the test dataset are unseen in the training dataset.

We compare different methods by using the Area Under Curve (AUC) metric and present the results in Table 1. As seen, our method outperforms other models in most cases and achieves the overall best performance. This clearly demonstrates the advantage of the proposed adversarial augmentation and self-supervised tasks. SRM [30] and F3Net [39] both rely on high-frequency components of a image to distinguish forgeries from pristine. Our experiment suggests that they attain worse generalization performance than our approach. This may be because the high-frequency cue identified effective on the FF++ dataset [41] may not necessarily generalize to other datasets that adopt different post-processing steps. RFM [47] encourages the use of multiple facial regions for forgery detection and thus leads to improved generalization performance. However, their method still cannot avoid some data source biases, such as all the whole facial parts in a training sample are from the same source. This limitation may explain why their performance is inferior to ours. Face X-ray [24] uses the blended artifacts in the forgery for generalization. Compared to our method, Their generalizability will degrade when these artifacts share different patterns in the training and test dataset. As a baseline, Xception [41] does not incorporate any augmentation or feature engineering. Its performance drops drastically in unseen forgeries.

| Training set | Method | Test set | |
|---|---|---|---|
| | | F2F | FS |
| F2F | LAE [14] | 0.909 | 0.632 |
| | ClassNSeg [34] | 0.928 | 0.541 |
| | Forensic-Trans [9] | 0.945 | 0.726 |
| | Ours | **0.960** | **0.848** |

Table 3. Comparisons with models adopt multi-task learning in the term of ACC. Our model performs favorably against these arts.

| Method | FF++ | CelebDF |
|---|---|---|
| Two-stream [59] | 0.701 | 0.538 |
| Meso4 [1] | 0.847 | 0.548 |
| MesoInception4 [1] | 0.830 | 0.536 |
| FWA [27] | 0.801 | 0.569 |
| DSP-FWA [27] | 0.930 | 0.646 |
| Xception [41] | 0.997 | 0.653 |
| VA-MLP [33] | 0.664 | 0.550 |
| Headpose [53] | 0.473 | 0.546 |
| Capsule [35] | 0.966 | 0.575 |
| SMIL [25] | 0.968 | 0.563 |
| Two-branch [32] | 0.932 | 0.734 |
| SPSL [29] | 0.969 | 0.724 |
| MADD [57] | **0.998** | 0.674 |
| Ours | 0.984 | **0.797** |

Table 4. Extensive evaluations with other state-of-the-art methods in the term of AUC. The models are trained on FF++ dataset. Our method performs favorbly when tested on FF++, and it outperforms others when tested on CelebDF.

**Generalizations under different compression levels**. Given that real world forgeries may be post-processed by different methods, such as compression. It is crucial that deployed deepfake detectors are not easily subverted by unseen post-processing processes. To evaluate the generalizability of the compared models regarding the varying post-processing methods, we separately train them on NT data, while testing them on DF and FS with different compression levels.

| Method | DF | | | F2F | | | FS | | | NT | | | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | |
| Xception [41] | 0.654 | 0.681 | 0.617 | 0.708 | 0.598 | 0.745 | 0.708 | 0.601 | 0.605 | 0.646 | 0.625 | 0.838 | 0.669 |
| Xception w/ adv | 0.717 | 0.703 | 0.674 | 0.739 | 0.778 | 0.735 | 0.737 | 0.644 | 0.602 | 0.662 | 0.722 | 0.794 | 0.709 |
| Ours w/ ran | 0.763 | 0.663 | 0.690 | 0.763 | 0.745 | 0.696 | 0.738 | 0.700 | 0.650 | 0.705 | 0.666 | 0.810 | 0.716 |
| Ours | **0.772** | **0.730** | **0.742** | **0.787** | **0.781** | **0.786** | **0.742** | **0.800** | **0.695** | **0.741** | **0.759** | **0.889** | **0.769** |

Table 5. Ablation studies regarding the effectiveness of the adversarial training. The metric is AUC. Please see Sec. 4.4 for detailed experiment settings.

