TAFIM: Targeted Adversarial Attacks against Facial Image Manipulations

Shivangi Aneja\textsuperscript{1}  Lev Markhasin\textsuperscript{2}  Matthias Nießner\textsuperscript{1}

\textsuperscript{1}Technical University of Munich  \textsuperscript{2}Sony Europe RDC Stuttgart

Figure 1. \textbf{Left}: We propose a novel approach to protect facial images from several image manipulation models simultaneously. Our method works by generating quasi-imperceptible perturbations using a learned neural network. These perturbations when added to real images force the face manipulation models to produce a predefined manipulation target as output (white/blue image in this case). Compared to existing methods that require an image-specific optimization, we propose to leverage a neural network to encode the generation of image specific perturbations, which is several orders of magnitude faster and can be used for real-time applications. \textbf{Right}: Without any protection applied, face image manipulation models can be misused to generate realistic fake images for malicious purposes.

Abstract

Face image manipulation methods, despite having many beneficial applications in computer graphics, can also raise concerns by affecting an individual’s privacy or spreading disinformation. In this work, we propose a proactive defense to prevent face manipulation from happening in the first place. To this end, we introduce a novel data-driven approach that produces image-specific perturbations which are embedded in the original images. The key idea is that these protected images prevent face manipulation by causing the manipulation model to produce a predefined manipulation target (uniformly colored output image in our case) instead of the actual manipulation. Compared to traditional adversarial attacks that optimize noise patterns for each image individually, our generalized model only needs a single forward pass, thus running orders of magnitude faster and allowing for easy integration in image processing stacks, even on resource-constrained devices like smartphones. In addition, we propose to leverage a differentiable compression approximation, hence making generated perturbations robust to common image compression. We further show that a generated perturbation can simultaneously prevent against multiple manipulation methods.

1. Introduction

The spread of disinformation on social media has raised significant public attention in the recent few years, due to its implications on democratic processes and society in general. The emergence and constant improvement of generative models, and in particular face image manipulation methods, has signaled a new possible escalation of this problem. For instance, face-swapping methods [10, 39]—often referred to as deepfakes—whose models are publicly accessible can be misused to generate non-consensual synthetic imagery with the potential to cause severe distress to the persons impacted. Other examples include face attribute manipulation methods [11, 12, 42] which change the appearance of real photos, thus generating fake images that might then be used for criminal activities [2].

As such powerful face image manipulation tools became easier to use and more widely available, many efforts to detect image manipulations were initiated by the research community [15]. This has led to the task of automatically detecting manipulations as a classification task where predictions indicate whether a given image is real or synthetically generated. Several learning-based approaches [4, 6, 8, 13, 14, 28, 29, 38, 45, 57, 62] have shown promising results in identifying manipulated images. Despite the success and high classification accuracies of these
2. Related Work

Image Manipulation Recent advances in image synthesis models have made it possible to generate detailed and expressive human faces [9, 21–24, 41, 52]. These images may be used for unethical activities or even fraud. Even more problematic can be the misuse of real face images to synthesize new ones. For instance, face-attribute modification techniques [11, 12, 42] and face-swapping models [10, 39] facilitate the manipulation of existing face images. Similarly, facial re-enactment tools [19, 25, 48, 53, 61] also use real images/videos to synthesize fake videos.

Facial Manipulation Detection The increasing availability of these image manipulation models calls for the need to reliably detect synthetic images in an automated fashion. Traditional facial manipulation detection leverages hand-crafted features such as gradients or compression artifacts, in order to find inconsistencies in an image [5, 17, 33]. While such self-consistency can produce good results, these methods are less accurate than more recent learning-based techniques [7, 8, 13, 14, 45], which are able to detect fake imagery with a high degree of confidence. In contrast to detecting forgeries, we aim to prevent manipulations from happening in the first place by rendering the respective manipulation models ineffective by introducing targeted adversarial attacks.

Adversarial Attacks Adversarial attacks were initially introduced in the context of classification tasks [16, 20, 37, 50] and eventually expanded to semantic segmentation and detection models [18, 56]. The key idea behind these methods is to make imperceptible changes to an image in order to disrupt the feature extraction of the underlying neural networks. To this end, adversarial attacks generate an individual noise pattern that is superimposed to a given input image. While these methods have achieved great success in fooling state-of-the-art vision models, one significant drawback is that optimizing a pattern for every image individually makes the optimization process quite slow. In order to address this challenge, generic universal image-agnostic noise patterns were introduced [35, 36]. This has shown to be effective for misclassification tasks but gives suboptimal results for generative models as we show in Sec. 4.

Manipulation Prevention Deep steganography and watermarking techniques [32, 51, 55, 63] can be used to embed an image-specific watermark to secure an image. For instance, FaceGuard [58] embeds a binary vector to the original image representative of a person’s identity and classifies whether the image is fake by checking if the watermark is intact after being used for face manipulation tasks. These methods, however, cannot prevent the manipulation of face images which is the key focus of our work.
At the same time, recent works aim to prevent image manipulations by exploiting adversarial attack techniques to break image manipulation models. Ruiz et al. [46] disrupt the output of deepfake generation models. Yeh et al. [59, 60] aim to nullify the effect of image manipulation models. Other approaches [30, 49] aim to disturb the output of face detection and landmark extraction steps, which are usually used as pre-processing by deepfake extraction methods. One commonality of these methods is that they optimize a pattern for each image separately which is computationally very expensive, thus having limited applicability for real-world applications like mobile phones or other resource-constrained devices. To this end, we propose a data-driven method that generates image-specific perturbations with conditional generative models, thus requiring orders of magnitude less computational effort than existing adversarial attacks works.

