Supplementary Material:
Self-Supervised Video Forensics by Audio-Visual Anomaly Detection

A.1. Video Results

We provide some qualitative video results of some random samples from the FakeAVCeleb dataset [52] in our webpage with audio. We show "ground truth" outputs from the synchronization model and autoregressive predictions over time. We also show a score indicating the probability that the example is fake (Eq. (4), main paper). We use the time delay distribution as our feature set, and display the predictions as a heat map. Each step $\mathbf{x}_t$ of the predicted outputs was obtained by providing the ground truth features from times $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{t-1}$ into the autoregressive model.

A.2. Cross-dataset Generalization

We also experiment with the another audio-driven video editing method LipGAN [60]. We test on a test set consisting of 100 real videos and 100 fake videos, which are based on the real videos of test set from FakeAVCeleb [52]. We show results on Tab. 5. Our method leverages continuous time delay distribution as in Sec. 4.6 and outperforms many supervised methods although it is only trained on real videos, indicating that our method possesses certain generalization capability to different manipulation methods.

Method		Modality	LipGAN [60]	
			AP	AUC
Supervised (transfer)	Xception [87]	$\mathcal{V}$	67.6	65.5
	LipForensics [41]	$\mathcal{V}$	82.2	85.2
	AD DFD [112]	$\mathcal{AV}$	82.7	84.8
	FTCN [109]	$\mathcal{V}$	76.9	73.9
	RealForensics [40]	$\mathcal{V}$	**94.3**	**96.5**
Unsupervised	AVBYOL [38, 40]	$\mathcal{AV}$	65.0	73.0
	VQ-GAN [32]	$\mathcal{V}$	50.7	50.4
	Ours	$\mathcal{AV}$	**93.3**	**94.1**

Table 5. **Generalization to LipGAN method** [60]. AP scores (%) and AUC scores (%) are reported on real videos of test set from FakeAVCeleb [52]. Fake videos are manipulated by LipGAN [60]. Supervised methods are trained on FakeAVCeleb dataset [52]. Ours is trained on LRS2 [3] and uses the feature set of continuous time delay distribution. Best results are in **bold**.

A.3. Ablation Study

Feature activation. We study the influence of the number of principal components D for the variation of the anomaly detection model that projects the audio-visual feature activations into a lower dimensional space. We set D to $\{11, 31, 32, 64, 128, 256, 512\}$ and keep other hyperparameters the same. Fig. 6 shows that as the D increases, the accuracy of the anomaly detection model decreases. The reason might be that the higher dimensional prediction problem is also more challenging.

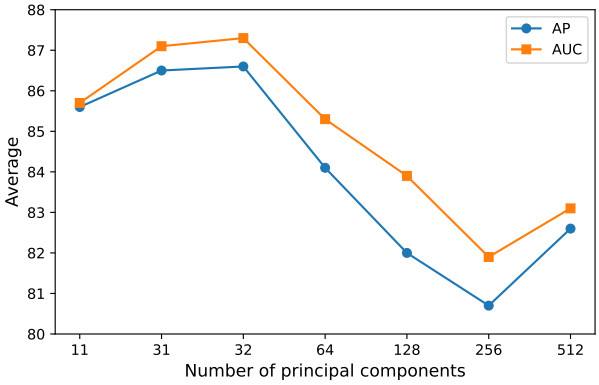

Figure 6. **Number of PCA projections.** We evaluate with different number of principal components for the model that obtains a feature set by projecting the audio-visual feature activations to a lower dimensional space using PCA. We report average AP scores (%) and AUC scores (%) on FakeAVCeleb dataset [52].

A.4. Feature Set Variations

We describe more details about building the autoregressive model on some of our feature sets in this section.

Binary cross entropy (BCE) model. We assume that the $2\tau+1$ possible time delays of each time step are independent, and use BCE loss for probability $S(i,j)$ of each time delay. Thus, we can decompose $p_\theta(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)$ as in Eq. (6):

$$p_\theta(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N) = \prod_{i=0}^{N-1} \prod_{q=1}^{2\tau+1} p_\theta\left(x_{i+1,q} | \mathbf{x}_1, \cdots, \mathbf{x}_i\right), \tag{6}$$

We then maximize $p_\theta\left(x_{i+1,q} | \mathbf{x}_1, \cdots, \mathbf{x}_i\right)$ using BCE loss:

$$L(\hat{\mathbf{x}}_i, \mathbf{x}_i) = -\sum_{j=1}^{2\tau+1} x_{i,j} \log(\hat{x}_{i,j}) + (1 - x_{i,j}) \log(1 - \hat{x}_{i,j}). \tag{7}$$

We constrain the prediction $\hat{x}_{i,j}$ to the range of $[0, 1]$ via a sigmoid function.

