Beyond Spatial Frequency: Pixel-wise Temporal Frequency-based Deepfake Video Detection

Supplementary Material

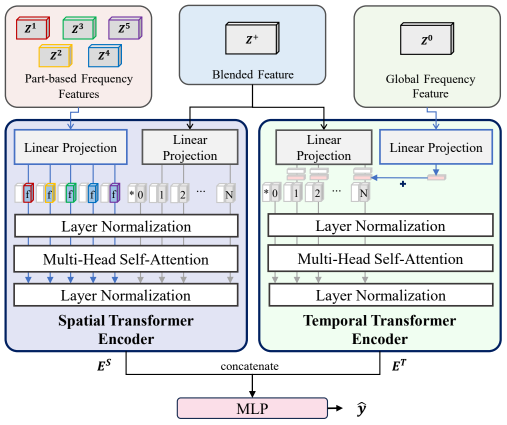


Figure A. **Visualization of Transformer-based Integration architecture.** The transformer-based integration takes the global frequency feature z^0, the part-based frequency features z^p, $p \in \{1,2,3,4,5\}$, and the blended feature Z^+. The transformer-based integration obtains the spatial embedding E^S using spatial transformer encoder and the temporal embedding E^T using temporal transformer encoder and uses these embeddings to make the final prediction $\hat{y}$. For STE and TTE, the frequency features are subjected to average pooling, producing $Pool(z^x) \in R^{B \times C}$, with $x \in \{0,...,5\}$.

A. Detail of Transformer-based Integration

In this section, we will describe in detail the structure of the transformer-based integration in the joint transformer module introduced in the previous section. Our transformer-based integration is mainly composed of a Spatial Transformer Encoder (STE) and a Temporal Transformer Encoder (TTE), and we visualize its architecture in Fig. A.

A.1. Spatial Transformer Encoder

We design the spatial transformer encoder to improve the interaction between the spatial feature from the feature blender and the part-based frequency features. STE outputs the spatial embedding $E^S \in R^{1024}$ by feeding the spatial features of the blended feature $Z^0 \in R^{D_C \times D_T \times D_H \times D_W}$ and each part-based frequency feature Z^p as a token to the Standard Transformer.

We first estimate the spatial features $Z^{sp} \in R^{D_C \times 1 \times D_H \times D_W}$ from blended feature Z^+ by averaging on the temporal axes, and STE generates the spatial embedding E^S by getting Z^{sp} as a token. We apply linear projection W^{sp} to map the sequence of features $z^{sp}_s \in R^{D_C}, s \in \{1,2,...,D_H \times D_W\}$ of Z^{sp} and add a 2D positional encoding pos^{sp}.

$$tokens^{sp}_{+} = [z^{sp}_{class}, W^{sp} z^{sp}_1, ..., W^{sp} z^{sp}_{D_H \times D_W}]^T + pos^{sp}, \tag{1s}$$

where pos^{sp} is 2D sincos positional encoding, Z^{sp}_{class} is extra class embedding, and 'sp' is short for spatial. Using the coordinate values (a_p, b_p) employed to crop each part-based frequency feature, the p^{th} part position encoding value pos^{part}_p is obtained by interpolating neighboring positional encoding values in pos^{sp} and then adding it to a frequency position value pos^{freq}. This frequency position value pos^{freq} serves as a specific indicator of frequency domain features.

$$pos^{part}_p = \text{interpolation}((a_p, b_p), pos^{sp}) + pos^{freq}, p \in \{1,...,5\}. \tag{2s}$$

We apply linear projection W^{freq} to map the sequence of part-based frequency features Z^p and add with part position pos^{part}_p.

$$tokens^{sp}_{freq} = [W^{freq} Z^1 + pos^{part}_1, ..., W^{freq} Z^5 + pos^{part}_5]^T. \tag{3s}$$

We concatenate the tokens $tokens^{sp}_{+}$ and $token^{sp}_{freq}$, which are fed into a transformer to get the spatial transformer embedding E^S.

$$E^S = \mathbf{STE}(tokens^{sp}_{+}, tokens^{sp}_{freq}). \tag{4s}$$

A.2. Temporal Transformer Encoder

Similar to STE, the Temporal Transformer Encoder (TTE) takes the temporal features $Z^{tp} \in R^{D_T \times D_C \times 1 \times 1}$ of the blended feature Z^+ and the global frequency features Z^0 and outputs the temporal embedding $E^T \in R^{1024}$. To make token $tokens_{tp}$, we apply linear projection W^{tp} to $Z^{tp}_t \in R^{D_C}, t \in \{1,2,...,D_T\}$ and add with 1D positional encoding pos^{tp}. Z^{tp}_{class} is extra class embedding and 'tp' is short for temporal.