3. Proposed Method

Our goal is to prevent face image manipulations and simultaneously identify which model was used for the manipulation. That is, for a given face image, we aim to find an imperceptible perturbation that disrupts the generative neural network of a manipulation method such that a solid color image is produced as output instead of originally-intended manipulation. Algorithmically, this is a targeted adversarial attack where the predefined manipulation targets make it computationally very expensive, thus having limited applicability for real-world applications like mobile phones or other resource-constrained devices. To this end, we propose a data-driven method that generates image-specific perturbations with conditional generative models, thus requiring orders of magnitude less computational effort than existing adversarial attacks works.

3.1. Method Overview

We consider a setting where we are given $K$ manipulation models $\mathcal{M} = \{f^k_\Theta\}_{k=0}^{K-1}$ pretrained on $K$ face datasets $\{D_k\}_{k=0}^{K-1}$ with $N_k$ images each, i.e., $D_k = \{X_i\}_{i=0}^{N_k-1}$ corresponds to the manipulation model $f^k_\Theta$. Note that each method requires a separate dataset since the input for each manipulation model is also different. For a given RGB image $X_i \in \mathbb{R}^{H \times W \times 3}$ of height $H$ and width $W$, the goal is to find the optimal perturbation $\delta_i \in \mathbb{R}^{H \times W \times 3}$ that is embedded in the original image $X_i$ to produce a valid protected image $X^p_i \in \mathbb{R}^{H \times W \times 3}$. The manipulation model $f^k_\Theta$, which is parametrized by its neural network weights $\Theta$, is also given as input to the method. Note that we use $f^k_\Theta$ only to drive the perturbation optimization and do not alter its weights. For the given image $X_i$, the output synthesized by the manipulation model $f^k_\Theta$ is denoted as $\hat{Y}_{ik} \in \mathbb{R}^{H \times W \times 3}$. We define the uniformly-colored predefined manipulation targets for the $K$ manipulation models as $\mathcal{Y} = \{Y_{ik}^{\text{target}}\}_{k=0}^{K-1}$.

In order to protect face images and obtain image perturbations, we propose two main ideas: First, we optimize for a global perturbation pattern $\delta_i \in \mathbb{R}^{H \times W \times 3}$, which is generalized across the entire data distribution. That is, this global perturbation is the same for each input image.

Second, we propose a generative neural network $g_{\Phi}$ parameterized by its weights $\Phi$ that produces image-specific perturbations $\delta_i$. This generative model is conditioned on the previously optimized global perturbation $\delta_i$, as well as the real image $X_i$. Our intuition is that the global perturbation provides a strong prior for the global noise structure, thus enabling the conditional model to produce more effective perturbations. Finally, we propose to incorporate a differentiable JPEG module to ensure the robustness of the perturbations towards compression. An overview of our method is shown in Fig. 2.
Hence, we formulate the overall loss as
\[
L^{\text{Total}} = \sum_{k=0}^{K-1} \left[ \sum_{i=0}^{N_k-1} L_i^{\text{recon}} + \lambda L_i^{\text{perturb}} \right],
\]
where the parameter \( \lambda \) regularizes the strength of perturbation added to the real image, \( N_k \) denotes the number of images in the dataset for the \( k \)-th manipulation model, and \( K \) denotes the number of manipulation methods. \( i \) denotes the image indexing and \( k \) denotes the manipulation model. \( L_i^{\text{recon}} \) and \( L_i^{\text{perturb}} \) represent reconstruction and perturbation losses for \( i \)-th image.

### 3.2.1 Global Perturbation Optimization

The global perturbation is a fixed, image-agnostic perturbation that is added to all the images in the dataset to apply protection. We optimize for this global perturbation \( \delta_G \in \mathbb{R}^{H \times W \times 3} \) for the entire dataset by minimizing Eq. 1. The perturbation \( \delta_G \) protects all images in the dataset against the \( K \) manipulation models
\[
\delta^*_G = \arg\min_{\delta_G} L^{\text{Total}},
\]
where \( L^{\text{Total}} \) refers to total loss (Eq. 1). The protected image \( X_i^p \) is generated as:
\[
X_i^p = \text{Clamp}_\varepsilon(X_i + \delta_G).
\]
The Clamp\( _\varepsilon(\xi) \) function projects higher/lower values of \( \xi \) into the valid interval \([-\varepsilon, \varepsilon]\). The global perturbation \( \delta_G \) is initialized with a random vector sampled from a multivariate uniform distribution, i.e., \( \delta^0_G \sim U(0, 1) \) and optimized iteratively. For the generated protected image \( X_i^p \) and the given manipulation model \( \Phi \), the reconstruction loss \( L_i^{\text{recon}} \) is formulated as
\[
L_i^{\text{recon}} = \| \Phi(X_i^p) - Y_k \|_2^2.
\]
The perturbation loss \( L_i^{\text{perturb}} \) for a real image \( X_i \) is given as
\[
L_i^{\text{perturb}} = \| X_i - X_i^p \|_2^2.
\]
Finally, the overall loss for the global perturbation optimization can then be written as
\[
L^{\text{Total}} = \sum_{k=0}^{K-1} \left[ \sum_{i=0}^{N_k-1} \| \Phi(X_i^p) - Y_k \|_2^2 + \lambda \| X_i - X_i^p \|_2^2 \right].
\]

