Adversarial Perturbations Fool Deepfake Detectors

Apurva Gandhi
Department of Electrical and Computer Engineering
University of Southern California
Los Angeles, USA
apurvaga@usc.edu

Shomik Jain
Department of Electrical and Computer Engineering
University of Southern California
Los Angeles, USA
shomikja@usc.edu

***Abstract*—This work uses adversarial perturbations to enhance deepfake images and fool common deepfake detectors. We created adversarial perturbations using the Fast Gradient Sign Method and the Carlini and Wagner L_2 norm attack in both blackbox and whitebox settings. Detectors achieved over 95% accuracy on unperturbed deepfakes, but less than 27% accuracy on perturbed deepfakes. We also explore two improvements to deepfake detectors: (i) Lipschitz regularization, and (ii) Deep Image Prior (DIP). Lipschitz regularization constrains the gradient of the detector with respect to the input in order to increase robustness to input perturbations. The DIP defense removes perturbations using generative convolutional neural networks in an unsupervised manner. Regularization improved the detection of perturbed deepfakes on average, including a 10% accuracy boost in the blackbox case. The DIP defense achieved 95% accuracy on perturbed deepfakes that fooled the original detector while retaining 98% accuracy in other cases on a 100 image subsample.**

***Index Terms*—Deepfakes, Adversarial perturbations, Lipschitz regularization, Deep Image Prior, Image restoration**

I. Introduction

This work enhances deepfakes with adversarial perturbations to fool common deepfake detectors. *Deepfakes* replace a “source” individual in an image or video with a “target” individual’s likeness using deep learning [1]. *Adversarial perturbations* are modifications made to an image in order to fool a classifier. An adversary can choose these perturbations to be small so that the difference between the perturbed and original images is visually imperceptible. Figure 1 shows a deepfake generated from source and target images as well as its adversarially perturbed version. A deepfake detector correctly classifies the original as fake but fails to detect the perturbed deepfake which looks almost identical. In our results, detectors achieved over 95% accuracy on unperturbed deepfakes, but less than 27% accuracy on perturbed deepfakes.

Deepfakes have been used for many malicious applications. In 2019, an app called DeepNude was released which could take an image of a fully-clothed woman and generate an image with her clothes removed [2]. Furthermore, Facebook found over 500 accounts spreading pro-President Trump, anti-Chinese government messages using deepfake profile pictures [3]. These harmful uses of deepfakes violate individuals’ identity and can also propagate misinformation, especially on social media. Ahead of the 2020 U.S. election, Facebook and Twitter stated plans to try and remove certain deepfakes [4]. But adversarial perturbations can compromise the performance of deepfake detection methods used on these platforms.

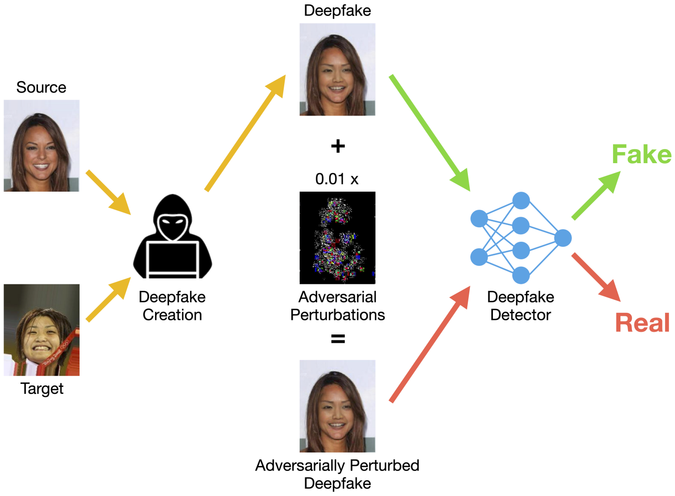


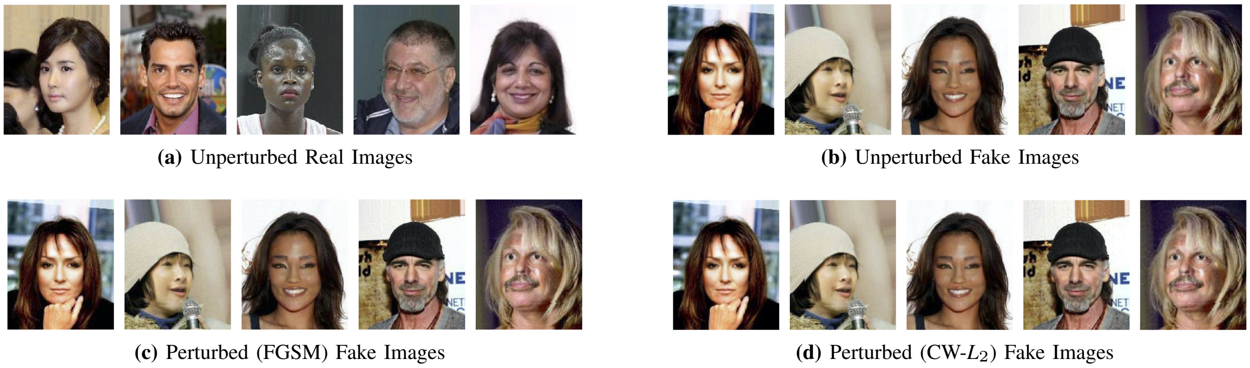
Fig. 1. Deepfake creation and adversarial perturbation. Example of creating a deepfake from source and target faces and adding adversarial perturbations to it. A deepfake detector correctly classifies the deepfake as fake but classifies the adversarially perturbed deepfake as real.

