Detecting Deepfake by Creating Spatio-Temporal Regularity Disruption

Jiazhi Guan, Hang Zhou, Mingming Gong, Errui Ding, Jingdong Wang, Youjian Zhao

Abstract—Despite encouraging progress in deepfake detection, generalization to unseen forgery types remains a significant challenge due to the limited forgery clues explored during training. In contrast, we notice a common phenomenon in deepfake: fake video creation inevitably disrupts the statistical regularity in original videos. Inspired by this observation, we propose to boost the generalization of deepfake detection by distinguishing the “regularity disruption” that does not appear in real videos. Specifically, by carefully examining the spatial and temporal properties, we propose to disrupt a real video through a Pseudo-fake Generator and create a wide range of pseudo-fake videos for training. Such practice allows us to achieve deepfake detection without using fake videos and improves the generalization ability in a simple and efficient manner. To jointly capture the spatial and temporal disruptions, we propose a Spatio-Temporal Enhancement block to learn the regularity disruption across space and time on our self-created videos. Through comprehensive experiments, our method exhibits excellent performance on several datasets.

Index Terms—Deepfake Detection, Digital Forensics

I. INTRODUCTION

In recent years, our social platforms have been flooded with high-quality manipulated or forged portrait videos. These videos are created by advanced generators [1]–[7], known as deepfakes, and often target at celebrities or political figures with misleading or seditious intentions, which can lead to potential misperception of the facts and even social unrest. Thus, the detection of deepfakes has become an important topic in the computer vision and related research communities.

With recent efforts devoted to identifying the differences between the real and fake videos [8]–[17], plausible performance has been achieved on popular datasets [18], [19]. However, the generalizability of deepfake detection methods remains one of the biggest concerns in the community. To address this challenging problem, [20] propose a dynamic masking strategy to avoid overfitting on obvious artifacts. With similar intuitions, [21], [22] propose to boost the generalization by adversarial training. Moreover, [23], [24] consider the generalizable temporal coherence, and [11], [25], [26] try to discover more subtle clues in the frequency domain. While their designs achieved better results, these methods leverage only existing datasets generated by limited forgery techniques. As a result, they inevitably tend to recognize only a subset of deepfake’s “fingerprint”. Different from the dataset-dependent methods, [27], [28] propose to imitate the deepfake pipeline and expect their models to learn the discriminatory factors depending on face swapping or warping traces. However, their models still focus on limited artifacts, thus leading to sub-optimal generalization.

One shared intuition among most of previous methods is that they pursue to cover the forgery clues caused by producing deepfakes, such as generative artifacts and evident post-processing traces. In contrast to previous methods, we do not focus on visible and explicit deepfake fingerprints, but rather on the underlying statistical properties of real and fake videos, which we refer to as “regularities”. A few important observations are firstly made: 1) The creation of fake videos inevitably requires modifying and re-assembling the facial parts of the real videos, regardless of the deepfake technique. 2) Spatial modifications would break the homogeneity of images’ statistical properties [29], which are usually introduced by the imaging [30], [31] and image/video compression [32], [33] pipelines. We show a specific noise analysis example in Fig. 1 (a), and (b). 3) The original temporal continuity of the real videos is destroyed when re-assembling separately generated deepfake frames. As shown in Fig. 1 (c), we horizontally stitch the vertical slices of all frames in a video in the same column. The real video shows smoother fluctuation than the fake one. We uniformly name the observed destruction of homogeneity in videos “regularity disruption”.

Guided by the observations above, we anchor our solution on creating pseudo-fake (p-fake) data to imitate both spatial and temporal regularity disruption. The model trained to distinguish real from our self-generated p-fake data should naturally be suitable for the task of deepfake detection once enough regularity disruption distributions can be covered. To this end, we propose a plug-and-play module named Pseudo-fake Generator (P-fake Generator). Spatially, we introduce the regularity disruption through our elaborately designed blend-after-edit pipeline, in which various kinds of edits are considered at photometric perspective, geometric perspective, and frequency domain. Temporally, frames are processed individually with random parameters to break the temporal regularity. Fig. 1 (a)(b)(c) show an example where our p-fake reproduces the irregularities in a very similar manner as the fake results. In addition, as shown in the T-SNE [34] visualization in Fig. 1(d), the generated p-fake covers a broad range of irregularity features.

Based on our P-fake generator, we present a specially devised Spatio-Temporal Enhancement (STE) block to jointly consider spatio-temporal regularity disruption. In STE, channel-wise temporal convolution is first applied to achieve adjacent interactions. Then, by sequentially plugging self-attention spatial re-weighting, the STE block yields spatio-temporally enhanced encodings.

The contributions of this paper are summarized as follows: 1) For the first time, we propose to detect deepfake...
II. RELATED WORKS

Media forensics has been extensively studied for a long time, many efforts are devoted to detect specific types of manipulation, such as splicing, copy-move, and object removal. Although promising results are delivered, these methods are hard to deal with the emerging deepfakes since deepfakes are often generated by advanced GAN-based methods without explicitly copying, pasting, removal, etc.