### 3.2.2 Conditional Generative Perturbation Model

In addition to the global perturbation noise, we propose a conditional generative neural network model that outputs image-specific perturbations. For this task, we use an encoder-decoder architecture based on a U-Net [43]. We learn the parameters \( \Phi \) of this conditional model \( g_\Phi \) in order to generate image-specific perturbations as
\[
\Phi^* = \arg\min_{\Phi} L^{\text{Total}}.
\]
The model \( g_\Phi \) is conditioned on the globally-optimized perturbation \( \delta_G \) as well as the original input image \( X_i \). Conditioning the model \( g_\Phi \) on \( \delta_G \) facilitates the transfer of global structure from the facial imagery to produce highly-efficient perturbations, i.e., these perturbations are more successful in disturbing manipulation models to produce results close to the manipulation targets. The real image \( X_i \) and global perturbation \( \delta_G \) are first concatenated channel-wise. \( X_i' = [X_i, \delta_G] \), to generate a six-channel input \( X_i' \in \mathbb{R}^{H \times W \times 6} \).

\( X_i' \) is then passed through the conditional model \( g_\Phi \) to generate image-specific perturbation \( \delta_i \), \( g_\Phi(X_i') \). These image-specific perturbations \( \delta_i \) are then added to the respective input images \( X_i \) with same clamping operation \( \text{Clamp}_\varepsilon \) to generate the protected image as
\[
X_i^p = \text{Clamp}_\varepsilon(X_i + \delta_i).
\]

For network architecture and further training details, we refer to the supplemental material.

### 3.2.3 Differentiable JPEG Compression

In many practical scenarios, images shared on social media platforms get compressed over the course of transmission. Our initial experiments suggest that protected images \( X_i^p \) generated from the previous steps can easily become ineffective by applying image compression. In order to make our perturbations robust, we therefore, propose to incorporate a differentiable JPEG compression into our generative model; i.e., we aim to generate perturbations that still disrupt the manipulation models even if the input is compressed. The actual JPEG compression technique [54] is non-differentiable due to the lossy quantization step (details in supplemental) where information loss happens with the round operation as, \( x := \text{round}(x) \). Therefore, we cannot train our protected images against the original JPEG technique. Instead, we leverage continuous and differentiable approximations [26,47] to the rounding operator. For our experiments, we use the sin approximation by Korus et al. [26]
\[
x := x - \frac{\sin(2\pi x)}{2\pi}.
\]
This differentiable round approximation coupled with other transformations from the actual JPEG technique can
be formalized into differentiable JPEG operation. We denote the full differentiable JPEG compression as $\Psi^q$, where $q$ denotes the compression quality.

For training, we first map the protected image $X^p_i$ to RGB colorspace $[0, 255]$ before applying image compression, obtaining $\tilde{X}^p_i$. Next, the image $\tilde{X}^p_i$ is passed through differential JPEG layers $\Psi^q$ to generate a compressed image $X^p_{ic}$, which is then normalized again as $X^p_i$ before passing it to the manipulation model $f_\Theta$.

Training with a fixed compression quality ensures robustness to that specific quality but shows limited performance when evaluated with different compression qualities. We therefore, generalize across compression levels by training our model with different compression qualities. Specifically, at each iteration, we randomly sample quality $q$ from a discrete uniform distribution $U_D(1, 99)$, i.e. $q \sim U_D(1, 99)$ and compress the protected image $X^p_i$ at quality level $q$.

This modifies the reconstruction loss $L_{\text{recon}}$ as follows

$$L_{\text{recon}} = \left\| f_\Theta(\Psi^q(X^p_i)) - Y^\text{target}_k \right\|_2,$$

and the overall loss is formulated as

$$L_{\text{Total}} = \sum_{k=0}^{K-1} \left( \sum_{i=0}^{N_k-1} \left\| f_\Theta(\Psi^q(X^p_{ic})) - Y^\text{target}_k \right\|_2 + \lambda \left\| X^p_i - X_{ic} \right\|_2 \right),$$

where $X^p_{ic} = \Psi^q(X^p_i)$ denotes the compressed protected image. Backpropagating the gradients through $\Psi^q$ during training ensures that the added perturbations survive different compression qualities. At test time, we evaluate results with actual JPEG compression technique instead of approximated/differential used during training to report the results.