To defend against these perturbations, we explore two improvements to deepfake detectors: (i) Lipschitz regularization, and (ii) Deep Image Prior. *Lipschitz regularization*, introduced in [5], constrains the gradient of the detector with respect to the input data. We use *Deep Image Prior* (DIP), originally an image restoration technique [6], to remove perturbations by iteratively optimizing a generative convolutional neural network in an unsupervised manner. To our knowledge, this is the first application of DIP for removing adversarial perturbations. Overall, the contributions of this work aim to highlight the vulnerability of deepfake detectors to adversarial attacks, as well as present methods to improve robustness.

II. Deepfake Creation and Detection

We focus on deepfake images of celebrity faces as the scope of this work. Our dataset consists of 10,000 images: 5,000 real and 5,000 fake. The 5,000 real images were randomly sampled from the CelebA dataset [7]. Fig. 2 includes examples of real and fake images from our dataset.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE



(a) Unperturbed Real Images

(b) Unperturbed Fake Images

(c) Perturbed (FGSM) Fake Images

(d) Perturbed (CW-L_2) Fake Images

Fig. 2. Examples of real, fake, and perturbed images. Adversarially perturbed deepfakes (c and d) look similar to unperturbed deepfakes (b). Some fake images look more realistic than others. We use the ResNet model to perturb the first 3 fake images and the VGG model to perturb the last 2 fake images. All adversarial examples shown fool both models.

A. Deepfake Creation

Most deepfake creation methods use generative adversarial networks (GAN) to replace the face of a "source" individual with that of a "target" individual [1]. The generator in these methods consists of an encoder-decoder based network. First, the methods train a common encoder but different decoders for each face. Then, the source image is passed through the common encoder and the target's decoder to create a deepfake. A shortcoming of these methods is that training the encoder and decoder networks requires many images of both the source and target individuals. Creating a dataset of deepfakes using these methods is difficult: We would require numerous images for each individual and would also have to train separate decoders for each target.

Instead, we created the 5,000 fake images in our dataset using an existing implementation called the "Few-Shot Face Translation GAN" [8]. This implementation takes inspiration from Few-Shot Unsupervised Image-to-Image Translation (FUNIT) [9] and Spatially-Adaptive Denormalization (SPADE) [10]. FUNIT transforms an image from a source domain to look like an image from a target domain. Moreover, it does so using only a single source image and a small set of (or even a single) target image. It achieves this by simultaneously learning to translate between images sampled from numerous source and target domains during training; this allows FUNIT to generalize to unseen source and target domains at test time [9]. SPADE is a normalization layer that conditions normalization on an input image segmentation map in order to preserve semantic information [10]. The Few-Shot Face Translation GAN adds SPADE units to the FUNIT generator allowing us to create a deepfake using only a single source and target image.

B. Detection Methods

Common deepfake detection methods use convolutional neural networks (CNNs) to classify images as "real" or "fake" [1]. Prior work has shown that the VGG [11] and ResNet [12] CNN architectures achieve high accuracy for detecting deepfakes from a variety of creation methods [1]. The original architectures for both these models have thousand-dimensional output vectors. Instead, we replaced the last layer of these architectures to output two-dimensional softmax vectors corresponding to the real and fake classes. We chose a softmax vector over a sigmoid scalar to make the models compatible with the Carlini and Wagner L_2 norm attack discussed in section III-B.

We tested the VGG-16 and ResNet-18 architectures on our dataset. The models achieved train accuracies of 99.9% and 94.7% as well as test accuracies of 99.7% and 93.2%, respectively. These results are based on a 75%-25% train-test split after 5 epochs of training with a batch size of 16. Table I reports Area Under the Receiver Operating Characteristic (AUROC) curve values and additional performance metrics for these deepfake detectors.

III. Adversarial Attacks

Deep neural networks and many other pattern recognition models are vulnerable to adversarial examples – input data that has been perturbed to make the model misclassify the input [13]. The adversary can further craft these *adversarial perturbations* to have small magnitude so that the adversarial examples are difficult to distinguish from the original unperturbed input data. We tested the effect of adversarial perturbations on deepfake detectors using the following two attacks: the Fast Gradient Sign Method (FGSM) [14] and the Carlini and Wagner L_2 Norm attack (CW-L_2) [15]. We chose FGSM to try a popular, efficient attack and CW-L_2 to try a slow but stronger attack. This work only considers perturbations of fake images: An adversary's goal is to manipulate deepfakes so that they are classified as real and not vice versa. Concurrent work has extended adversarial perturbations of deepfakes with additional attacks [16] and even to videos [17].