| Method | DF | | | F2F | | | FS | | | NT | | | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | |
| Ours w/ ran ops | 0.735 | 0.666 | **0.782** | 0.694 | 0.647 | 0.827 | 0.737 | 0.601 | 0.646 | 0.691 | 0.685 | 0.812 | 0.710 |
| Ours w/ ops [55] | 0.721 | 0.726 | 0.777 | 0.686 | 0.647 | **0.840** | 0.737 | 0.534 | **0.720** | 0.678 | 0.642 | 0.829 | 0.711 |
| Ours w/ aug [24] | 0.722 | 0.692 | 0.712 | 0.722 | 0.710 | 0.739 | 0.719 | 0.726 | 0.640 | 0.714 | 0.669 | 0.841 | 0.717 |
| Ours w/ aug [58] | 0.754 | 0.679 | 0.687 | 0.746 | 0.604 | 0.753 | 0.726 | 0.697 | 0.674 | **0.770** | 0.713 | 0.863 | 0.722 |
| Ours | **0.772** | **0.730** | 0.742 | **0.787** | **0.781** | 0.786 | **0.742** | **0.800** | 0.695 | 0.741 | **0.759** | **0.889** | **0.769** |

Table 6. Ablation studies regarding the effectiveness of the data augmentation strategies. The metric is AUC. Please see Sec. 4.4 for detailed experiment settings.

The evaluated AUC values are presented in Table 2. We can observe that almost all models generalize competitively when trained and tested on data with the same compression levels. However, models that based on imperceptible image patterns [41, 24, 39, 30] suffer from large performance drop when test on unseen LQ data. The results are not surprising since the low-level clues are largely destroyed when the images are highly compressed. The same results are reported for RFM [47]. Because all facial parts in a face sample usually share the same compression level, exploring more facial regions cannot guarantee well generalizability across different compression levels. On the other hand, the proposed method is substantially less affected by the compression levels, outperforming all other methods, as it uses more generalizable forgery configurations instead of just the low-level artifacts. We believe the improvements over Xception [41], which is with a similar network setting, are due to the adopted adversarial self-supervised framework.

### 4.3. State-of-the-art Comparisons

**Comparison to multi-task learning detectors**. Using multi-task learning strategies to boost deepfake detection has been explored in previous works, including LAE [14], ClassNSeg [34], and Forensic-Trans [9]. All these works suggest simultaneously classifying and localizing forgery regions. Unlike their localization task, the self-supervised task involved in our self-supervised framework aims to recognize the forgery configurations selected by the synthesizer network, which not only avoids the tedious annotation work, but also considers more configurations that are prevalent in forgeries.

To ensure fair comparisons, we use the same evaluation settings in our experiments with that from the compared arts. We train our model on the F2F data and test it on F2F and FS. The comparison results are presented in Table 3, where our model outperforms other algorithms in test samples with both seen and unseen deepfake techniques. Note the results are directly cited from the reported statistics in their original papers.

**Comparison to other state-of-the-art detectors**. We further evaluate our method against several state-of-the-art models, including the Two-stream [59], MesoNet [1], Headpose [53], FWA [27], VA-MLP [33], Capsule [35], SMIL [25], Two-branch [32], SPSL [29], MADD [57]. We train our model on the FF++ dataset and test it on FF++ and CelebDF. Results of some methods are directly cited from [29]. As shown in Table 4, our method performs competitively with other models when tested on the FF++ dataset, and it achieves the best performance when tested on CelebDF, which further validates the effectiveness and superior generalizability of our method.

### 4.4. Ablation Study

This section analyzes the effectiveness of adversarial learning, data augmentation, and self-supervised tasks. We provide more analyses in our supplementary material.