4. Results

We compare our method against well-studied adversarial attack baselines I-FGSM [27] and I-PGD [34]. To demonstrate our results, we perform experiments with two different models: (1) pSp Encoder [42] which can be used for self-reconstruction and style-mixing (protected with solid white image as manipulation target), and (2) Sim-Swap [10] for face-swapping (protected with solid blue as manipulation target). For both these models, we use the publicly available pre-trained models. For pSp encoder, we use a model that is trained for a self-reconstruction task. The same model can also be used for style-mixing to synthesize new images by mixing the latent style features of two images. For style-mixing and face-swapping, protection is applied to the target image. We introduce a custom split on FFHQ [23] and CelebA-HQ [31] datasets for our experiments; see Tab. 1. All results are reported on the corresponding test sets for each task respectively.

### Experimental Setup

All images are first resized to $256 \times 256$ pixels. The global perturbation is optimized with a step size of 0.001 using the Adam optimizer for 75k iterations. For the face protection model $g_\Phi$, we use a UNet-64 encoder-decoder architecture, optimized for 100k iterations with a learning rate of 0.0001 and Adam optimizer. We use a batch size of 1 for all our experiments. For I-PGD, we use a step size of 0.01. Both I-FGSM and I-PGD are optimized for 100 steps for every image in the test split.

### Metrics

To evaluate the output quality, we compute relative performance at different perturbation levels, i.e., we plot a graph with the x-axis showing different perturbation levels for the image and the y-axis showing how close is the output of the face manipulation model to the predefined manipulation target. We plot the graph for two metrics: MSE (Mean Squared Error) and PSNR (Peak Signal To Noise Ratio). In the optimal setting, for a low perturbation in the image, the output should look identical to the manipulation target; i.e., a lower graph is better for MSE, and higher for PSNR. For qualitative results, we show the images from the corresponding line charts with minimum perturbation in the image, i.e., the leftmost points from the MSE graph.

### Baseline Comparisons

To compare our method against other adversarial attack baselines, we first evaluate the results of our proposed method on a single manipulation model without compression; i.e., neither training nor evaluating for JPEG compression. Visual results for self-reconstruction and style mixing are shown in Figs. 3 and 4. The performance for different perturbation levels is shown in Fig. 5. We observe that the model conditioned on the global perturbation as well as real images outperforms the model trained only with real images, indicating that the global perturbation provides a strong prior in generating more powerful perturbations.

### Robustness to JPEG Compression

Next, we investigate the sensitivity of perturbations to different compression

| Task       | Mode | # Images | Dataset                  |
|------------|------|----------|--------------------------|
| Self-Recon | Train| 5000     | FFHQ [23]                |
| Self-Recon | Val  | 1000     | CelebHQ [31]             |
| Face-Swap  | Train| $2 \times 5000$ | FFHQ [23] |
| Face-Swap  | Val  | $2 \times 1000$ | CelebHQ [31] |

Table 1. Dataset split for different tasks used in our experiments. For the self-reconstruction task, the manipulation model takes one image as input. For style-mixing and face-swapping, two images are fed as input to the manipulation model.
qualities. We apply the actual JPEG compression technique to report results. We observe that without training the model against different compression leads to degraded results when evaluated on compressed images, Fig. 7 and 8. Training the model with fixed compression quality makes the perturbation robust to that specific compression quality; however, it fails for other compression levels; see Fig. 6. We therefore train across different compression levels varied during training iterations.

Multiple Manipulation Models We can use the same model to simultaneously protect against multiple manipulation models by color-coding the manipulation targets for different models. This protection technique has an advantage over simple disruption since it gives more information about which technique was used to manipulate the image. For the sake of simplicity, we conduct experiments with two methods: self-reconstruction with solid white image as the manipulation target and face-swapping with solid blue image as target. Visual results for style-mixing and face-swapping are shown in Fig. 10, and the performance graph
Figure 5. Comparison with different optimization techniques evaluated on self-reconstruction (white target). We plot the output image quality (y-axis) corresponding to different levels of perturbations added to the image (x-axis). Orig and Protected refer to the original and protected image. Output refers to the output of the manipulation model and Target indicates the predefined manipulation target. Note that our method outperforms other baselines at all the different perturbation levels.

Figure 6. Comparison for our method trained without compression, fixed compression, and random compression for self-reconstruction task (white target). The fixed compression model was trained with compression quality C-80. All methods evaluated on compression quality C-30. The randomly compressed model outperforms fixed and no compression models.

is visualized in Fig. 9. Our combined model performs well without significant degradation when handling multiple manipulation methods.

**Runtime Comparison** We compare run-time performance against state-of-the-art in Tab. 2. I-FGSM [27] and I-PGD [34] optimize for perturbation patterns for each image individually at run time; hence they are orders of magnitude slower than our method. I-FGSM and I-PGD suffer from long run-times for each image; however, our model achieves similar performance in real-time.

Figure 7. Qualitative comparison in the presence of JPEG compression (white target). Methods trained without compression struggle; in contrast, our model trained with compression is able to produce perturbations that are robust to compression. Ours w/ Compression refers to the model trained with random compression. Ours w/o Compression refers to model trained without compression. Compression is applied on the protected images. All methods are evaluated at compression quality C-80.