Deepfake detection is turning out to be a very hot research topic, as high-quality deepfakes are seriously impairing the dissemination of real information in the Internet age. Early works are devoted to identifying deepfake by its obvious artifacts. Li et al. notice the abnormal eye blink signal that is not well presented in early deepfakes. Yang et al. consider the inconsistency of head pose across the central and whole face. Despite promising performances at the time, it is difficult today to rely on these apparent forgery artifacts to counter the increasingly sophisticated deepfake techniques.

As the deepfakes become photo-realistic, recent efforts are devoted to exploring more subtle forgery clues. Dang et al. propose to detect deepfakes with an attention mechanism, which highlights the forgery regions to support classification. Zhao et al. develop a multi-attention framework with the similar idea to enrich the shallow feature encoding. Moreover, without a complex model, Das et al. take a simple but efficient way to augment the training data by randomly masking parts of facial features. Combining with the state-of-the-art (SOTA) CNN backbone, they won the first prize in the most famous deepfake detection challenge. A related idea is studied in, which further takes a dynamic masking processing to suppress overfitting on obvious artifacts. In addition to the RGB domain, many works notice that forgery clues are more discriminative in the frequency domain. Qian et al. explore both the local and global frequency representations of deepfakes. Masi et al. devise frequency encoding into their two-stream model using the Laplacian of Gaussian operator. Although frequency

Fig. 1. (a): Differences between the three kinds of data, p-fake keeps the identity and facial movements of the original real one, while fake is generated by forgery techniques with modified identity or facial movements. (b): Noise analysis of three kinds of data, where our p-fake simulates the noise regularity disruption. It can be observed that the facial region displays smoother textures. (c): Temporal visualization of three kinds of data, where the real one shows smoother fluctuation, and our created p-fake demonstrates the similar temporal irregularity as the fake. (d): The T-SNE visualization of features extracted by our model trained using only real and p-fake data, each dot corresponds to the feature of one test video.
features show remarkable promotion, it is foreseeable that defects in the frequency domain will be less conspicuous, just like in the RGB domain.

To boost the generalization, another commonly applied method is adversarial training. Wang et al. [21] propose a blurring-based adversarial training mechanism and a GAN-based generator, where the former brings the detector better robustness to visual compression and latter acts as a surrogate deepfake model in training. Also with adversarial training, a recent work of Chen et al. [22] additionally restrict the detector to recognize the forgery type as well as the authenticity. These efforts introduce a trainable surrogate deepfake model in the training pipeline to enhance the diversity of deepfakes, which may be able to create deepfakes with fewer visual artifacts, but still cannot jump out of the arms race between detectors and forgeries, since untapped generators always produce different “fingerprint”.

When video data are available, temporal features are also considered in recent works [15, 16, 23, 53–55]. Combined with a multiple-instance learning framework, Li et al. [56] investigate the temporal features using basic 1D-convolutions. In [53], PPG signal [57], which measures the minuscule periodic changes of skin color due to blood pumping, is adopted to identify deepfakes. Halissos et al. [15] propose to utilize the high-level semantic irregularities in mouth movements for a more generalizable deepfake detection. More generally, the temporal inconsistency of deepfakes can be reflected in more low-level features. In a data-driven way, Ganiyusufoglu et al. [54] adopt the 3D convolutional neural networks to achieve deepfake detection. Based on the 3D convolution, Lu et al. [55] propose a 3D-attention mechanism to further improve the performance. To avoid spatial overfitting, Zheng et al. [23] propose a fully temporal convolutional network to enhance the generalization capability.

The closest works to us [27, 28, 58] imitate the deepfake pipeline and try to identify deepfakes depending on face swapping or warping traces. Though we also create negative samples for training using only real data, the most essential differences are twofold: 1) they propose a data generation pipeline to create deepfake artifacts, which are not substantially different from using existing datasets, while we are not interested in creating artifacts, but to break the regularities of real videos. 2) we anchor our solution to the fundamental regularity disruption in both spatial and temporal spaces, which is more generalizable to various kinds of deepfakes. We notice the concurrent work of [59] also proposes a similar self-blending pipeline to create more hardly recognizable forgery traces for better generalization. Different from [59], we draw inspiration from a spatio-temporal regularity perspective that does not capitalise on designated kind of forgery trace but on the overall inconsistency (which is intuitively validated in Fig. 6).

III. PROPOSED METHOD

In this section, we first introduce our proposed Pseudo-fake Generator, then elaborate on the Spatio-Temporal Enhancement block, and finally explain how to train the deepfake detector with the proposed Pseudo-fake Generator.

A. Pseudo-fake Generator

As mentioned in our motivation, we aim to create p-fake with the fundamental regularity disruption, thus allowing our model to learn more generalized discriminatory features. To this end, we propose the P-fake Generator to create p-fake using only the real video. We illustrate the main processing of the P-fake Generator in Fig. 2(a) with a overview description in Algorithm 1 and depict more details about the three modules in Fig. 2(b), (c), (d), respectively. Given a real video, our P-fake Generator handles it frame by frame, where the Mask Generator is responsible for editing frames, and the Blender will blend parts of these disruptions into the real frame with the mask provided by the Mask Generator to create p-fake. In addition, there is a Random Parameter Generator (RPG) that plays an important role in the whole process by controlling the diversity of disruption generation, injection regions, and blending methods over the three modules. For all our notations in follows, we uniformly use subscripts for indices and superscripts to distinguish variables.