Discrete 2D probability model. The *discrete prob.* model generates the entire 2D frame/time delay matrix autoregressively in "raster scan" order, similar to models such as PixelCNN [90,98]. We unroll the time delay distribution sequence into 2D grid like a grayscale image as shown in Fig. 7. We use K-means ($K = 8$) clustering to quantize each entry and assume each probability $\hat{S}(i,j)$ of time delay not only depends on the previous time delay distributions but also the

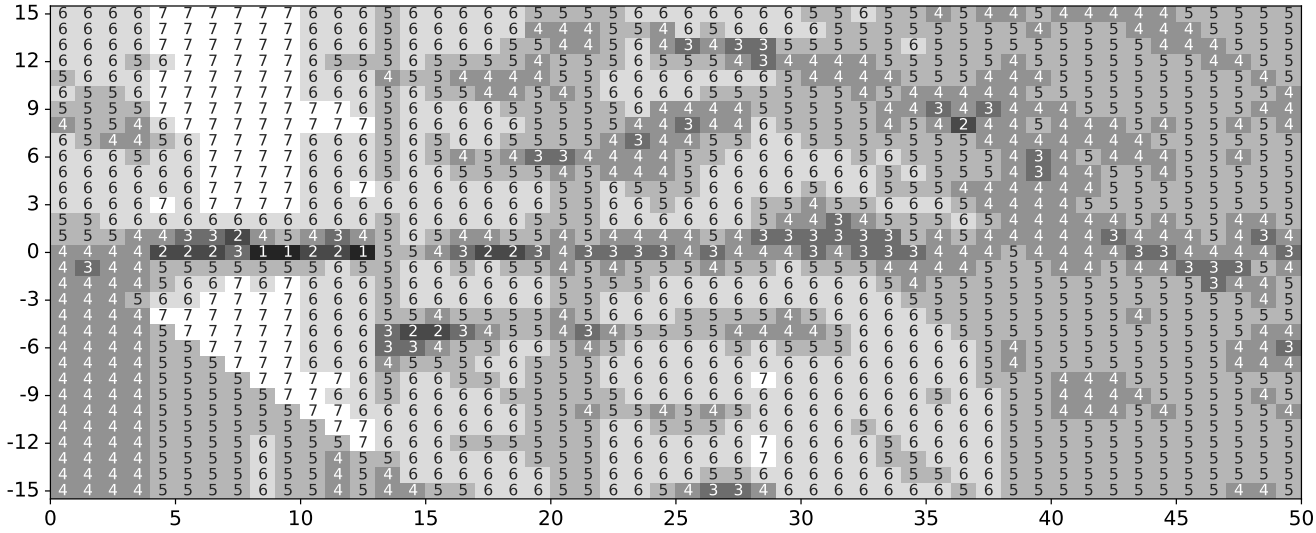

Figure 7. **Visualization of discrete probability grid.** We use K-means clustering to quantize the probability space to convert continuous synchronization probability $S(i,j)$ to discrete probability bin $\hat{S}(i,j)$. Then we build a autoregressive Transformer [99] model on the probability grid.

previous probabilities within the same distribution. Then we decompose $p_\theta(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)$:

$$p_\theta(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N) = \prod_{i=0}^{N-1} \prod_{q=1}^{2\tau+1} p_\theta(x_{i+1,q} | \mathbf{x}_1, \cdots, \mathbf{x}_i; x_{i+1,1}, \cdots, x_{i+1,q-1}). \tag{8}$$

We utilize a similar loss function in PixelCNN [90, 98] to supervise the autoregressive model.

A.5. Implementation Details

Audio-visual synchronization model. Following prior work [6, 18, 58], we utilize curriculum learning to train the audio-visual synchronization model. Specifically, we use the two-stage training procedure. During the first phase, the negatives are from different videos, while for the second stage, the negatives are sampled within the same videos randomly (the starting time steps are sampled randomly).