A. Fast Gradient Sign Method (FGSM)

Let $\mathbf{x}$ be the vector of pixel values in an input image and $\mathbf{y}$ be the corresponding true target class value. Let $J(\mathbf{x}, \mathbf{y}, \theta)$ be the training loss function (e.g. categorical cross-entropy loss for a softmax classifier) where θ represents the parameters of the model. FGSM exploits the gradient of the loss with respect to the input, $\nabla_x J(\mathbf{x}, \mathbf{y}, \theta)$, to generate the adversarial example, $\mathbf{x}_{adv}$:

$$\mathbf{x}_{adv} = \mathbf{x} + \epsilon \ \mathrm{sign}(\nabla_x J(\mathbf{x}, \mathbf{y}, \theta)). \tag{1}$$

Here, ϵ is a hyperparameter that controls the magnitude of the perturbation per pixel. By keeping ϵ small, we can limit the magnitude of the perturbations and thus minimize visual distortions in the adversarial examples. In practice, the pixel values of the adversarial examples are further clipped to a range of floating point values between 0 and 1. We used an ϵ value of 0.02 to generate our FGSM adversarial examples. This value was chosen after evaluating the attack effectiveness and visual distortions for several ϵ values in the range [0.01, 0.10].

To see why this attack is effective in causing a misclassification, we examine the linear approximation of the loss using its Taylor series expansion:

$$\begin{aligned} J(\mathbf{x}_{adv}, \mathbf{y}, \theta) \approx\ & J(\mathbf{x}, \mathbf{y}, \theta) \\ & + \epsilon \, \nabla_x J(\mathbf{x}, \mathbf{y}, \theta)^T \, \mathrm{sign}(\nabla_x J(\mathbf{x}, \mathbf{y}, \theta)). \end{aligned} \tag{2}$$

Using the sign function of the gradient ensures that the dot product in the second term of (2) is non-negative. Thus, FGSM chooses the perturbation that causes the maximum increase in the value of the linearized loss function subject to the ϵ pixel-perturbation control parameter.

B. Carlini and Wagner L_2 Norm Attack (CW-L_2)

This attack simultaneously minimizes two objectives. Let **x'** be a perturbed image. The first objective is to minimize the L_2 norm of the perturbation:

$$\min_{x'} \{ \|\mathbf{x'} - \mathbf{x}\|_2^2 \}. \tag{3}$$

The second objective tries to make the perturbation cause a misclassification. Let $\mathbf{Z}(\mathbf{x})$ represent the pre-softmax vector output (or logits) of a multi-class neural network classifier. The second objective is as follows:

$$\begin{gathered} \min_{x'} \{ f(\mathbf{x'}) \} \\ \text{where } f(\mathbf{x'}) = \max(\max_{i \neq y} \{ \mathbf{Z}(\mathbf{x'})_y - \mathbf{Z}(\mathbf{x'})_i \}, -\kappa). \end{gathered} \tag{4}$$

Here, i and y index into $\mathbf{Z}(\mathbf{x'})$ with y being the index of the true target class. By minimizing $f(\mathbf{x'})$, we try to maximize the difference between the logit of an incorrect class and the logit of the true class. Since the predicted class corresponds to the maximum logit, minimizing $f(\mathbf{x'})$ effectively tries to cause a misclassification. κ is a parameter that defines a threshold by which the logit corresponding to the incorrect predicted class should exceed the logit of the true target class.

The attack also performs a change of variable from **x'** to $\boldsymbol{\omega}$:

$$\mathbf{x'} = \frac{1}{2}(\tanh(\boldsymbol{\omega}) + 1). \tag{5}$$

This ensures that the perturbed image (**x'**) has floating point pixel values between 0 and 1. Putting (3), (4) and (5) together, we obtain the CW-L_2 attack:

$$\begin{aligned} \boldsymbol{\omega}^* &= \arg\min_{\omega} \{ \|\mathbf{x'} - \mathbf{x}\|_2^2 + c\, f(\mathbf{x'}) \} \\ \mathbf{x}_{adv} &= \frac{1}{2}(\tanh(\boldsymbol{\omega}^*) + 1). \end{aligned} \tag{6}$$

c is positive and controls the relative strength of the two objectives. In practice, c is chosen using a modified binary search which finds the smallest value of c in a provided range, such that $f(\mathbf{x}_{adv})$ is less than 0. This search along with the iterative gradient descent optimization process makes the attack very slow. However, this attack breaks many previously proposed defenses against adversarial examples [13]. For further details about the attack, we refer the reader to the CW-L_2 paper [15] and the implementation we used [18].

For all adversarial examples generated using this method, we chose $[10^2, 10^4]$ as the range for c with 5 search steps. We performed a maximum of 1000 iterations for optimization with a learning rate of 0.01. We used 200 for the value of κ. The value of κ was chosen by trying out values in the range $[0, 500]$. The range of c was chosen by initially performing attacks using a range of $[10^{-10}, 10^{10}]$ and then narrowing down the range to include the values of c most commonly chosen by the search steps. The values for κ and range of c were evaluated objectively based on the decrease in accuracy of the classifier under attack and subjectively based on the amount of visible distortions in the perturbed images. We left all other parameters to the defaults recommended by the implementation [18].