1) Image Editor: Given a real frame $I_{\text{real}} \in \mathbb{R}^{H \times W \times 3}$, we consider the possible editing methods from three aspects.

Photometric Perspective. We apply several editing methods: ISO noise, sharpen, downsampling, and color jitter, to modify the original statistic properties of the real frame. Here we present one of them in detail. For color jitter, it is used to alter the original brightness, contrast, and saturation of the real frame. We first define a mapping table $T = [0, 1, \ldots, 255] \in \mathbb{R}^{256}$ and three random parameters $\theta^b, \theta^c, \theta^s$. The brightness adjustment is implemented as:

$$T^b_k = \min(\max([T_k \cdot \theta^b], 0), 255), \quad 0 \leq k < 256,$$

$$I_{i,j,c}^{\text{edited}} = T^b_{i,j,c} \cdot I_{i,j,c}^{\text{real}}, \quad 0 \leq i < H, \quad 0 \leq j < W, \quad 0 \leq c < 3.$$

In a similar manner, contrast adjustment can be learned by:

$$\mu = \frac{1}{H \times W} \sum \text{RGB2GRAY}(I_{\text{real}}),$$

$$T^c_k = \min(\max([T_k \cdot \theta^c + \mu \cdot (1 - \theta^c)], 0), 255).$$

Algorithm 1: Pseudo-fake Generator.

**Input:** real video clip $v_{\text{real}} \in \mathbb{R}^{L \times H \times W \times C}$, facial landmarks $f \in \mathbb{R}^{L \times 68 \times 2}$.

**Output:** pseudo-fake video clip $v_{\text{p-fake}} \in \mathbb{R}^{L \times H \times W \times C}$.

1. for $t \leftarrow 1$ to $L$ do
2. \hspace{1em} $I_{\text{real}} \in \mathbb{R}^{H \times W \times C} \leftarrow v_{\text{real}}^t$;
3. \hspace{1em} $f \in \mathbb{R}^{68 \times 2} \leftarrow f^t$;
4. \hspace{3em} // generate random parameters by RPG for each frame
5. \hspace{3em} $\Theta^1, \Theta^M, \Theta^B \leftarrow \text{RPG}$
6. \hspace{3em} // edit and blend
7. \hspace{3em} $I_{\text{edited}} \leftarrow \text{Image Editor}(I_{\text{real}}; \Theta^1)$;
8. \hspace{3em} $I_{\text{mask}} \leftarrow \text{Mask Generator}(f; \Theta^M)$;
9. \hspace{3em} $I_{\text{p-fake}} \leftarrow \text{Blender}(I_{\text{real}}, I_{\text{edited}}, I_{\text{mask}}; \Theta^B)$;
10. end

11. $v_{\text{p-fake}} \leftarrow I_{\text{p-fake}}$.
where $0 \leq k < 256$, 

$$I_{i,j,c}^{\text{edited}} = T^k_{i,j,c} I_{i,j,c}^{\text{real}}, \quad 0 \leq i < H, \ 0 \leq j < W, \ 0 \leq c < 3. \quad (5)$$

Furthermore, saturation adjustment is implemented at each pixel by:

$$I_{i,j,c}^{\text{edited}} = \min(\max(I_{i,j,c}^{\text{real}} \cdot \theta^s + \text{RGB2GRAY}(I_{i,j}^{\text{real}}) \cdot (1 - \theta^o)), 0), 255). \quad (6)$$

Note RGB2GRAY in Eq. (3), (6) denotes the transformation from RGB color space to gray scale, which is implemented by weighted average from RGB channels.

**Geometric Perspective.** We also introduce different ways to disrupt the original regularity including elastic transform, dense warp and triangular stretch. We elaborate on elastic transform here and illustrate some examples for better understanding in Fig [3]

Specifically, the elastic transform is implemented by per-pixel displacement to disrupt the original spatial relations. We first create two grid-point matrices as:

$$G^x = \begin{bmatrix} [0, 1, \ldots, W - 1]^T, \ldots \end{bmatrix}_{[0, 1, \ldots, H - 1]^{\text{repeat} \ H}},$$

$$G^y = \begin{bmatrix} [0, 1, \ldots, H - 1], \ldots \end{bmatrix}_{[0, 1, \ldots, W - 1]^{\text{repeat} \ W}}. \quad (7, 8)$$

Then, by random sampling and Gaussian blur, we have the disruption matrix $\Delta' = \text{GaussianBlur}(\Delta, \theta^s) \cdot \theta^o$, where $\Delta \in \mathbb{R}^{H \times W}$ is sampled from the uniform distribution $U(-1, 1)$, and $\theta^s, \theta^o$ are two parameters given by the RPG. After that, the disruption matrix is added to the grid-point matrix, we have $G^{x'} = \max(\lfloor G^x + \Delta' \rfloor, 0)$. Meanwhile, $G^{y'}$ is calculated in the same way. As a result, the edited frame is given by:

$$I_{i,j,c}^{\text{edited}} = I_{i,j,c}^{\text{real}} G^x_{i,j,c} G^y_{i,j,c}, \quad (9)$$

where $0 \leq i < H, \ 0 \leq j < W, \ 0 \leq c < 3.$

![Fig. 2. Overview of the P-fake Generator. We illustrate the main processing in (a). The Image Editor is responsible for editing frames, and the Blender will blend parts of these disruptions into the real frame with the mask provided by the Mask Generator to create p-fake. Frames in one video are processed individually with random parameters given by the RPG to break the temporal regularity. More details of the three modules are illustrated in (b), (c), and (d).](image)