Figure 8. Performance comparison in the presence of JPEG compression. Our method without differentiable JPEG training manages to disrupt the model; however, training with random compression levels significantly outperforms the uncompressed baselines. All methods are evaluated at compression quality C-80.
Figure 9. Comparison of individual models with the combined model. Ours (Self-Recon) refers to the model trained only with the self-reconstruction task. Ours (Face-Swap) refers to the model trained only for face-swapping. Ours (Both) refers to the single model trained for two tasks: self-reconstruction and face-swapping. Results are evaluated on face-swapping (left) and style-mixing (right).

| Method       | Time       | ±       |
|--------------|------------|---------|
| I-FGSM [27]  | 17517.71 ms| ±124.08 ms |
| I-PGD [34]   | 17523.01 ms| ±204.15 ms |
| Ours         | 10.66 ms   | ±0.21 ms |

Table 2. Run-time performance (averaged over 10 runs) to generate perturbation for a single image on the self-reconstruction task. Our method runs an order of magnitude faster than existing works that require per-image optimization. All timings are measured on an Nvidia Titan RTX 2080 GPU.

**Limitations** Even though our work is much faster than existing methods and can protect against multiple manipulation models simultaneously, we believe that there are still many ways to improve. In this work, we examined our method and other baselines in the white-box setting; i.e., we assume that we have detailed knowledge of the manipulation models and their parameters. However, in real-life scenarios the manipulation model might not be known, thus making it difficult to train the protection model directly. In these scenarios, black-box techniques could be leveraged, for instance, by learning the substitute model [40].

5. Conclusion

In this work, we proposed a data-driven approach to protect face images from being manipulated. Our method can prevent and simultaneously identify the manipulation technique by generating the predefined manipulation target as output. In comparison to existing works, our method is not only orders of magnitude faster but also achieves superior performance; i.e., with smaller perturbations of a given input image, we can achieve larger disruptions in the respective manipulation methods. In addition, we proposed an end-to-end compression formulation to make the perturbation robust to compression. In summary, we believe our generalized, data-driven method is an important stepping stone towards addressing the potential misuse of face image manipulation techniques.
Acknowledgments

This work is supported by a TUM-IAS Rudolf Mößbauer Fellowship, the ERC Starting Grant Scan2CAD (804724), and Sony Semiconductor Solutions Corporation. We would also like to thank Angela Dai for video voice over.

Supplemental Material

In this supplemental document, we provide additional details: the terminology used in the paper is further explained in Section A. The jpeg compression technique is explained in Section B. In Section C, we elaborate the implementation of the baseline methods. We then provide details about the network architecture in Section D. Finally, in Section E, we perform ablations and additional experiments.

A. Definitions

A.1. Targeted Adversarial Attack

Adversarial attacks [3] are typically used in the context of classification tasks as a technique to fool the classifier model by maximizing the probability of outputting an incorrect target class label other than its actual class. In a similar spirit, we formalize targeted attacks for face image manipulation as a technique to trick these generative models; however, in contrast to misleading a classifier, our goal is to produce a predefined target image (uniformly colored solid white/blue images in our experiments).

Recently, Ruiz et al. [46] proposed the term targeted disruption. Their key aspect is to use a specific target image to drive the optimization such that it destroys the output of the generative model. We on the other side propose to learn a perturbation that forces the manipulation model to produce a specific target image instead of a random output image; we refer to this as a targeted adversarial attack.

B. JPEG Compression

JPEG – Joint Photographic Experts Group – is a widely used lossy compression technique, standard in most digital cameras. JPEG is a lossy compression technique i.e. once an image is compressed, the original image cannot be reconstructed. However, the minute details removed from images are not visually noticeable with the naked eye, see Fig. 11. The general steps for the JPEG compression technique are shown in Fig. 12 and explained as follows.

The image is first transformed from RGB color space to YCbCr color space. YCbCr color space represents the image as its luminance (Y) and chrominance (Cb and Cr) components. The luminance channel encodes the brightness of the image. Cb represents the blueness and Cr represents the redness of the image. The YCbCr color space holds exactly the same amount of information as the original RGB image. Since the changes in the chrominance channels are not noticeable to the human eye, we can easily downsample the information in the Cb and Cr color channels in the subsampling step without any perceptible changes to the human eye, see Fig. 13. For experiments in the main paper, we downsample the chrominance channels by a factor of two.

Next, the image is split into the set of non-overlapping 8 × 8 pixel groups (Fig. 14). Thereafter, we perform Discrete Cosine Transform (DCT) on each of these 8 × 8 blocks independently. The two-dimensional DCT is given as

\[
C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} A_{x,y}(u, v),
\]

where

\[
A_{x,y}(u, v) = f(x, y) \cos \left(\frac{\pi(2x+1)u}{2N}\right) \cos \left(\frac{\pi(2y+1)v}{2N}\right)
\]

for \( u, v = 0, 1, 2, \ldots, N - 1 \) and

\[
\alpha(u) = \begin{cases} 
\sqrt{\frac{1}{N}} & \text{for } u = 0, \\
\frac{\sqrt{2}}{N} & \text{for } u \neq 0.
\end{cases}
\]
Using 2-D DCT, each $8 \times 8$ pixel group is encoded separately with its own 64 Discrete Cosine Transform (DCT) coefficients for the corresponding 64 basis functions shown in Fig. 15. The DCT coefficients represent the weight for each of the 64 DCT basis functions. An image usually contains a lot more low frequency information compared to high frequency details, therefore the low-frequency DCT coefficients have significantly higher values than the high-frequency coefficients.