C. Attack Types

Adversarial attacks on machine learning models fall into two types depending on the amount of information available to the adversary about the model under attack:

- *Whitebox Attack*: The adversary has complete access to the model under attack, including the model architecture and parameters. It may be unlikely for an adversary to have access to model parameters in many scenarios. However, machine learning solutions such as deepfake detectors often use existing, publicly known and accessible architectures for transfer learning purposes [1].
- *Blackbox Attack*: The adversary has limited or almost no information about the model under attack. Previous research [14], [19], [20] has shown that adversarial examples created using whitebox attacks on one model also damage performance of different models trained for the same task. Furthermore, these attacks do not even have to be in the same family of classifiers. For example, adversarial examples created using a neural network also work on support vector machines and decision tree classifiers [20]. This *transferability* of adversarial examples is what makes blackbox attacks possible. Blackbox attacks can involve varying degrees of access to the model under attack, such as access to the predicted probabilities, predicted class or even the training data [21]. This work assumes the last of these

and performs blackbox attacks on the VGG model by creating whitebox examples for the ResNet model. Similarly, blackbox examples for the ResNet model are generated by creating whitebox examples for the VGG model. We note that since our models obtained over 94% training accuracy, access to ground-truth class information is almost equivalent to access to only the predicted class information.

D. Attack Results

Adversarial attacks significantly reduced the performance of both the VGG and ResNet deepfake detection models. We compare results on datasets of unperturbed and perturbed fake images created using the test set. The datasets exclude real images since they were not perturbed. Fig. 3 and Table II show the adversarial attack results.

For unperturbed fake images, VGG achieved an accuracy of 99.7% and ResNet achieved 95.2%. In the blackbox FGSM case, the accuracy decreased to 8.9% for VGG and 20.8% for ResNet. Blackbox CW-L_2 reduced the accuracy of VGG to 26.6% and ResNet to 4.6%. In the whitebox FGSM case, the accuracy dropped to 0.0% for VGG and 7.5% for ResNet. Whitebox CW-L_2 lowered the accuracy of both VGG and ResNet to 0.0%. As in section II-B, these results are based on a 75%-25% train-test split.

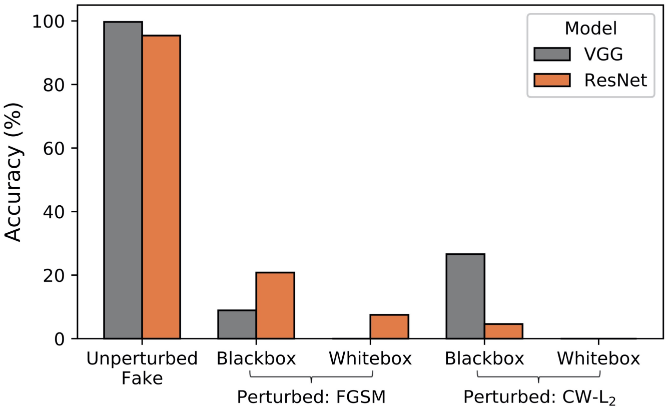


Fig. 3. Adversarial attack results. VGG and ResNet accurately detected unperturbed fake images but performed significantly worse on adversarially perturbed fake images. Blackbox attacks were less effective than whitebox attacks. CW-L_2 was generally more effective than FGSM.

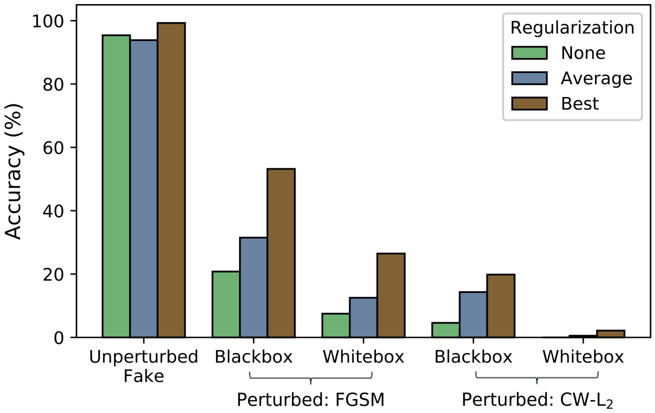


Fig. 4. Regularization results. Regularization improved the detection of adversarially perturbed deepfakes overall. Regularization mostly maintained accuracy on unperturbed images. Results are plotted for the average and best performances among models with varying regularization strength.