**Adversarial Learning**. To validate whether the adversarial training strategy can improve the generalizability, we conduct an ablation study by comparing our methods with the following variant. (1) Xception [41]: the baseline approach without using both adversarial learning and self-supervised learning (2) Xception w/ adv: we add adversarial augmentation (but not self-supervised learning) to the Xception baseline. (3) Ours w/ ran: we replace adversarial augmentation with random augmentation. That is, instead of relying on the forgery synthesizer to generate configurations, we randomly select configurations to perform augmentation. This

| | | | DF | | | F2F | | | FS | | | NT | | | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{L}_R$ | $\mathcal{L}_T$ | $\mathcal{L}_A$ | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | DFDC | CelebDF | DF1.0 | |
| ✓ | ✓ | - | 0.770 | 0.686 | 0.687 | 0.768 | 0.714 | 0.779 | 0.722 | 0.709 | 0.653 | 0.735 | 0.720 | 0.856 | 0.733 |
| ✓ | - | ✓ | 0.763 | 0.716 | 0.709 | 0.760 | 0.734 | **0.800** | **0.776** | 0.636 | **0.707** | 0.724 | 0.683 | 0.835 | 0.737 |
| - | ✓ | ✓ | 0.722 | 0.685 | 0.656 | 0.765 | 0.733 | 0.771 | 0.713 | 0.698 | 0.659 | 0.735 | 0.709 | 0.838 | 0.724 |
| - | - | - | 0.717 | 0.703 | 0.674 | 0.739 | 0.778 | 0.735 | 0.737 | 0.644 | 0.602 | 0.662 | 0.722 | 0.794 | 0.709 |
| ✓ | ✓ | ✓ | **0.772** | **0.730** | **0.742** | **0.787** | **0.781** | 0.786 | 0.742 | **0.800** | 0.695 | **0.741** | **0.759** | **0.889** | **0.769** |

Table 7. Effectiveness of the proposed self-supervised auxiliary tasks. The metric is AUC. We disable the self-supervised task by assigning a zero weight to the corresponding loss.

variant is used to test if the improvement brought by the adversarial augmentation is from the "adversarial training" or the "augmentation" or both.

The experimental comparison is shown in Table 5. We can see that using adversarial augmentation alone (Xception w/ adv) can lead to significant (around 4%) improvement over the baseline approach (Xception). This shows the effectiveness of our adversarial augmentation. We also observe that if replacing the adversarial augmentation with random augmentation, the performance of our method leads to a significant drop. This suggests that adversarial training is essential for our system.

**Data augmentations**. We first replace our augmentation step with general augmentation strategies in [10, 55], which includes 16 image operations, such as rotation and cutout [13], and 10 magnitudes. Two augmentation forms are compared. The first one imposes random image operations on the training dataset (*i.e*. Ours w/ ran ops); the second uses a same synthesizer network to select different operations and magnitudes for different samples, which is similar to the setting in [55] (*i.e*. Ours w/ ops [55]). Note the goals of the detector are to estimate the types of image operation and their magnitudes in these two settings. We further compare our augmentation strategy with that from [24] and [58] (*i.e*. Ours w/ aug [24] and [58]). Both these two methods suggest synthesizing new forgeries to augment the training data, where the forgery regions are fixed in the inner faces, and alpha blending is adopted for all samples in [24] while Poisson blending is mainly used in [58].

Evaluation results are shown in Table 6. We observe that strategies with commonly used image operations perform less effectively than our method, and the improvements gained from the adversarial training are also subtle compared to ours. The main reason is that the commonly used image operations are used for general tasks. They may be ineffective in our task. In contrast, our augmentation method is specially designed for deepfake detection, which is more related to the forgery synthesizing process. Thus, it is not surprising that our augmentation strategy outperforms general image operations. Meanwhile, different from the empirically designed augmentations [24, 58], our augmentation is with multiple forms. With the help of adversarial training, more diverse samples can be created to improve the generalization. These advantages explain why our augmentation strategy performs better than [24, 58].

**Self-supervised tasks**. We seamlessly integrate three auxiliary self-supervised tasks to boost the generalizability of our model. To evaluate the effectiveness of these tasks, we conduct ablation studies by assigning zero weight to the loss corresponds to each of these tasks (*i.e*. $\mathcal{L}_R$, $\mathcal{L}_T$, and $\mathcal{L}_A$ correspond to the forgery region estimation mask, blending type estimation task, and the blending ratio estimation task, respectively). All models are trained on the four methods of FF++ and evaluated on benchmark datasets using the same settings. Results are shown in Table 7. We can observe that each auxiliary task plays a vital role in our framework, where our method performs the best when all three self-supervised tasks are incorporated. On the other hand, excluding any of these tasks will decrease the overall performance, and using only the main classification loss lead to significantly worse performance than others. This validates the effectiveness of the proposed self-supervised tasks.