Anomaly detection model. We process each input vector $\mathbf{x}_i$ with an affine transformation, before passing it into the autoregressive model. This projects the input (e.g., a time delay distribution $S_i \in \mathbb{R}^{31}$) to $\mathbb{R}^{256}$. Also, we add an affine transformation to project the embedding back into the feature set space, e.g., $\mathbb{R}^{256} \rightarrow \mathbb{R}^{31}$ for the time delay distribution. We use learnable positional encodings for both the synchronization and autoregressive models.

Hyperparameters. For the synchronization model, we use Adam [54] with a learning rate of 1×10^{-4}, with a batch size of 16 for the first phase and 40 for the second. During the second stage, we sample four 5-frame short clips per video. For the autoregressive model (anomaly detection model), we use the Adam optimizer [54] with 1×10^{-3} learning rate, weight decay of 1×10^{-6} and warm-up and cosine learning rate decay [68] strategies. We use a batch size of 16 and the dropout [94] rate of 0.1. We train the synchronization model on 8 NVIDIA A40 GPUs and use a single GeForce RTX 2080 Ti GPU for the autoregressive model.

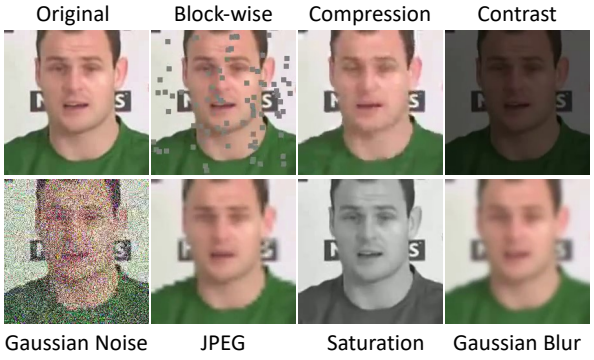

Figure 8. **Visualization of corrupted images.** Examples of the corruptions taken into account at intensity level 5. The set of corruptions is introduced in Jiang et al. [49], and it consists of color saturation, local block-wise distortion, color contrast, Gaussian blur, white Gaussian noise, JPEG compression, and video compression rate change.

A.6. Visualization of perturbed images

We visualize some images used for unseen perturbations robustness test in Fig. 8.

A.7. Temporal localization

To test the temporal localization ability of our method, we selected random 5-frame subsequences from real videos

of FakeAVCeleb [52] test set and manipulated frames using Wav2Lip [82] with audio (since none of our benchmark datasets provided relevant videos for the task). Our model predicts the fake score for each frame (Fig. 9). We can see that the score was significantly higher for manipulated frames, suggesting that our model is able to temporally localize manipulations. Besides, we can achieve top-5 accuracy of 92.0%. It is worth mentioning that all supervised and unsupervised baselines can not localize the manipulated frames temporally.

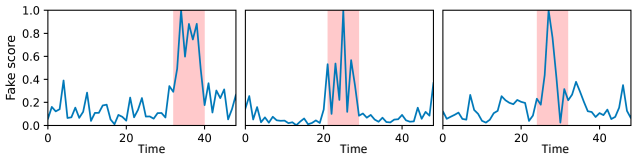

Figure 9. **Temporal localization.** We manipulate a random 9-time step interval (pink region). We show the fake score (negative log-likelihood) for each frame. We scale the y axis by the maximum score.

A.8. Robustness to background noise

We test our method when the audio stream of FakeAVCeleb [52] test set is added Gaussian noise under different signal-to-noise ratio (SNR). As shown in Fig. 10, our detector can basically maintain its performance when SNR decreases, indicating that our method is robust to the background Gaussian noise (our method has never seen audio Gaussian noise during training).

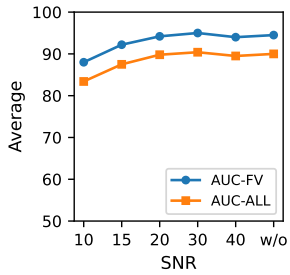

Figure 10. **Robustness to background noise.** AUC scores (%) of our detector as a function of signal-to-noise ratio (SNR) when Gaussian noise is added to the audio stream. "w/o" means no noise is added. AVG-ALL means the average over all the categories. AVG-FV represents the average over four fake video categories.