![Fig. 3. Examples of geometric editing results. (a) and (d) are two original images, (b), (c), and (e) present three kinds of editing.](image)
Frequency Domain. Recent studies \cite{11, 25, 26, 51, 60} have found that inconsistencies hidden in the frequency domain can be used to reliably distinguish deepfakes. Discrete Cosine Transform (DCT) \cite{61}, which is widely used in image processing, such as JPEG and H.264, can be more compatible and efficient with the description of compression artifacts that are present in forgery patterns \cite{25, 26, 60}. We thus first and adaptable method to describe the compression artifacts processing, such as JPEG and H.264, can be a more effective Discrete Cosine Transform (DCT) \cite{61}, a commonly used technique in image be utilized to reliably identify deepfakes. Discrete Cosine that concealed irregularities in the frequency domain can recent studies \cite{11, 25, 26, 51, 60} have found that inconsistencies hidden in the frequency domain remain in spatial patterns, which greatly enriches the diversity of masks by joining the landmarks to form a closed polygon. For example, generation of the “whole face” and “mouth region” masks is demonstrated in Fig. 4. Other masks are generated using a similar method, with different polygons formed to cover the desired region. Using generation, we randomly adopt a mask from {whole face, narrow face, face with forehead} with a probability of 0.75, and from {face boundary, mouth region, facial organs} with a probability of 0.25. This stochastic selection is also controlled by the RPG. After that, the mask is deformed using elastic transform as introduced earlier, but such practice serves a different purpose of mask augmentation. Then, the mask is softened with a Gaussian blur with a randomly chosen kernel size \( \theta^k \). We also randomly soften the background or foreground to create inconsistent overall smoothness after blending. After that, we stitch the real and edited frames in the certain mask region using randomly chosen blending method (see Fig. 2 (b)). Taking alpha blending as an example, the \( p \)-fake is finally obtained as: 

\[
I_{\text{p-fake}} = I_{\text{real}}(1 - \text{mask}) + I_{\text{edited}} \cdot \text{mask},
\]

where \( I_{\text{p-fake}} \) fuses \( I_{\text{real}} \) and \( I_{\text{edited}} \) together to represent regularity disruption.

3) RPG: RPG is the key to creating temporal irregularity. As described above, during the \( p \)-fake generation, RPG provides parameters (\( \theta^s \)) to affect the final result. Before processing each frame, RPG will sample a set of parameters from certain ranges uniformly. In this way, frames in a video are given a distinct set of parameters, allowing different frames to be edited to different degrees or by different combinations of methods, resulting in temporal regularity disruption. To prevent frequent temporal changes, we ensure that the parameters between randomly selected consecutive frames remain the same.

B. Spatio-Temporal Enhancement

Deepfake exhibits irregularities in both spatial and temporal space, so we specially incorporate the spatio-temporal designs in our model. Inspired by the motion modeling designs in \cite{62–64}, we propose the STE block to enhance the spatio-temporal learning of the CNN backbone.

The STE block is plugged between convolutional layers as shown in Fig. 5 (a). We feed the sequential spatial features into the Temporal Conv and Spatial Attention block to achieve spatio-temporal encoding. Given the features \( \mathcal{F} \in \mathbb{R}^{C \times L \times H \times W} \), where \( C \) is the channel dimension, \( L \) is the temporal dimension, and \( H \times W \) are the spatial dimensions, the Temporal Conv block imposes the channel-wise temporal convolution as:

\[
\hat{F}_{c, t, x, y} = \sum_{i \in [-1, 0, 1]} K^c_i \cdot \mathcal{F}_{c, t+i, x, y},
\]

where \( K^c_i \) is the kernel of channel \( c \). We implement this operation using 3D convolutional layer with the kernel size of \( 3 \times 1 \times 1 \), i.e., the features \( \hat{F}_t \) at timestamp \( t \) only interacts with its adjacent frames. In addition, the Spatial Attention block considers the spatial relations and imposes a patch-wise self-attention operation. Given \( \hat{F}_t \in \mathbb{R}^{C \times H \times W} \) at timestamp \( t \), we first split the features at \( H \times W \) dimensions

(1) Face Crop (2) Landmark Detect
(3) Form Polygon (4) Generate Mask

Fig. 4. Mask generation: (1) The face region is cropped from the input frame. (2) We detect 68-point facial landmarks using an off-the-shelf tool. (3)&(4) We select desired points to form a polygon that covers the region to be masked.
into local patches using patch-convolutional operation, where the convolution kernel size and slide window stride are both set to \(7 \times 7\). Meanwhile, we squeeze the patch features at channel dimension with a reduction factor \(r\) of 8. The Patch Squeeze (see Fig. 5(c)) operation can be denoted as:

\[
\{S_1, S_2, ..., S_p\} = \text{PatchSqueeze}(\tilde{F}_t), \quad (12)
\]

where \(p = \lfloor H/7 \rfloor \times \lfloor W/7 \rfloor\) denotes the patch number, \(S_i \in \mathbb{R}^{C/r}\) is the output of the patch-convolution at a specific \(7 \times 7\) region. Next, we feed the local patch features into a multi-head self-attention (“SelfAtt”) block [65] as:

\[
\{S'_1, S'_2, ..., S'_p\} = \text{SelfAtt}(\{S_1, S_2, ..., S_p\}). \quad (13)
\]

