Evading DeepFake Detectors via Adversarial Statistical Consistency
(Supplementary Material)

Yang Hou1, Qing Guo2*, Yihao Huang3, Xiaofei Xie4, Lei Ma5,6, Jianjun Zhao1
1Kyushu University, Japan, 2Centre for Frontier AI Research (CFAR), A*STAR, Singapore,
3Nanyang Technological University, Singapore, 4Singapore Management University, Singapore
5University of Alberta, Canada, 6The University of Tokyo, Japan

In the main paper, we have reported the attack results on various DeepFake detectors and compared the transferability and image quality with four baseline attacks. In the supplementary material, (1) We describe in detail the polynomial model and the smoothing function used to generate the adversarial exposure. (2) For the adversarial blur, we visualize the attack effect of adversarial Gaussian blur with different kernel sizes. (3) We show more image quality comparisons with baseline attacks. (4) We provide more implementation details.

1. Polynomial Model

The model provided by Gao et al. [2] consists of two parts, the first part is a polynomial model for generating the exposure, and the second part is a smoothing formula that maintains the naturalness of the exposure. The polynomial model is shown below:

\[ \mathbf{E}_4 = \sum_{i=0}^{D} \sum_{j=0}^{D-t} a_{i,j} T_{\varphi}(x_i)^t T_{\varphi}(y_j)^j \]  

(1)

Where the “\(^{\text{\cdot}}\)” presents the logarithmic operations, \( \{a_{i,j}\} \) and \( D \) denote the parameters and degree of the polynomial model, respectively. The number of parameters are \(|\{a_{i,j}\}| = \frac{(D+1)(D+2)}{2} \), the lower degree \( D \) leads to less model parameters \( \{a_{i,j}\} \). \( T_{\varphi}(\cdot) \) is the offset transformation with \( \varphi \) being the control points. We denote \( i \) as the \( i \)-th pixel, and its corresponding coordinate as \( (x_i, y_i) \). The \( (T_{\varphi}(x_i), T_{\varphi}(y_i)) \) indicates that the exposure field is warped by offsetting the control points of each coordinate. In addition, The polynomial model with fewer parameters \( \{a_{i,j}\} \) leads to a smoother exposure field. For convenient representations, we concatenate \( \{a_{i,j}\} \) as \( \mathbf{a} \). We can maintain the smoothness of the generated exposure by restricting the \( \mathbf{a} \) and \( \varphi \), and the smoothing function can be written as follow:

\[ S(\mathbf{a}, \varphi) = -\lambda_a ||\mathbf{a}||_2^2 - \lambda_\varphi ||\nabla \varphi||_2^2 \]  

(2)

*Corresponding author: tsingguo@ieee.org

The first term is to maintain the sparsity of \( \mathbf{a} \) and thus ensure the smoothness of the generated adversarial exposure. The second term is to force the control point \( \varphi \) to vary in a smaller range, encouraging a minor warping of the adversarial exposure field. The hyper-parameters \( \lambda_a \) and \( \lambda_\varphi \) are used to regulate the balance between adversarial attack and smoothness. In detail, we initialize \( \mathbf{a} \) be a vector whose entries share a common value (e.g., 0), the \( \mathbf{a} \) and \( \varphi \) are then gradually optimized in each attack iteration to produce smooth and effective adversarial exposure.

2. Different Kernel Sizes

The success of adversarial Gaussian blur is dependent on the size of the Gaussian kernel and the value of \( \sigma \). While a larger Gaussian kernel and \( \sigma \) can improve the success rate of transfer attacks, they can also result in more blurry images. To illustrate this point, we present a visualization of the adversarial examples generated by adversarial Gaussian blur using different kernel sizes, as shown in Figure 1.

![Figure 1](image.png)

Figure 1. Adversarial examples generated by adversarial Gaussian blur with different kernel sizes. From left to right are the original image and the generated adversarial example with the Gaussian kernel size \( k \) set to 3, 5, and 7, respectively.

3. Additional Quality Comparisons

As shown in Figure 2, we provide more quality comparisons of the adversarial samples generated with the baseline attacks i.e. PGD [4], FGSM [3], MIFGSM [1] and VMIFGSM [5].
4. More Implementation Details

In our experiments, to balance the effectiveness of the attack and the quality of generated adversarial examples, we set the degree \( D \) of the polynomial model to 11, the learning rate of \( \alpha \) and \( \varphi \) is \( 10^{-2} \) and \( 10^{-3} \), respectively. For adversarial Gaussian blur, we set the Gaussian kernel size \( k \) to 3. All the experiments were performed on a server running Ubuntu 6.0.8-arch1-1 system on a 10-core 3.60 GHz i9-10850k CPU with 31 GB RAM and an NVIDIA GeForce RTX 3090 GPU with 24 GB memory.
References