## 5. Conclusions and Discussions

In this paper, we propose a deepfake detection method that can generalize well in unseen scenarios. Our design is based on the intuition that a generalizable deepfake detector should be sensitive to different types of forgeries. We thus leverage a synthesizer and adversarial training framework to dynamically generate forgeries. Training to identify those generated forgeries, the network can learn more robust feature representation and lead to a more generalizable deepfake detector. Through extensive experiments, we demonstrate the effectiveness of the proposed method.

The major limitation of the current method is that the augmentation type in the synthesizer is still limited. One promising direction is to use GAN or other generative models to directly generate forgery images and create self-supervised auxiliary tasks. For example, the generated forgeries can be controlled by some latent variable and the auxiliary task is to predict those latent variables.

**Ethic Statement.** All face images used in this paper are adopted from existing works and are properly cited. There is no violation of personal privacy while conducting experiments in this work.

## References


[1] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Mesonet: a compact facial video forgery detection network. In *WIFS*, 2018.

[2] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. In *CVPR Workshops*, 2019.

[3] Irene Amerini, Leonardo Galteri, Roberto Caldelli, and Alberto Del Bimbo. Deepfake video detection through optical flow based cnn. In *ICCV Workshops*, 2019.

[4] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. In *ICLR*, 2017.

[5] Vishal Asnani, Xi Yin, Tal Hassner, and Xiaoming Liu. Reverse engineering of generative models: Inferring model hyperparameters from generated images. *arXiv preprint arXiv:2106.07873*, 2021.

[6] Chaoqi Chen, Jiongcheng Li, Xiaoguang Han, Xiaoqing Liu, and Yizhou Yu. Compound domain generalization via meta-knowledge encoding. In *CVPR*, 2022.

[7] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised gans via auxiliary rotation loss. In *CVPR*, 2019.

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[9] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*, 2018.

[10] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *CVPR*, 2019.

[11] DeepFakes. www.github.com/deepfakes/faceswap Accessed 2021-04-24.

[12] Deepfake detection challenge. https://www.kaggle.com/c/deepfake-detection-challenge Accessed 2021-04-24.

[13] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[14] Mengnan Du, Shiva Pentyala, Yuening Li, and Xia Hu. Towards generalizable deepfake detection with locality-aware autoencoder. In *CIKM*, 2020.

[15] FaceSwap. www.github.com/MarekKowalski/FaceSwap Accessed 2021-04-24.

[16] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *ISBI*, 2018.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[18] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R Venkatesh Babu. Deligan: Generative adversarial networks for diverse and limited data. In *CVPR*, 2017.

[19] Alexandros Haliassos, Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. Lips don't lie: A generalisable and robust approach to face forgery detection. In *CVPR*, 2021.

[20] Liming Jiang, Ren Li, Wayne Wu, Chen Qian, and Chen Change Loy. Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection. In *CVPR*, 2020.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[23] Jiaming Li, Hongtao Xie, Jiahong Li, Zhongyuan Wang, and Yongdong Zhang. Frequency-aware discriminative feature learning supervised by single-center loss for face forgery detection. In *CVPR*, 2021.

[24] Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen, and Baining Guo. Face x-ray for more general face forgery detection. In *CVPR*, 2020.

[25] Xiaodan Li, Yining Lang, Yuefeng Chen, Xiaofeng Mao, Yuan He, Shuhui Wang, Hui Xue, and Quan Lu. Sharp multiple instance learning for deepfake video detection. In *ACMMM*, 2020.

[26] Yuezun Li, Ming-Ching Chang, and Siwei Lyu. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In *WIFS*, 2018.

[27] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts. In *CVPR Workshops*, 2018.

[28] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-df: A new dataset for deepfake forensics. In *CVPR*, 2020.

[29] Honggu Liu, Xiaodan Li, Wenbo Zhou, Yuefeng Chen, Yuan He, Hui Xue, Weiming Zhang, and Nenghai Yu. Spatial-phase shallow learning: rethinking face forgery detection in frequency domain. In *CVPR*, 2021.