These DCT coefficients are quantized using pre-computed quantization tables. Since JPEG compression works on the assumption that the more important details in the images lie in the low-frequency regime, it discards information from high frequency DCT coefficients. The amount of discarded coefficients depends on the compression quality, i.e. for a higher compression more coefficients are discarded. Note that information loss happens during the quantization step. More specifically, quantization is performed on the frequency-domain Cb and Cr channels with their respective quantization tables. JPEG quantizes the information using the round operation $x := \text{round}(x)$, which makes the full JPEG pipeline non-differentiable. Therefore, in the main paper we use the differentiable rounding approximation during training.

Finally, the image then gets decompressed. For decompressing, the inverse operations to those from the compression step are performed. First, the DCT coefficients are de-quantized, then inverse DCT is applied to convert from frequency domain to spatial domain. The inverse DCT transform is defined as

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u) \alpha(v) C(u, v) \cos \left( \frac{\pi(2x + 1)u}{2N} \right) \cos \left( \frac{\pi(2y + 1)v}{2N} \right)$$

for $x, y = 0, 1, 2, \ldots, N - 1$. Next, block merging is done and finally image is converted from the YCbCr color space.
to the RGB color space, which gives the final compressed image.

Alternate Methods for JPEG Approximation  Shin et al. [47] also use JPEG approximation to make adversarial attacks on images robust to compression. The key idea behind their approach is to first train multiple image classification models with images compressed at different compression qualities. Thereafter, they use an ensemble of these models to drive the generation of image-specific noise robust to compression. We however propose to add differentiable JPEG to our optimization pipeline, which is much faster and does not require training auxiliary models.

C. Baseline Implementations

We compare our method against adversarial attack baselines I-FGSM [27] and I-PGD [34]. These methods target classification tasks, where attack patterns are optimized for each image individually. To this end, we adapt their method for our targeted adversarial attack task on generative models. Similar to their original implementations, we define a loss that only consists of the reconstruction term; i.e.,

$$L^\text{Total} = L^\text{recon} = \| f_\Theta (X^p) - Y^\text{target} \|_2^2,$$

where $X^p$ refers to the protected image and $Y^\text{target}$ refers to the predefined manipulation target (solid color white/blue in our case). The amount of perturbation in the image can be controlled with the magnitude of the update step.

I-FGSM [27] (Iterative Fast Gradient Sign Method): The protected image is first initialized with the original image as:

$$X^p_0 := X,$$

and then updated iteratively as

$$X^p_{n+1} = \text{Clamp}_\zeta \{ X^p_n - \alpha \text{sign} (\nabla_X L^\text{recon} (X^p_n, Y^\text{target})) \} ,$$

where $\alpha$ is the perturbation strength, $\text{Clamp}_\zeta (\xi)$ clips the higher and lower values in the valid range $[-\epsilon, \epsilon]$ and $\text{sign}(\zeta)$ returns the sign vector of $\zeta$. We report results with 100 iterations.

I-PGD [34] Iterative Projected Gradient Descent): The protected image is obtained by the following update steps:

$$X^p_0 := X,$$

$$X^p_{n+1} = \Pi_S \{ X^p_n - \alpha \text{sign} (\nabla_X L^\text{recon} (X^p_n, Y^\text{target})) \}$$

where $\alpha$ is the step size and $\Pi_S(.)$ refers to the projection operator that projects onto the feasible set $S$. We report results with 100 iterations and a step size of 0.01.

D. Architecture Details

To train our model, we use a convolutional neural network based on U-Net [44] architecture. More specifically, we use UNet-64 architecture with 29.24M parameters, as shown in Fig. 16. The weights of the network are normal initialized with scaling factor of 0.02. The network takes a 6 channel input, channel-wise concatenated original image $X$, and the globally optimized perturbation $\delta_G$ to predict the image-specific perturbation $\delta_i$.

E. Additional Experiments

In this section, we report results for some additional experiments. We first analyze the distribution of different baselines in Subsection E.1. Then, we report the performance of our method for different norms in Subsection E.2. In Subsection E.3, we evaluate the robustness of our method when compression is applied multiple times to the image. We show visual results for high perturbation levels in Subsection E.4. Finally, in Subsection E.5, we compare different rounding approximations for quantization step for the differentiable JPEG compression implementation. For brevity, we show results on self-reconstruction task with white image as manipulation target.

E.1. Analysis of different baselines

We analyze the output distribution of the manipulation model for different methods in Fig. 17. We notice that per-image optimization techniques, like I-PGD [34] show a number of outliers with high mean squared error in the generated output images, whereas our proposed method shows much less variance. One such outlier is visualized in Fig. 18.

E.2. Ablations with different Norms: $L_1$, $L_2$, $L_\infty$

Our loss function/minimization objective is formulated in terms of the $L_2$ norm. In this section, we compare results of our method with $L_1$ and $L_\infty$ norm respectively. Visual results are shown in Fig. 20 and performance comparison in Fig. 19. We notice that $L_2$ norm significantly outperforms the other norms evenly distributing perturbation throughout the image making the changes in the image much less perceptually visible. In addition, it is more effective in erasing the image traces from the output of manipulation model.