TABLE I. Unperturbed Data Results

Model	Accuracy	AUROC	Fake		Real	
			Precision	Recall	Precision	Recall
VGG	99.7%	99.9%	99.8%	99.7%	99.7%	99.8%
ResNet	93.2%	97.9%	91.5%	95.4%	95.2%	91.1%
ResNet (Regularized: λ=5)	95.0%	99.5%	91.5%	99.3%	99.2%	90.8%
ResNet (Regularized λ=50)	94.1%	98.6%	96.6%	91.4%	91.8%	96.8%
ResNet (Regularized λ=500)	87.5%	92.3%	85.3%	90.6%	89.9%	84.4%
ResNet (Regularized λ=5000)	96.2%	99.1%	98.0%	94.2%	94.5%	98.1%
ResNet (Average Regularized)	93.2%	97.4%	92.9%	93.9%	93.9%	92.5%
ResNet (Data Augmented)	98.7%	99.9%	98.5%	98.9%	98.9%	98.5%

Note: Unperturbed results listed for a test dataset containing 1,250 images each of real and fake classes.

TABLE II. Adversarial Attack Results

Model	Unperturbed	Perturbed: FGSM		Perturbed: CW-L_2	
		Blackbox	Whitebox	Blackbox	Whitebox
VGG	99.7%	8.9%	0.0%	26.6%	0.0%
ResNet	95.4%	20.8%	7.5%	4.6%	0.0%
ResNet (Regularized λ=5)	99.3%*	42.2%	26.5%*	14.5%	0.0%
ResNet (Regularized λ=50)	91.4%	17.8%	12.7%	6.2%	0.0%
ResNet (Regularized λ=500)	90.6%	53.2%*	9.0%	19.8%*	0.0%
ResNet (Regularized λ=5000)	94.2%	12.8%	1.9%	16.7%	2.2%*
ResNet (Average Regularized)	93.9%	31.5%	12.5%	14.3%	0.5%
ResNet (Data Augmented)	98.9%	17.0%	2.2%	3.8%	0.1%

*Note: Adversarial attacks conducted using only the fake images. *Best regularized performances.*

Whitebox attacks were more effective than blackbox attacks. This is expected because whitebox attacks have complete access to the model under attack whereas blackbox attacks do not. Whitebox attacks reduced model accuracies on fake images to 0% in all cases except ResNet with FGSM. Still, blackbox attacks resulted in less than 27% accuracy on perturbed fake images. Furthermore, CW-L_2 was more effective than FGSM in all cases except the blackbox attack on VGG. We suspect CW-L_2 overfits to the ResNet model in this case.

IV. Regularization as a Defense

A. Lipschitz Regularization

Lipschitz regularization, introduced in [5], constrains the gradient of the detector with respect to the input data. We achieve this by training the model using an augmented loss function involving the L_2 norms of the logit gradients:

$$J_{aug}(\mathbf{x}, \mathbf{y}, \theta) = J(\mathbf{x}, \mathbf{y}, \theta) + \frac{\lambda}{CN} \sum_{i=1}^{C} \|\nabla_x \mathbf{Z}(\mathbf{x})_i\|_2^2. \quad (7)$$

Here, we use $J_{aug}(\mathbf{x}, \mathbf{y}, \theta)$ to represent the augmented loss function and $J(\mathbf{x}, \mathbf{y}, \theta)$ to represent the training loss function before augmentation. $\mathbf{Z}(\mathbf{x})_i$ represents the pre-softmax scalar output (or logit) corresponding to class i for a multi-class neural network classifier. C is the total number of target classes and N is the dimensionality of the input vector. As before, $\mathbf{x}$ is the input vector, $\mathbf{y}$ is the corresponding true target class value and θ represents the model parameters. λ controls the strength of the regularization term in the augmented loss function.

Linearizing the (non-augmented) loss function provides some intuition into why this regularization can help:

$$\begin{aligned} J(\mathbf{x}_{adv}, \mathbf{y}, \theta) &\approx J(\mathbf{x}, \mathbf{y}, \theta) + \nabla_x J(\mathbf{x}, \mathbf{y}, \theta)^T (\mathbf{x}_{adv} - \mathbf{x}) \\ &= J(\mathbf{x}, \mathbf{y}, \theta) + \sum_{i=1}^{C} \frac{\partial J}{\partial \mathbf{Z}_i} \nabla_x \mathbf{Z}(\mathbf{x})_i^T (\mathbf{x}_{adv} - \mathbf{x}). \end{aligned} \quad (8)$$

As shown above, the linear approximation can be written in terms of the gradients of the detector logits with respect to the input. Then, we expect that minimizing the norm of these gradients will desensitize the loss from small perturbations, allowing the network to retain performance on inputs with adversarial perturbations. In the extreme case, if the norms of these gradients are zero, then the loss for the original unperturbed image equals the loss for the adversarially perturbed image (subject to the linear approximation).

B. Regularization Results

Lipschitz regularization improved the detection of adversarially perturbed deepfakes by ResNet models on average. We do not report regularization results for VGG given computational constraints and the slow nature of the CW-L_2 attack (around 2 minutes per image). We trained models with the following values for the regularization strength (λ): 5, 50, 500, and 5000. Table II shows the results for all λ values. Fig. 4 and our discussion below focus on results for the values on average and for the values that achieved the best results. Regularization did not affect the accuracy on the unperturbed test data: We observed 93.2% accuracy for both unregularized and regularized models on average (Table I).