Thus, local patches can interact with the whole spatial space through this block. After that, the attention features are excited by a point-convolutioinal operation (kernel size and slide window stride are set to \(1 \times 1\), “PointConv”) to recover the original channel dimension as:

\[
S''_i = \text{PointConv}(S'_i), \quad 0 < i \leq p. \quad (14)
\]

Then we put the patch feature back to its original spatial location (see Fig. 5(c)) and interpolate the result to recover the original size. Finally, we have the Spatio-Temporal Enhanced feature as:

\[
\tilde{F}_t = \tilde{F}_t \cdot \sigma(\text{Interp-Reshape}(\{S''_1, S''_2, ..., S''_p\})), \quad (15)
\]

where \(\sigma\) denotes sigmoid function and Interp-Reshape indicates interpolation after reshaping operation.

In this work, we plug the STE block before the first convolutional layer of every ResBlock in ResNet-34 [66] as our model.

C. Detector Training and Discussion

There are two basic training schemata depending on whether or not the fake data is used. 1) Using only the real data of existing deepfake dataset; 2) Both real and fake data are available. Regardless of the schema, the P-fake Generator will create p-fake on-the-fly to support the training. The label of our generated p-fake is considered as the same as fake data. As a binary classification problem, our model is trained with 0/1 label (0 for real, 1 for fake and p-fake) supervision using binary cross-entropy loss:

\[
\mathcal{L}(\mathcal{W}) = -y_{gt}\log(y_{\text{pred}}) - (1 - y_{gt})\log(1 - y_{\text{pred}}), \quad (16)
\]

where \(\mathcal{W}\) is the trainable parameters in our network, \(y_{gt}\) denotes the ground truth label, and \(y_{\text{pred}}\) is the predicted confidence score.

With the p-fake support in training, our method shows several fabulous properties for deepfake detection: 1) Deep Neural Networks can be trained without requiring labor-intensive efforts. The training of Deepfake detection mostly relies on publicly available deepfake methods which are limited in specific domains. Additionally, annotating in-the-wild fake videos accurately is challenging given the increasing fidelity of deepfake generation techniques. For example, we can hardly recognize fake video footage\(^1\) without title information. 2) The real data is almost inexhaustible. Thus, with our proposed P-fake Generator, in theory, we have an infinite variety of data for training. 3) Alternatively, combined with existing deepfake datasets, our proposed P-fake Generator can work as a plug-and-play module to enhance these datasets and be applied to all the existing learning-based methods.

IV. EXPERIMENTS

In this section, we first briefly introduce the experiment setups, then present abundant experimental results to demonstrate the SOTA performance of our method, and finally validate the effectiveness of our designs through ablations.

Datasets. We conduct our experiments on several recently published datasets. 1) FaceForensics++ (FF++) [18] is the most popular dataset used to evaluate deepfake detection performance. It contains 1000 real videos collected from the Internet, and 4000 fake videos generated by four kinds of deepfake techniques including Deepfakes (DF) [67], FaceSwap

---

\(^1\)https://www.youtube.com/watch?v=iYiOVUbsPcM
Train

FF++

99.38 99.53 99.36 97.29 98.89
99.76 99.68 99.65 99.40
100.00 98.97 99.86 97.63 99.11
98.88 98.13 94.13 88.38

Data Preprocessing. Except for Deepwild (which provides (EER) as our evaluation metrics. [58], we use Accuracy score (ACC), Area Under the Receiver Evaluation Metrics. Following the previous works [9], [28], [58], we use Accuracy score (ACC), Area Under the Receiver Operating Characteristic Curve (AUC), and Equal Error Rate (EER) as our evaluation metrics. Data Preprocessing. Except for Deepwild (which provides face crops rather than videos), we first extract frames from videos and crop the face regions using MTCNN [75]. After that, face crops are resized to 299 \times 299. For the real data used to generate p-fake, we extract the 68-points facial landmark using Dlib [76].

Training Strategy. We train our model using Adam optimizer and weight decay 10^{-7}. If the validation loss does not decline for 20 epochs, we decay the learning rate by the factor of 0.3, so that our model can converge after several decays.

Implementation Details. We construct our model based on the CNN backbone, ResNet-34 [66] and we plug the proposed STE block before the first convolutional layer of each ResBlock. For the input data, our model takes sequences with 32 successive frames. Common data augmentations including horizontal flip, random rotate, and image compression are applied. All the experiments are conducted on the NVIDIA 3090 GPU.

A. Comparing with Previous Methods

As the goal we mentioned previously, to achieve deepfake detection, we need only the real data, and P-fake Generator will provide the p-fake as the negative sample.