E.3. Multi-level Compression

In a practical use case, an image might be compressed multiple times when shared on social media platforms. Therefore, the protection applied to images should be robust to consecutively applied compression. For simplicity, we show results for bi-level compression, i.e. the compression is applied twice to the image. We first apply a high
Figure 16. Architecture Overview: Network architecture used for experiments. Conv(.) and ConvTranspose(.) refer to the 2D convolution and 2D transposed convolution operation. For both Conv and ConvTranspose, we use a kernel size of $4 \times 4$. For conv layers, values in bracket refers to the input and output channels; e.g., Conv(6, 64) denotes a 6 channel input and 64 channel output. LReLU(0.2) denotes the LeakyReLU activation with a negative slope 0.2. Concat denotes the concatenation operation.

Figure 17. Results for the self-reconstruction task [42] with white image as manipulation target. We show violin plots for different methods visualizing the mean squared error of the manipulation model output with the predefined manipulation target image. Our method, denoted as Ours, outperforms alternate methods showing a lot less variance in the output distribution. In contrast, per-image optimization techniques such as I-PGD have long tail distribution indicating that the method is not equally effective for all the samples in the dataset.

Figure 18. Outlier case for the self-reconstruction task [42] with white image as manipulation target. In some cases, I-FGSM [27] and I-PGD [34] produce outliers with high mean squared error; cf. the high variance compared to our method in Figure 17.

E.4. Manipulation for higher perturbation levels

We show visual comparisons for lower perturbation levels since these are most useful for practical purposes. To this end, in the graphs shown in the main paper, we have compression (C-30) and thereafter a low compression (C-80) for evaluation. The second compression is applied to the first compressed image. Results are shown in Fig. 21. We show that our method is robust to multi-level compression as well.
analyzed our method at several different perturbation levels. In this section, we visualize our results for higher perturbation levels, which show more disturbance in the original images, but illustrated more significant disruptions for manipulation model predictions. Fig. 23 shows the visual results for the self-reconstruction task.

E.5. Ablations with different JPEG approximations

There are three different approaches to approximate the round operation used in the quantization step of the original JPEG compression technique. These are formalized as:

1. Cubic Approximation [47]
   \[ x := \lfloor x \rfloor + (x - \lfloor x \rfloor)^3. \]

2. Soft Approximation [26]
   \[ \hat{x} = x - \frac{\sin(2\pi x)}{2\pi}, \]
   \[ x := \left( \text{round}(x) - \hat{x} \right)_{\text{detach}} + \hat{x}, \]
   where "detach" indicates that no gradients will be propagated during backpropagation.

3. Sin Approximation [26]
   \[ x := x - \frac{\sin(2\pi x)}{2\pi} \]

   We also compare against the identity operation \( x := x \) as the baseline. The performance comparison for these methods is visualized in Fig. 22. We notice that Sin approximation is better in performance compared to other approximations, therefore we use it for experiments in the main paper.
Figure 21. Results for the self-reconstruction task with white image as manipulation target and for multi-level compression. First Compression denotes the first applied high compression C-30, Second Compression denotes the second compression (C-80) applied to the first compressed image. We apply different compression to evaluate robustness. Compression Difference denotes the difference between first and second compressed images amplified by 20X for visibility. Our method can efficiently handle multiple compression levels of different qualities.

Figure 22. PSNR performance for training our method with different approximations to round operation for JPEG compression for the self-reconstruction task [42] with white image as manipulation target. Sin approximation outperforms other approximations with comparable results for Soft approximation.

Figure 23. Visual results for higher perturbation levels for the self-reconstruction task [42] with white image as manipulation target. Even for a higher perturbation level, our proposed approach outperforms alternate methods.
References

[1] JPEG files and Colour. https://www.youtube.com/watch?v=n_uNPbdenRs. Accessed: 2015-04-21. 10

[2] Maisy Kinsley fake account. https://twitter.com/sokane1/status/1111023838467362816. Accessed: 2019-03-27. 1

[3] Targeted Adversarial Attacks. https://www.kaggle.com/c/nips-2017-targeted-adversarial-attack. Accessed: 2017-07-01. 9

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

[5] Shruti Agarwal and Hany Farid. Photo forensics from jpeg dimples. In 2017 IEEE Workshop on Information Forensics and Security (WIFS), pages 1–6, 2017. 2

[6] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting World Leaders Against Deep Fakes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, page 8, Long Beach, CA, June 2019. IEEE. 1

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

[8] Shivangi Aneja and Matthias Nießner. Generalized Zero and Few-Shot Transfer for Facial Forgery Detection. In ArXiv preprint arXiv:2006.11863, 2020. 1, 2

[9] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In International Conference on Learning Representations, 2019. 2

[10] Renwang Chen, Xuanhong Chen, Bingbing Ni, and Yanhao Ge. Simswap: An efficient framework for high fidelity face swapping. In MM ’20: The 28th ACM International Conference on Multimedia, pages 2003–2011. ACM, 2020. 1, 2, 5

[11] Yunjei Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 1, 2

[12] Yunjei Choi, Youngjung Uh, Jaegun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020. 1, 2

[13] Davide Cozzolino, Andreas Rössler, Justus Thies, Matthias Nießner, and Luisa Verdoliva. Id-reveal: Identity-aware deepfake video detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 15108–15117, October 2021. 1, 2