In the blackbox case, unregularized models obtained an accuracy of 20.8% for FGSM and 4.6% for CW-L_2 on perturbed fake images. Regularized models improved detection of perturbed images to 31.5% for FGSM and 14.3% for CW-L_2 on average. In the best case, regularized models achieved an accuracy of 53.2% for FGSM and 19.8% for CW-L_2 on perturbed images.

Similarly, regularized models also performed better than unregularized models in the whitebox case. Unregularized models obtained an accuracy of 7.5% for FGSM and 0.0% for CW-L_2 on perturbed fake images, as reported in section III-D. On average, regularized models improved detection of perturbed images to 12.5% for FGSM and to 0.5% for CW-L_2. In the best case, regularized models achieved an accuracy of 26.5% for FGSM and 2.2% for CW-L_2 on perturbed images. Overall, although regularization slightly improved robustness to adversarial perturbations, the performance remains impractical for real world applications.

V. Deep Image Prior

Another approach for defending against adversarial attacks is to pre-process the input to remove perturbations before feeding it to the classifier. We do this by using an unsupervised technique called Deep Image Prior (DIP) which was originally introduced in [6] for image restoration purposes such as image denoising, inpainting and super resolution.

A. Image Restoration with DIP

This section summarizes the key ideas from the original DIP paper [6]. Let $\mathbf{x}_c$ be a corrupted image (e.g. a noisy image) and $\mathbf{x}$ be the ground truth uncorrupted image. Recovering $\mathbf{x}$ from $\mathbf{x}_c$ can be formulated as the following optimization problem:

$$\min_x \{E(\mathbf{x}, \mathbf{x}_c) + R(\mathbf{x})\}. \quad (9)$$

Here, $E(\mathbf{x}, \mathbf{x}_c)$ represents a domain-dependent “distance” or dissimilarity between $\mathbf{x}_c$ and $\mathbf{x}$. $R(\mathbf{x})$ is a regularization term that represents knowledge about ground truth images. The prior knowledge from regularization is critical since recovering $\mathbf{x}$ from $\mathbf{x}_c$ is generally an ill-posed problem.

We can replace $\mathbf{x}$ in (9) with a surjective function $\mathbf{g} : \theta \to \mathbf{x}$ and optimize over θ instead:

$$\min_\theta \{E(\mathbf{g}(\theta), \mathbf{x}_c) + R(\mathbf{g}(\theta))\}. \quad (10)$$

The DIP technique uses a generative CNN, $\mathbf{f}(\theta, \mathbf{z})$, with parameters θ and random seed $\mathbf{z}$ in place of $\mathbf{g}(\theta)$. Through experimentation, [6] shows that the architecture of a convolutional neural network itself encodes a prior that favors natural images over corrupted ones. This allows us to get a good reconstruction even if we ignore the regularization term R, leading to the following optimization problem:

$$\min_\theta \{E(\mathbf{f}(\theta, \mathbf{z}), \mathbf{x}_c)\}. \quad (11)$$

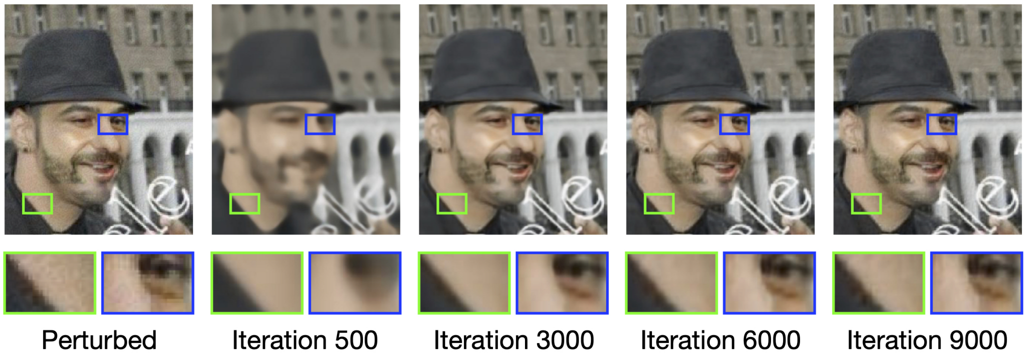


Fig. 5. Generated images during DIP optimization. These are images generated along the optimization path for a perturbed (FGSM) fake image. The image sharpness increases along with the number of optimization iterations. However, as the images gain detail, adversarial perturbations become more apparent. The generated image at iteration 9,000 is slightly noisier than the ones at iteration 3,000 and 6,000. Image format from [6]. *(Electronic zoom-in recommended).*

TABLE III. DIP Defense Results Overall

Classifier Threshold	Accuracy	AUROC	Fake Precision	 Recall	Real Precision	 Recall
0.50	97.0%	99.2%	96.8%	100%	100%	70.0%
0.25	97.0%	99.2%	98.9%	97.8%	81.8%	90.0%
Baseline	60.0%	41.9%	100.0%	55.5%	20.0%	100.0%

Note: DIP Results based on 100 images subsampled according to section V-C.