In-Dataset Performance on FF++. We first conduct the experiment on FF++ to demonstrate the effectiveness of our main idea. Using only the 720 real videos from the train set, we trained our model on FF++ and evaluated it on four subsets. The results are shown in the Table I. Xception [77] represents the commonly used baseline in the field of deepfake detection, which predicts the fake probability by averaging the outcomes of individual frames. When training and testing are performed on data with the same distribution of forgery cues, good performance is naturally achieved by learning-based methods, e.g., Xception. Note only this baseline is trained using both real and fake videos of the train set. However, other methods in Table I use only a subset of the train set (only the real part) and do not rely on fitting forgery features of existing deepfakes. The better performance of ours demonstrates the effectiveness of our approach. In our later experiments, we find that incorporating fake data from the train set can further improve the performance under this in-dataset evaluation. Nonetheless, we notice the performances evaluated on the raw version of FF++ turn to saturation. We thereupon further compare with recent arts on the compressed c32 version in Table II. When the video is compressed, specific forgery traces are less significant in the spatial dimension, thus Face X-ray [28] and SBI [59] both demonstrate suboptimal results. While our method consider the overall spatio-temporal regularity, thus maintaining better performance even the forgery traces at spatial dimension are less significant. Performances within FF++ validate the feasibility of our approach, while generalizability remains a major problem of existing deepfake detection methods, as the performance is not guaranteed when testing on deepfakes generated by unseen techniques. Consider further the reality of the arms race between detectors and forgers, generalizability is an important criterion to measure the effectiveness of a detection method in the real world.

Cross-Dataset Performance. Considering the more challenging cross-dataset setting, we further evaluate our method on

### Table I

**Performance evaluated on FF++ (raw). †: Re-implemented using the official code by ourself.**

| Method        | Train Set | Test Set (AUC%) |
|---------------|-----------|-----------------|
|               | DF | P2F | FS | NT | Avg |
| Xception [77] | 99.38 | 99.53 | 99.36 | 97.29 | 98.89 |
| Face X-ray [28] | 99.17 | 98.57 | 98.21 | 98.13 | 98.52 |
| PCL+2G [58]   | 100.00 | 98.97 | 99.86 | 97.63 | 99.11 |
| EB4+SBI [59]  | 99.99 | 99.79 | 99.58 | 97.81 | 99.29 |
| Ours          | 99.76 | 99.68 | 99.65 | 99.40 | 99.62 |

### Table II

**Performance evaluated on FF++ (c32, compressed). †: Re-implemented using the official code by ourself.**

| Method        | Train Set | Test Set (AUC%) |
|---------------|-----------|-----------------|
|               | DF | P2F | FS | NT | Avg |
| Face X-ray [28] | 97.53 | 88.99 | 96.42 | 82.80 | 91.44 |
| EB4+SBI [59]  | 98.88 | 98.13 | 94.13 | 88.38 | 94.88 |
DFD, DFDCP, Deepwild, and CDF. Following the setup in the SOTA method [9], we train our model on the c23 version of FF++ and test on other four datasets (using the official split of the test sets as we described in the beginning of this section).

Note that our method, Face X-ray [28], PCL+I2G [58], and SBI [59] use only the 720 real data of FF++ in training. While other learning-based methods are trained using 720 real and 2880 fake videos. The results are shown in Table III where our method presents superior performance on all four datasets. As only the face-blending artifacts are considered in Face X-ray, it shows relatively inferior performance. Comparing with PCL+I2G, which models the face-swapping inconsistency in spatial dimension to identify deepfakes, our method outperforms it on DFDCP with a clear margin. Moreover, SBI is a concurrent work that proposes a similar blending-after-editing pipeline as our p-fake Generator. It achieves similar performance with a stronger backbone (EfficientNet-B4 [83]), while using the same backbone, ResNet-34 (R34) [66], our model shows better performance attributing to the fundamental spatio-temporal regularity disruption spotting. On the most challenging Deepwild, our method surpasses the SOTA method by about 10 percentage points in terms of AUC%. We think this is due to the large diversity of deepfakes in Deepwild, which makes other methods fail to generalize well from seen deepfakes.

B. Ablation Study

Performance Gain of p-fake. To clearly demonstrate the effect of the proposed p-fake, we train our model using different data combinations on FF++. Both the in-dataset and cross-dataset results are shown in Table IV. Although better in-dataset performance is reported using "real+fake", model trained using "real+p-fake" shows significantly better performance on unseen datasets. As the AUC% is improved from 86.36% to 99.15%, from 77.80% to 90.17% on FSh and CDF, respectively. When combining the three types of data together, the performance on FF++ is further improved, we leave more in-dataset results of "real+fake+p-fake" combination in the next paragraph. Another observation is that adding fake data from FF++ to the training set resulted in a significant drop in performance on DFDCP and CDF, while the performance on FF++ and FSh datasets increased. This suggests that including more fake data with different forgery patterns in training (i.e., fake of FF++ train set) may not necessarily benefit discovering of out-of-domain forgeries (i.e., test set of DFDCP and CDF).

"real+fake+p-fake" Combination Evaluation. In-dataset evaluation on FF++ is widely adopted by many previous methods. Here we combine the self-created p-fake with real and fake data of FF++ ("real+fake+p-fake") in training, and test our model on different compressed versions of FF++, respectively. The results are shown in Table V. Note we exclude the raw version here since even a simple baseline such as Xception can achieve saturated performance on the raw data (see Table I), which hardly reveals differences among methods. Our method achieves the best performance on both the compressed versions. Similar results are reported by F3-Net, while they explore the subtle frequency forgery clues with more complex model (∼41.6M parameters). In contrast, we achieve the SOTA performance with about half the parameters (∼23.5M), benefiting from the discriminative irregularity learning aided by the P-fake Generator and spatio-temporal encoding of the STE block.