[14] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Rieß, Matthias Nießner, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. arXiv, 2018. 1, 2

[15] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The deepfake detection challenge (dfdc) dataset. 2020. 1

[16] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9185–9193, 2018. 2

[17] Pasquale Ferrara, Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Image forgery localization via fine-grained analysis of cfa artifacts. IEEE Transactions on Information Forensics and Security, 7(5):1566–1577, 2012. 2

[18] Volker Fischer, Mummadi Chaitanya Kumar, Jan Hendrik Metzen, and Thomas Brox. Adversarial examples for semantic image segmentation, 2017. 2

[19] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8649–8658, June 2021. 2

[20] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. 2

[21] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In International Conference on Learning Representations, 2018. 2

[22] Tero Karras, Miika Aittala, Samuli Laine, Erik Häkkinen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Aliiia:free generative adversarial networks. In Proc. NeurIPS, 2021. 2

[23] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019. 2, 5

[24] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020. 2
[25] 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* 2018 (TOG), 2018. 2

[26] Pawel Korus and Nasir Memon. Content authentication for neural imaging pipelines: End-to-end optimization of proof provenance in complex distribution channels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 4, 13

[27] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *ArXiv*, abs/1611.01236, 2017. 5, 7, 8, 11, 12

[28] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5001–5010, 2020. 1

[29] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts, 2018. 1

[30] Yuezun Li, Xin Yang, Baoyuan Wu, and Siwei Lyu. Hiding faces in plain sight: Disrupting ai face synthesis with adversarial perturbations. *ArXiv*, abs/1906.09288, 2019. 2, 3

[31] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 5

[32] Xiyang Luo, Ruohan Zhan, Huwen Chang, Feng Yang, and Peyman Milanfar. Distortion agnostic deep watermarking. 2020 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13545–13554, 2020. 2

[33] Siwei Lyu, Xunyu Pan, and Xing Zhang. Exposing region splicing forgeries with blind local noise estimation. *International Journal of Computer Vision*, 110(2):202–221, 2014. 2

[34] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2018. 5, 7, 8, 11, 12

[35] Jan Hendrik Metzen, Mummad chaiithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *submitted*, 2017. 2

[36] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[37] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016. 2

[38] Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019. 1

[39] Yuval Nirkin, Yosi Keller, and Tal Hassner. FSGAN: Subject agnostic face swapping and reenactment. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7184–7193, 2019. 1, 2

[40] Nicolas Papernot, Patrick McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017. 8

[41] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Gaugan: Semantic image synthesis with spatially adaptive normalization. In *ACM SIGGRAPH 2019 Real-Time Live!, SIGGRAPH ’19*, New York, NY, USA, 2019. Association for Computing Machinery. 2

[42] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Shay 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)*, June 2021. 1, 2, 5, 12, 14

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 4

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. cite arxiv:1505.04597Comment: conditionally accepted at MICCAI 2015. 11

[45] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics++: Learning to detect manipulated facial images. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 2

[46] Nataniel Ruiz, Sarah Adel Bargal, and Stan Sclaroff. Disrupting deepfakes: Adversarial attacks against conditional image translation networks and facial manipulation systems. 2020. 2, 3, 9

[47] Richard Shin. Jpeg-resistant adversarial images. 2017. 4, 11, 13

[48] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *Conference on Neural Information Processing Systems (NeurIPS)*, December 2019. 2

[49] Pu Sun, Yuezun Li, Honggang Qi, and Siwei Lyu. Landmark breaker: Obstructing deepfake by disturbing landmark extraction. In *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2020. 2, 3

[50] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014. 2

[51] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegasamp: Invisible hyperlinks in physical photographs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[52] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics* 2019 (TOG), 2019. 2

[53] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. Face2face: Real-time face capture and reenactment
of rgb videos. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, *IEEE*, 2016. 2

[54] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. 4

[55] Eric Wengrowski and Kristin Dana. Light field messaging with deep photographic steganography. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1515–1524, 2019. 2

[56] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *International Conference on Computer Vision*. *IEEE*, 2017. 2

[57] Xin Yang, Yuezun Li, and Siwei Lyu. Exposing deep fakes using inconsistent head poses. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019. 1

[58] Yuankun Yang, Chenyue Liang, Hongyu He, Xiaoyu Cao, and Neil Zhenqiang Gong. Faceguard: Proactive deepfake detection, 2021. 2

[59] Chin-Yuan Yeh, Hsi Chen, Shang-Lun Tsai, and Shang-De Wang. Disrupting image-translation-based deepfake algorithms with adversarial attacks. 2020 IEEE Winter Applications of Computer Vision Workshops (WACVW), pages 53–62, 2020. 2, 3

[60] Chin-Yuan Yeh, Hsi-Wen Chen, Hong-Han Shuai, De-Nian Yang, and Ming-Syan Chen. Attack as the best defense: Nullifying image-to-image translation gans via limit-aware adversarial attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16188–16197, October 2021. 2, 3

[61] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models, 2019. 2

[62] Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S. Davis. Two-stream neural networks for tampered face detection, Jul 2017. 1

[63] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *ECCV*, 2018. 2