TABLE IV. DIP Defense Results by Category

Classifier Threshold	Attack	Unperturbed Fake-Correct	 Real-Correct	Blackbox: Perturbed Fake-Wrong	 Fake-Correct	Whitebox: Perturbed Fake-Wrong	 Fake-Correct
0.50	FGSM CW-L_2	100%	70%	100% 100%	100% 100%	100% 100%	100% 100%
0.25	FGSM CW-L_2	100%	90%	100% 100%	100% 100%	80% 100%	100% 100%
Baseline	FGSM CW-L_2	100%	100%	0% 0%	100% 100%	0% 0%	100% 100%

Note: DIP Results based on 100 images subsampled according to section V-C.

In practice, if the network is optimized for too long, it learns to generate the corruptions. However, the network learns to generate "natural" features before learning to generate the corruptions due to the prior that the CNN encodes. In other words, a good image reconstruction tends to exist somewhere along the optimization trajectory.

B. Eliminating Adversarial Perturbations with DIP

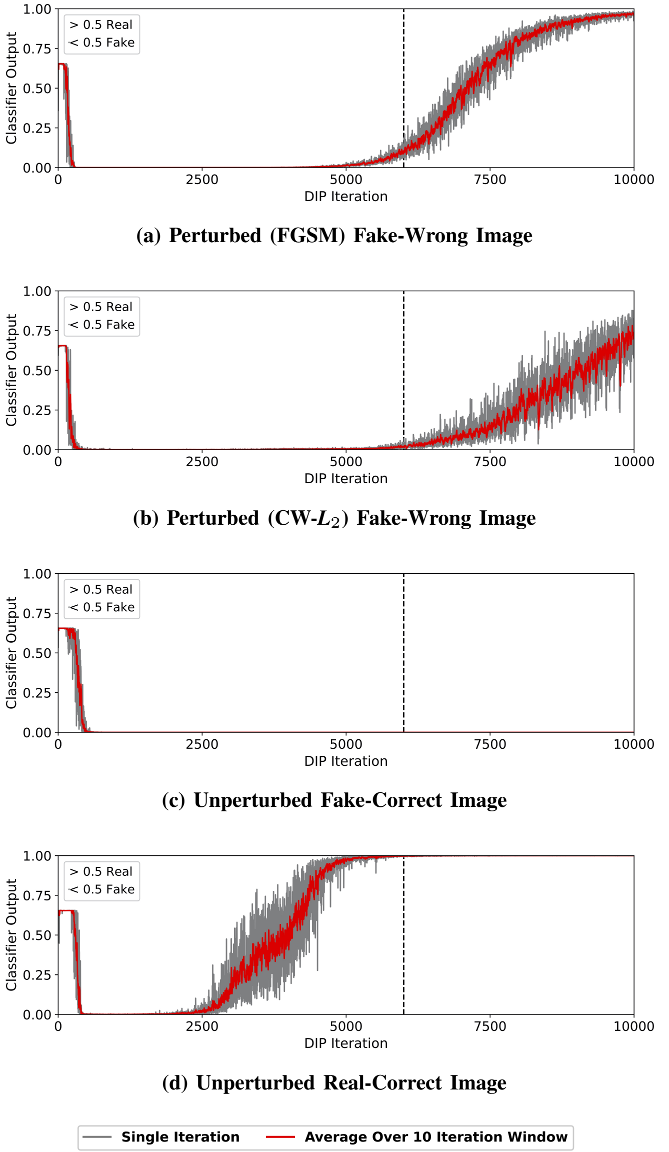
We can use the image restoration framework described above to remove adversarial perturbations from adversarial examples. We simply replace $\mathbf{x}_c$ with $\mathbf{x}_{adv}$ in (11). We chose Mean Squared Error (MSE) calculated pixel-wise over the images as our dissimilarity metric, E. This metric was chosen since it was effective in [6] for various applications including image denoising, super resolution and JPEG compression artifact removal. Thus, we modify the DIP optimization in (11) to remove adversarial perturbations as follows:

$$\min_{\theta}\{\mathrm{MSE}(\mathbf{f}(\theta, \mathbf{z}), \mathbf{x}_{adv})\}. \tag{12}$$

We propose the following deepfake defense using (12). Given that an unperturbed image tends to occur somewhere along the DIP optimization trajectory, we feed the generated image at an intermediate iteration into an existing classifier. The classifier output for the generated image at the intermediate iteration is then used to make a final classification for the image. Throughout section V, the "classification of the DIP defense" refers to the final classification made using this process, and "classifier" refers to the CNN model used to obtain the classification.

We used only the ResNet model for DIP due to computational constraints as described in section V-C. We also trained the classifier for an additional 10 epochs on the training dataset. For these 10 epochs, the training dataset was augmented so that approximately 40% of it contained blurry images. This was done because the reconstructed DIP images without the perturbations tended to be slightly less sharp compared to the original images in the training and test sets.

We created the blurry images by preprocessing training images using a Gaussian blur kernel with σ values selected uniformly from the range [3.0, 5.0]. Table I lists the performance metrics of the ResNet model trained on the augmented dataset. Table II reports the adversarial attack results for this model, which are similar to the results for the model without data augmentation.