[30] Yuchen Luo, Yong Zhang, Junchi Yan, and Wei Liu. Generalizing face forgery detection with high-frequency features. In *CVPR*, 2021.

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[32] Iacopo Masi, Aditya Killekar, Royston Marian Mascarenhas, Shenoy Pratik Gurudatt, and Wael AbdAlmageed. Two-branch recurrent network for isolating deepfakes in videos. In *ECCV*, 2020.

[33] Falko Matern, Christian Riess, and Marc Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *WACV Workshops*, 2019.

[34] Huy H Nguyen, Fuming Fang, Junichi Yamagishi, and Isao Echizen. Multi-task learning for detecting and segmenting manipulated facial images and videos. *arXiv preprint arXiv:1906.06876*, 2019.

[35] Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos. In *ICASSP*, 2019.

[36] Yuval Nirkin, Lior Wolf, Yosi Keller, and Tal Hassner. Deepfake detection based on discrepancies between faces and their context. *IEEE TPAMI*, 2021.

[37] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[38] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *SIGGRAPH*. 2003.

[39] Yuyang Qian, Guojun Yin, Lu Sheng, Zixuan Chen, and Jing Shao. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *ECCV*, 2020.

[40] Weize Quan, Ruisong Zhang, Yong Zhang, Zhifeng Li, Jue Wang, and Dong-Ming Yan. Image inpainting with local and global refinement. *IEEE TIP*, 2022.

[41] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *ICCV*, 2019.

[42] Christos Sagonas, Epameinondas Antonakos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: database and results. *IVC*, 47:3–18, 2016.

[43] Zekun Sun, Yujie Han, Zeyu Hua, Na Ruan, and Weijia Jia. Improving the efficiency and robustness of deepfakes detection through precise geometric features. In *CVPR*, 2021.

[44] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *TOG*, 38(4):1–12, 2019.

[45] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016.

[46] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. In *Nips*, 2017.

[47] Chengrui Wang and Weihong Deng. Representative forgery mining for fake face detection. In *CVPR*, 2021.

[48] Cong Wang, Fan Tang, Yong Zhang, Weiming Dong, and Tieru Wu. Towards harmonized regional style transfer and manipulation for facial images. *arXiv preprint arXiv:2104.14109*, 2021.

[49] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE SPL*, 25(7):926–930, 2018.

[50] Shujun Wang, Lequan Yu, Caizi Li, Chi-Wing Fu, and Pheng-Ann Heng. Learning from extrinsic and intrinsic supervisions for domain generalization. In *ECCV*, 2020.

[51] Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, and Qifeng Chen. High-fidelity gan inversion for image attribute editing. *arXiv preprint arXiv:2109.06590*, 2021.

[52] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *ML*, 8(3):229–256, 1992.

[53] Xin Yang, Yuezun Li, and Siwei Lyu. Exposing deep fakes using inconsistent head poses. In *ICASSP*, 2019.

[54] Fei Yin, Yong Zhang, Xiaodong Cun, Mingdeng Cao, Yanbo Fan, Xuan Wang, Qingyan Bai, Baoyuan Wu, Jue Wang, and Yujiu Yang. Styleheat: One-shot high-resolution editable talking face generation via pretrained stylegan. *arXiv preprint arXiv:2203.04036*, 2022.

[55] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial autoaugment. In *ICLR*, 2020.

[56] Yong Zhang, Le Li, Zhilei Liu, Baoyuan Wu, Yanbo Fan, and Zhifeng Li. Controllable descendant face synthesis. *arXiv preprint arXiv:2002.11376*, 2020.

[57] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, and Nenghai Yu. Multi-attentional deepfake detection. In *CVPR*, 2021.

[58] Tianchen Zhao, Xiang Xu, Mingze Xu, Hui Ding, Yuanjun Xiong, and Wei Xia. Learning self-consistency for deepfake detection. In *ICCV*, 2021.

[59] Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S. Davis. Two-stream neural networks for tampered face detection. In *CVPR Workshop*, 2017.

[60] Xiangyu Zhu, Hao Wang, Hongyan Fei, Zhen Lei, and Stan Z Li. Face forgery detection by 3d decomposition. In *CVPR*, 2021.