Does p-fake benefit different models? To examine whether the performance improvement achieved with the use of p-fake is transferable to existing methods, we conduct additional experiments on three types of learning-based models: the commonly used CNN backbone Xception [77], the popular Transformer-based [84] model Swin [80], and a temporal-aware model TSM-R34 [81]. Three models are train on FF++ using data combinations of "real+fake" (denoted w/o p-fake) and "real+p-fake" (denoted w/ p-fake). As the results shown in Table VI we see a clear promotion is obtained for all the models when p-fake is included. We also observed that the Transformer-based [84] model achieves the best performance without aids of p-fake. This may suggest that the patch-based learning could be more effective for the task of deepfake detection, as most of the menacing deepfakes involve partial modifications such as face swaps and lip syncing. This finding is also supported by our later ablation of the patch-level.

| Method          | Train Set | FF++ | DFD | DFD | Deepwild | CDF |
|-----------------|-----------|------|-----|-----|----------|-----|
|                 | AUC%      | EER% | AUC%| EER%| AUC%     | EER%|
| F3-Net [26]     | 86.10     | 26.17| 72.88| 33.38| 67.71    | 40.17|
| MAT [13]        | 87.58     | 21.73| 67.34| 38.31| 70.15    | 36.53|
| GFF [17]        | 85.51     | 25.64| 71.58| 34.77| 66.51    | 41.52|
| LTW [78]        | 88.56     | 20.57| 74.58| 33.81| 67.12    | 39.22|
| LipForensics [15]† | 75.27 | 34.16| 67.17| 39.18| 66.14    | 37.80|
| FTCN-TT [23]    | -         | -    | 74.00| -    | -        | -    |
| Local-relation [79] | -    | -    | -    | -    | 86.90    | -    |
| DCL [9]         | 89.24     | 20.32| 76.53| 32.41| 68.76    | 37.50|
|                 | 91.66     | 16.63| 76.71| 31.97| 71.14    | 36.17|

- Re-implemented using the official code by ourselves.

TABLE III

Cross-dataset evaluation. We train our model using only real videos in FF++ and tested on DFD, DFDCP, Deepwild, and CDF, respectively. We report the performance of AUC%↑ and EER%↓, with the best in bold and the second best underlined.
TABLE IV
PERFORMANCE GAIN OF p-fake evaluated on FF++ (c23), FSh, DFDCP, and CDF. We report the performance of AUC%↑ and EER%↓.

| Train Data      | FF++ AUC% | FF++ EER% | FSh AUC% | FSh EER% | DFDCP AUC% | DFDCP EER% | CDF AUC% | CDF EER% |
|-----------------|-----------|-----------|----------|----------|------------|------------|----------|----------|
| real            | 99.12     | 2.86      | 86.36    | 20.71    | 73.20      | 32.37      | 77.80    | 30.59    |
| real+p-fake     | 94.88     | 13.21     | 99.15    | 3.57     | 85.01      | 23.74      | 90.17    | 17.51    |
| real+fake+p-fake| 99.33     | 1.79      | 99.61    | 2.86     | 83.51      | 24.51      | 85.68    | 21.47    |

TABLE V
IN-DATASET EVALUATION ON FF++ WITH DIFFERENT COMPRESSION QUALITIES. OUR METHOD SHOWS BETTER PERFORMANCE (ACC%↑ AND AUC%↑) ON LOWER-QUALITY VIDEOS, WHERE THE BEST IS SHOWN IN BOLD TEXT AND THE SECOND BEST IS UNDERLINED.

| Method       | c23 ACC% | c23 AUC% | c40 ACC% | c40 AUC% |
|--------------|----------|----------|----------|----------|
| Add-Net [35] | 96.78    | 97.74    | 87.50    | 91.01    |
| Two-Branch   | 96.43    | 97.80    | 86.34    | 86.59    |
| MAT [13]     | 97.60    | 99.29    | 88.69    | 90.40    |
| FDFL [25]    | 96.69    | 99.30    | 89.00    | 92.40    |
| Ours         | 98.29    | 99.33    | 90.47    | 93.71    |

TABLE VI
PERFORMANCE EVALUATED ON FSH AND CDF. TWO NUMBERS IN EACH CELL DENOTE TEST RESULTS OF MODEL TRAINED W/O AND W/ p-fake.

| Model ⎯→      | Xception [77] | Swin [80] | TSM-R34 [81] |
|---------------|----------------|-----------|---------------|
| AUC%↑         | FSh            | 78.50/95.84 | 84.21/92.90 | 81.75/96.08 |
|               | CDF            | 74.90/82.50 | 81.70/84.75 | 75.42/82.38 |

designed STE.

Diversity of p-fake. In the P-fake Generator, we consider editing methods from different aspects, different disruption regions, and different blending methods to cover larger irregularity distribution. Here we validate these designs in FF++ and show the results in Table VII. It shows that diverse heterogeneous features introduced by editing methods at different perspectives result in better performance. In combination with the random mask and blending, our designs consistently improve the final performance.