$$tokens^{tp} = [Z^{tp}_{class}, W^{tp} Z^{tp}_1, ..., W^{tp} Z^{tp}_{D_T}]^T + pos^{tp}.$$

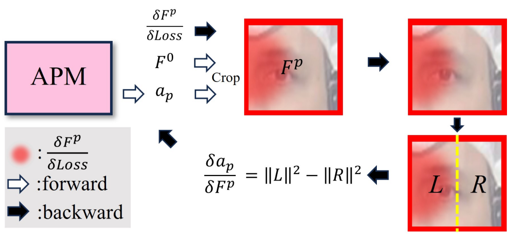
We put $tokens^{tp}$ into a transformer to get the temporal transformer embedding E^T after adding with mapped global frequency feature by linear projection W^{tp}_{freq}.

$$E^T = \mathbf{TTE}(tokens^{tp} + W^{tp}_{freq} Z^0). \tag{5s}$$

To get the final prediction $\hat{y}$, we concatenate $W^b E^S$ and E^T, which are put into the final classifier ϕ^{final}.

$$\hat{y} = \phi^{final}(W^b E^S, E^T), \tag{6s}$$

where $W^b \in R^{1024}$ is a linear projection that aligns distributions.



APM
$\frac{\delta F^p}{\delta Loss}$
F^0
a_p
Crop
F^p
$\frac{\delta a_p}{\delta F^p} = ||L||^2 - ||R||^2$
L
R
$\frac{\delta F^p}{\delta Loss}$
:forward
:backward

Figure B. **Visualization of Updating the Attention Proposal Module.** We visualize the process of updating APM.

B. Detail of Updating Attention Proposal Module.

This section details the operation of the Attention Proposal Module (APM) within the Frequency Feature Extractor (introduced in Section 4.1). The APM automatically identifies and focuses on artifact-rich regions within the input, enabling adaptive localization of deepfake manipulations. Unlike methods relying on predefined regions (e.g., facial landmarks), the APM leverages classification gradients to pinpoint areas most relevant for deepfake detection.

To update the parameters of the APM, the following steps are performed. First, the input frequency spectra are masked by applying $M_p(a_p,b_p)$ according to Eq. 4, ensuring proper gradient computation during backpropagation. The regions corresponding to elements with a value of 1 in M_p are then cropped for part-based frequency extraction. Importantly, this cropping operation does not interfere with gradient propagation, as masked-out regions (with zero values) naturally propagate zero gradients. Next, gradients derived from the binary classification loss are backpropagated through the cropped regions, guided by the APM's outputs a_p and b_p in Eq. 3. Taking the x-axis as an example, the gradients are divided into left L and right R segments ($R,L \in \mathbb{R}^{H\times\frac{W}{2}}$). The squared magnitudes of the gradient values ($||L||^2$ and $||R||^2$) corresponding to each segment are calculated and their difference $||L||^2 - ||R||^2$ is computed and transmitted as the gradient value for a_p. Thus, if the gradient magnitude is larger on the left segment, the APM is encouraged to propose a smaller x value (to the left). Similarly, b_p is updated along the corresponding axis. Through this training procedure, the APM learns to effectively focus on regions exhibiting prominent gradient responses (artifact-rich areas), thereby enhancing its ability to localize and analyze deepfake anomalies. We describe this process in Fig. B.

C. Analysis of Temporal Frequency Extraction

We experimented to check how to extract the temporal frequency for deepfake video detection. Table A and Fig. C are the results of a cross-synthesis experiment using the ResNet-50 [10] classifier trained by global temporal frequency only.

For the experiment using one synthesis method, we evaluated our method by training a model with the training data generated by the other three methods. We experiment with four different synthesis methods (DF, FS, F2F, NT) in FaceForensics++ (FF++) [18].

Filter	Train on remaining three				
	DF	FS	F2F	NT	Avg
None	73.01	53.26	64.40	58.33	62.25
Median	**98.05**	**80.98**	**95.29**	**95.02**	**92.34**
Mean	96.85	78.13	90.88	93.39	89.81

(a) **Filtering Method.**

Feature	Train on remaining three				
	DF	FS	F2F	NT	Avg
Magnitude	**98.05**	**80.98**	**95.29**	95.02	**92.34**
Phase	54.36	50.82	55.25	60.93	55.34
All	95.45	73.22	92.81	**95.96**	89.36

(b) **Comparison of Frequency Features.**

Table A. **Experiment about temporal frequency extraction methods.**

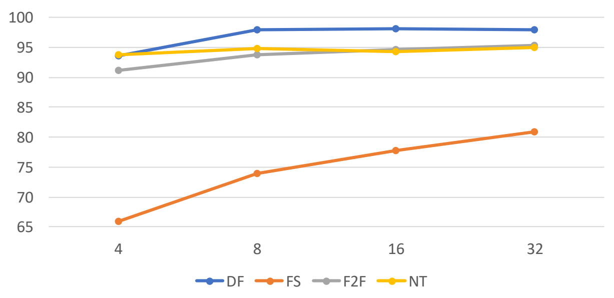


Figure C. **Comparison of frame intervals for temporal-frequency extraction.** The horizontal axis is the frame interval used to extract the temporal frequency and the vertical axis is the video-level AUC(%) performance of the cross-synthesis experiments for each method.

In our default setting, as a pre-processing, we apply the median filter to extract the frequency for 32 frames, using the magnitude of the filtered frequency to detect the deepfake video. Before performing 1D Fourier transform, we obtain a pre-processed frame $\hat{I}$ by removing the dominant components through a median filter as

$$\hat{I} = gray(I - filter(I)), \tag{7s}$$

where $filter(I)$ is an image filtered by a median filter and $gray(I)$ means the gray-scaling function from the colored image I.

We compare the performance of the process with and without a filter and the performance of different types of filters and show the results in Table Aa. The first row (None) is the result without filter, which means the result when the original image I is gray-scaled and Fourier transformed and fed into ResNet-50, the second row (Median) is the result when median filter is applied as filter, and the last row (Mean) is the result when Mean filter is applied in Eq. 7s. Even though the mean filter was also effective

Method	FF++	CDF	FSh
AltFreezing	99.7 → 62.2 (−37.6 %)	89.0 → 56.8 (−36.2 %)	99.0 → 63.2 (−36.2 %)
Ours	99.6 → 56.0 (−43.8 %)	89.7 → 57.8 (−35.6 %)	99.4 → 53.1 (−46.6 %)

Table B. **Performance degrade on shuffled frames.**

Arch.	APM	FSh	DFDC	DFo	Avg.
MLP		85.0	64.0	88.6	79.2
MLP	✓	97.4	69.6	97.6	88.2 (11.4% ↑)

Table C. **Isolated Effectiveness of the APM.** We trained the model on FF++ and evaluated it for FSh, DFDC and DFo.

for our method, the median filter showed the best performance.

Table Ab is the result when using phase and magnitude of temporal frequency. Table Ab presents the performance for different frequency features—Magnitude, Phase, and a combination of both (All). We observe that employing magnitude outperforms employing phase alone.

Fig. C is a performance table for each frame interval over which frequencies are extracted. For example, an interval of 4 means that the temporal frequency was extracted every 4 frames for a total of 32 frames. We find that performance increased with higher frame intervals but saturated at 8 for all, except for FS.

Through these experiments, we use the Fourier transform to extract the temporal frequency magnitude from 32 frames preprocessed by a median filter.

D. Additional Analysis

D.1. Clarification of the Use of Temporal Information

To demonstrate that our pixel-wise temporal-frequency-based approach focuses on temporal information rather than encoding static appearance, we randomly permuted the order of the 32-frame input sequence and evaluated its resulting performance. As shown in Tab. B, our method shows a significant performance drop (i.e., -43.8%), compared to AltFreezing, demonstrating the strong dependency on the temporal information. These results confirm that, rather than encoding static appearance cues, our approach focuses on temporal dynamics.

D.2. Effectiveness of APM.

To evaluate the effectiveness of the APM in isolation, independent of modules like STE and TTE, we conducted experiments as shown in Table C. These experiments clearly demonstrate that incorporating the APM significantly improves performance. Specifically, the model without APM resulted in a lower performance, adding the APM alone led to a performance increase of up to 11.4%.