Benefits of STE. In this part, we conduct ablation study on the effect of our proposed STE block. As we insert the STE block before the ResBlock in ResNet-34 to enhance the spatio-temporal modeling, by removing SET blocks, the model presents only spatial encoding ability. We illustrate the empirical results in Table VIII. With the unified spatio-temporal modeling, our model shows non-trivial improvements, especially on FSh and CDF, the average AUC% is improved from 88.00% to 94.66%. We also compare with two spatio-temporal baselines, TSM-R34 [81] and SlowFast-R50 [82], which are firstly proposed for action recognition. The results indicate that our elaborately devised STE block is more suitable for the task of deepfake detection as more fine-grained patch-level modeling is considered.

Does STE contribute w/o p-fake? We further validate the effectiveness of STE by conducting comparison without aids.

![Fig. 6. (a),(d): The original frames. (b),(e): CAMs of our model trained using "real+p-fake" data. (c),(f): CAMs of the model trained without p-fake](image-url)
of p-fake. We train three models on training set of FF++ and test on FSh and CDF. The results are tabulated in Table IX. Both TSM-R34 [81] and our model are built based on ResNet (R34) [66], but the proposed STE considers more fine-grained patch-level spatio-temporal modeling shows more powerful forgery spotting ability. Comparing with R34, the performance is improved from 72.73% to 82.08% averaged from FSh and CDF.

### C. Activation Visualization

To intuitively demonstrate different patterns learned with p-fake support, we compare the CAM [85] visualizations between models trained with and without p-fake data. As shown in Fig. [6], our model focuses on the overall regularity under the supervision of p-fake. While the model trained without p-fake only pays attention to the most semantically suspicious facial features. Based on the results of our quantitative experiments, we believe that paying global attention to both the background and facial parts can improve the generalization ability of deepfake detectors.

### V. Conclusions

Jointly considering spatio-temporal properties of real videos, we introduce a plug-and-play module, P-fake Generator, to create p-fake videos as negative samples for training. Such practice brings a broad range of possible regularity disruptions and greatly improve the generalizability of the detector. Based on our P-fake Generator, we further present a specially devised STE block to better capture the spatio-temporal irregularity patterns. Extensive experimental results demonstrate the superior performance of our method. We hope that the idea of regularity disruption spotting can be incrementally studied to better deal with the real-world demand of deepfake detection.

### References

[1] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, “Attgan: Facial attribute editing by only changing what you want,” IEEE Transactions on Image Processing, vol. 28, no. 11, pp. 5464–5478, 2019.

[2] H. Kim, P. Garrido, A. Tewari, W. Xu, J. Thies, M. Niessner, P. Pérez, C. Richardt, M. Zollhöfer, and C. Theobalt, “Deep video portraits,” ACM Transactions on Graphics (TOG), vol. 37, no. 4, pp. 1–14, 2018.

[3] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, “Synthesizing obama: learning lip sync from audio,” ACM Transactions on Graphics (ToG), vol. 36, no. 4, pp. 1–13, 2017.

[4] Y. Xu, X. Xu, J. Jiao, K. Li, C. Xu, and S. He, “Multi-view face synthesis via progressive face flow,” IEEE Transactions on Image Processing, vol. 30, pp. 6024–6035, 2021.

[5] M. Cao, H. Huang, H. Wang, X. Wang, L. Shen, S. Wang, L. Bao, Z. Li, and J. Luo, “Unifacegan: A unified framework for temporally consistent facial video editing,” IEEE Transactions on Image Processing, vol. 30, pp. 6107–6116, 2021.

[6] Z. Shao, H. Zha, J. Tang, X. Lu, and L. Ma, “Explicit facial expression transfer via fine-grained representations,” IEEE Transactions on Image Processing, vol. 30, pp. 4610–4621, 2021.

[7] N. Yang, Z. Zheng, M. Zhou, X. Guo, L. Qi, and T. Wang, “A domain-guided noise-optimization-based inversion method for facial image manipulation,” IEEE Transactions on Image Processing, vol. 30, pp. 6198–6211, 2021.

[8] B. Bayar and M. C. Stamm, “A deep learning approach to universal image manipulation detection using a new convolutional layer,” in Proceedings of the 4th ACM workshop on information hiding and multimedia security, 2016, pp. 5–10.

[9] K. Sun, T. Yao, S. Chen, S. Ding, R. Ji et al., “Dual contrastive learning for general face forgery detection,” arXiv preprint arXiv:2112.13522, 2021.

[10] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen, “Distinguishing computer graphics from natural images using convolution neural networks,” in 2017 IEEE Workshop on Information Forensics and Security (WIFS). IEEE, 2017, pp. 1–6.

[11] Y. Luo, Y. Zhang, J. Yan, and W. Liu, “Generalizing face forgery detection with high-frequency features,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16317–16326.

[12] L. Chai, D. Bau, S.-N. Lim, and P. Isola, “What makes fake images detectable? understanding properties that generalize,” in European Conference on Computer Vision. Springer, 2020, pp. 103–120.

[13] H. Zhao, W. Zhou, D. Chen, T. Wei, W. Zhang, and N. Yu, “Multi-attentional deepfake detection,” arXiv preprint arXiv:2103.02406, 2021.

[14] S. Das, S. Seferbekov, A. Datta, M. Islam, M. Amin et al., “Towards solving the deepfake problem: An analysis on improving deepfake detection using dynamic face augmentation,” in Proceedings of the