In Fig. D, we further analyzed APM activations by visualizing normalized heatmaps of APM regions for DF, FS, F2F, and NT. Regions are meaningfully different according to datasets, denoting no overfitting. Specifically, APM tends to focus on facial parts that were previously helpful to reveal deepfakes: the mouth for F2F, eyes and nose for FS.

In Fig. E-A, APM-selected patches show higher average temporal frequency magnitudes than surrounding areas, indicating stronger flickering effects.

The number of parts	CDF	DFDC	FSh	DFo	DFD	Avg.
0	88.2	74.7	**99.4**	$\underline{98.9}$	96.8	91.6
1	$\underline{89.3}$	74.7	$\underline{99.3}$	**99.4**	96.9	$\underline{91.9}$
3	88.4	**75.3**	$\underline{99.3}$	**99.4**	**97.3**	$\underline{91.9}$
5 (ours)	**89.7**	$\underline{75.2}$	$\underline{99.3}$	**99.4**	**97.3**	**92.2**
7	87.1	74.5	$\underline{99.3}$	**99.4**	$\underline{97.0}$	91.5

Table D. **Performance comparison by each number of parts proposed by APM.** We trained the model on FF++ and evaluated it for five unseen datasets.

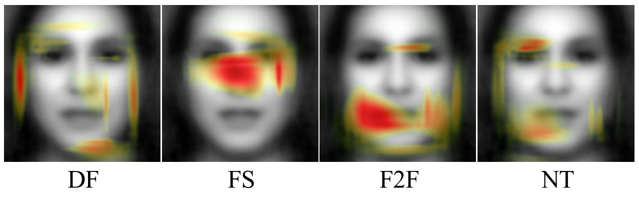


Figure D. **APM Over-Selection Heatmaps (red: regions selected more frequently than average) for each deepfake type.**

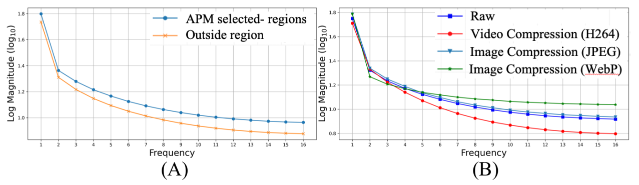


Figure E. **Pixel-wise Temporal Frequency Distributions in FF++**: (A) shows inside APM-selected patches versus the remaining area, (B) presents under different compression.

D.3. Ablation Study for the number of parts

Table D shows the performance of a model trained with varying numbers of parts extracted by APM across different datasets. We observe performance improvements after incorporating part-based frequency features from APM, particularly with a notable increase in performance on the DFo dataset, generally increasing with up to 5 parts and peaking at 5. However, performance declined with too many parts due to redundancy from observing similar regions repeatedly, with noticeable drops in performance on the DFDC and CDF datasets.

D.4. Ablation Study for the Feature Blending.

To analyze methods for integrating raw RGB features with temporal frequency features, we conducted experiments comparing different integration strategies in Table E. Our findings show that using 1×1 Conv. for feature blending improves performance while omitting it reduces adaptation

Feature blending	FSh	CDF	DFDC	DFo	Avg.
no blending	99.3	85.1	75.4	99.3	89.8
add	77.3	68.2	59.9	75.7	70.3
concatenate	99.0	85.4	74.4	99.4	89.6
1×1 Conv. (Ours)	99.3	**89.7**	75.2	99.4	**90.9**
1×1 Conv. ($\times2$)	**99.4**	87.1	**75.6**	**99.5**	90.4

Table E. **Analysis for feature blending methodologies.** We trained the model on FF++.

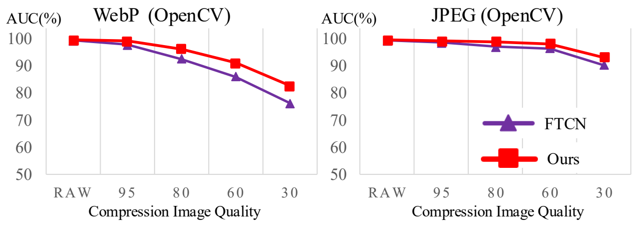


Figure F. **Evaluation of the robustness against WebP/JPEG.**

between RGB and temporal frequency domains, leading to a drop (e.g. no blending, add, concatenate). Additionally, increasing convolution depth offers no significant gain.

D.5. Additional Results for Perturbations Robustness.

We conducted a study on the robustness of our model against various perturbations as proposed in DFo [11]. As shown in Table F, our model shows impressive robustness against perturbations like resize and blur. This robustness indicates that our model is robust for frame-wise perturbation. Meanwhile, our method suffers from performance drops for perturbations that distort temporal information, such as compression and noise. These performance drops are due to temporal distortion interfering with the extraction of temporal frequency.

To evaluate our approach on more general compression, we conducted additional compression robustness experiments on WebP and JPEG compression. As shown in Fig. F, under severe degradation using WebP and JPEG compression, our method demonstrates superior robustness.

E. Discussion.

Fig. G denotes a failure case in which specular reflections on eyeglass lenses lead the model to misclassify genuine frames as forgeries. Moreover, as shown in Fig. 4 of the main paper and Fig. F, the temporal frequency generally remains robust under various video manipulations, such as resizing and saturation adjustments. However, heavy video and image compression can merge multiple neighboring pixels into a single subpixel, diminishing pixel-level motions and amplifying domain shifts in the temporal frequency.

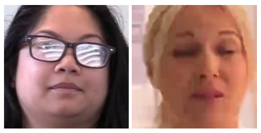


Figure G. **Failure Case.**

Fig. E-B compares the average pixel-wise temporal frequency under diverse image and video compression methods (H264, JPEG, and WebP). At low frequency, compressed images/videos follow an uncompressed signal (i.e., RAW) closely; while at high frequency, compressed images/videos increasingly diminish the spectrum. This attributes the performance drop to ours under severe compression. The performance drop under severe compression remains a limitation, and our future work may deal with this by exploring temporal-frequency regularization.

F. More Detail Setting

In this section, we will present the detailed setup of our experiments.

F.1. Datasets

We use the following forgery video datasets: (1) **FaceForensics++** (FF++) [18] consists of four face forgery methods (Deepfakes (DF), FaceSwap (FS), Face2Face (F2F), and NeuralTextures (NT)), with a total of 5000 videos (1000 real videos and 1000 fake videos for each method). (2) **Celeb-DF-v2** (CDF) [15] is a dataset consisting of 590 real videos and 5,639 fake videos that are synthesized by advanced techniques compared to FF++. (3) **DFDC-V2** (DFDC) [5], which has a 3,215 test video set, is more challenging than other datasets and was made under extreme conditions. (4) **FaceShifter** (FSh) [14] and (5) **DeeperForenscis-v1** (DFo) [11] contains high-quality forgery videos generated from the real videos from FF++. (6) **DeepFake Detection** (DFD) [7] is a popular benchmark dataset for forgery detection with 363 real videos and 3071 synthetic videos. (7) **Korean DeepFake Detection Dataset** (KoDF) [13] dataset is composed of 62,166 real videos and 175,776 fake videos generated using six face forgery methods. We employed a validation set for our experiments and utilized the initial 110 frames from each video for analysis.

F.2. Comparisons

To demonstrate the effectiveness of our method, we compare it with various types of deepfake detectors, including image-based and video-based detectors. Image-based detectors can be categorized into RGB-based and spatial frequency-based approaches. RGB-based detectors use RGB images to identify deepfakes, while spatial frequency-based approaches apply filters or Fourier transforms to extract spatial frequency, combining these with RGB information to improve detection.

Similarly, video-based detectors are categorized as RGB-based and stacked spatial frequency-based approaches. RGB-based methods in video detectors use individual RGB video frames for deepfake detection, whereas stacked spatial frequency-based approaches extract spatial frequencies (similar to the image-based frequency methods) and stack them temporally for enhanced detection.

Also, we categorize certain approaches based on whether they utilize external datasets. Specifically, LipForensics [8]

Method	Clean	Saturation	Contrast	Block	Noise	Blur	Resize	Compress	Avg.
Xception	99.8	99.3	98.6	99.7	53.8	60.2	74.2	62.1	78.3
CNN-aug	99.8	99.3	99.1	95.2	54.7	76.5	91.2	72.5	84.1
Patch-based	99.9	84.3	74.2	99.2	50.0	54.4	56.7	53.4	67.5
CNN-GRU	99.9	99.0	98.8	97.9	47.9	71.5	86.5	74.5	82.3
LipForensics*	99.6	99.3	98.8	98.7	64.3	96.7	95.8	90.9	92.1
FTCN*	99.5	98.0	93.7	90.1	53.8	95.0	94.8	83.7	87.0
Ours	99.7	99.1	95.8	91.9	55.0	97.3	97.5	88.0	89.2

Table F. **Average robustness for perturbations.** We present video-level AUC(%) for each perturbation.

and RealForensics [9] incorporate the LRW [3] dataset to learn representations of lip or facial motions, while StyleFlow [2] leverages a PsP encoder [17] pre-trained on the FFHQ [12] dataset to extract style latent.

We have reproduced several methods, including FTCN [20], CADDM [6], HFF [16], LipForensics [8], RealForensics [9], AltFreezing [19], and StyleFlow [2]. When available, we used their pre-trained weights for performance comparison, obtained from the official GitHub repositories. If the official weights did not align with our experimental settings, or if the official weights were unavailable—such as for CADDM, which was trained on FF++ with FSh—we retrained the models according to our specific experimental setup.

We report performance using Video-level AUC and Video-level EER, which are calculated by averaging the clip-level predictions for each clip input sequence within a video to obtain a single video-level prediction. The AUC and EER are then computed based on these aggregated video-level predictions.

F.3. Implementation Details

We use RetinaFace [4] to detect faces and perform face tracking with SORT [1]. When cropping faces, we picked a region based on the average point of the landmarks of the faces over 32 frames, cropped the same area for 32 frames, and fed our model these consecutive 32 frames as input.

Most experiments were conducted using four Nvidia A6000 48GB GPUs with an AMD Ryzen Threadripper Pro 3955WX 16-Cores CPU, or two A100 40GB GPUs with an Intel Xeon Gold 6240R CPU. In contrast, experiments using only 2D ResNet, such as Table 1 of the main paper, were conducted using four Nvidia RTX-3090 24GB GPUs with an Intel i9-10900X CPU.

References

[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016. 5

[2] Jongwook Choi, Taehoon Kim, Yonghyun Jeong, Seungryul Baek, and Jongwon Choi. Exploiting style latent flows for generalizing deepfake video detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1133–1143, 2024. 5

[3] Joon Son Chung and Andrew Zisserman. Lip reading in the wild. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II 13*, pages 87–103. Springer, 2017. 5

[4] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5203–5212, 2020. 5

[5] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The deepfake detection challenge (dfdc) dataset. *arXiv preprint arXiv:2006.07397*, 2020. 4

[6] Shichao Dong, Jin Wang, Renhe Ji, Jiajun Liang, Haoqiang Fan, and Zheng Ge. Implicit identity leakage: The stumbling block to improving deepfake detection generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3994–4004, 2023. 5

[7] Nick Dufour and Andrew Gully. Contributing Data to Deepfake Detection Research — ai.googleblog.com. `https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html`, 2019. [Accessed 30-07-2023]. 4

[8] Alexandros Haliassos, Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. Lips don't lie: A generalisable and robust approach to face forgery detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5039–5049, 2021. 4, 5

[9] Alexandros Haliassos, Rodrigo Mira, Stavros Petridis, and Maja Pantic. Leveraging real talking faces via self-supervision for robust forgery detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14950–14962, 2022. 5

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[11] 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. 4

[12] Tero Karras, Samuli Laine, and Timo Aila. A style-based gen-

erator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018. 5

[13] Patrick Kwon, Jaeseong You, Gyuhyeon Nam, Sungwoo Park, and Gyeongsu Chae. Kodf: A large-scale korean deepfake detection dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10744–10753, 2021. 4

[14] Lingzhi Li, Jianmin Bao, Hao Yang, Dong Chen, and Fang Wen. Faceshifter: Towards high fidelity and occlusion aware face swapping. *arXiv preprint arXiv:1912.13457*, 2019. 4

[15] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3207–3216, 2020. 4

[16] Yuchen Luo, Yong Zhang, Junchi Yan, and Wei Liu. Generalizing face forgery detection with high-frequency features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16317–16326, 2021. 5

[17] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 5

[18] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018. 2, 4

[19] Zhendong Wang, Jianmin Bao, Wengang Zhou, Weilun Wang, and Houqiang Li. Altfreezing for more general video face forgery detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4129–4138, 2023. 5

[20] Yinglin Zheng, Jianmin Bao, Dong Chen, Ming Zeng, and Fang Wen. Exploring temporal coherence for more general video face forgery detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15044–15054, 2021. 5