(a) Perturbed (FGSM) Fake-Wrong Image

(b) Perturbed (CW-L_2) Fake-Wrong Image

(c) Unperturbed Fake-Correct Image

(d) Unperturbed Real-Correct Image

Fig. 6. DIP optimization graphs. This figure plots the ResNet classifier softmax output corresponding to the real class for the generated images along the DIP optimization path for four cases. The grey curves plot the output at every iteration, while the red curves plot the average of the outputs over a window size of 10 centered at each iteration. We use 0.5 as our classification threshold for this figure. (a) and (b) correspond to fake images perturbed in a whitebox setting using FGSM and CW-L_2 respectively. (c) and (d) correspond to unperturbed fake and real images respectively. In each case, the model classifies the generated image correctly at iteration 6,000.

Fig. 5 shows a perturbed (FGSM) fake image being reconstructed using the DIP framework. This image was chosen such that the ResNet model classifies it as real. We observe that as the number of DIP optimization iterations increases, the images gain more detail. But as the image sharpness increases, the generated images also tend to include adversarial perturbations: The generated image at iteration 9,000 is slightly noiser than the ones at iterations 3,000 and 6,000.

Fig. 6a shows the classifier output for the perturbed (FGSM) fake image in Fig. 5 along the DIP optimization path. We ignore the predictions for the first 500 iterations where the generative CNN is still learning how to produce a natural-looking image. We observe that, after this, the classifier output remains flat and close to 0 (fake) until around iteration 5,000. Following this, the classifier output increases as the generated image begins including the perturbations until it flattens out at 1 (real). Fig. 6b shows a similar pattern for a perturbed fake (CW-L_2) image. In contrast, for an unperturbed fake image (Fig. 6c), the graph flattens out at a fake prediction and never reaches a real prediction. For a real unperturbed image (Fig. 6d), the graph reaches a real prediction much earlier than in the perturbed fake cases. In each case, the classifier predicts the correct class at iteration 6,000.

C. DIP Experiments

We performed the DIP optimization for 10,000 iterations on a total of 100 images based on the test set. Iteration 6,000 was used to obtain the classification of the DIP defense after evaluating iterations in the range of 2,500 to 7,500. We used a U-Net architecture [22] for the generative CNN, $\mathbf{f}(\theta, \mathbf{z})$, in the DIP optimization described in (12). This architecture was used because it was shown to be effective in [6] for both denoising and removing JPEG compression artifacts from images. For the exact architecture details, we refer to our code repository linked at the end of this paper. The optimization process is slow and took approximately 30 minutes for each image using a NVIDIA Tesla K80 GPU on Google Colab. For this reason, we chose only 100 images for the experiments.

We randomly sampled 10 images each from the following 2 categories for both perturbed FGSM and CW-L_2 images in blackbox and whitebox settings (80 images total):

- *Perturbed Fake-Wrong*: A perturbed fake image that the classifier predicts as real.
- *Perturbed Fake-Correct*: A perturbed fake image that the classifier predicts as fake.

In addition, we sampled 10 images each from the following 2 categories for unperturbed images (20 images total):

- *Unperturbed Fake-Correct*: An unperturbed fake image that the classifier predicts as fake.
- *Unperturbed Real-Correct*: An unperturbed real image that the classifier predicts as real.

All images were sampled such that the ResNet model (trained on the augmented dataset) obtained a correct prediction on the unperturbed versions of the images.

D. DIP Results

We report DIP results using classification thresholds of 0.5 and 0.25. Table III reports the overall performance of the DIP defense across all 100 images while Table IV includes the accuracy of the DIP defense for each category of images. The tables also include a baseline performance from the classifier without the DIP defense.

The DIP defense achieved 95% on perturbed deepfakes that fooled the original detector (Perturbed Fake-Wrong), while retaining 98% accuracy in other cases (Real-Correct and Fake-Correct) with a 0.25 threshold. Overall, on the 100 image subsample, the defense obtained 97.0% accuracy and 99.2% AUROC for both classification thresholds. Varying the threshold reveals the tradeoff between incorrectly predicting real images as fake (false positives) and fake images as real (false negatives). For deepfake detection, false positives are generally less of a problem than false negatives.

Using a threshold of 0.5 yielded 100% recall for both unperturbed and perturbed fake images. But this threshold resulted in only 70% recall for real images. On the other hand, using a threshold of 0.25 improved the recall for real images to 90%, but also reduced recall for fake images to 97.8%. Specifically, the accuracy in the whitebox perturbed (FGSM) fake-wrong category decreased from 100% to 80%.

VI. Discussion and Limitations

Our results demonstrate that adversarial perturbations can enhance deepfakes, making them significantly more difficult to detect. Lipschitz regularization made the CNNs more robust to adversarial perturbations in general. However, the performance boost from regularization alone may not be enough for practical use in deepfake detection. This was especially true in the whitebox CW-L_2 setting where even the regularized model only classified 2.2% of the perturbed fake images correctly. The DIP defense shows more promising results. It achieved a recall of 97.8% for perturbed and unperturbed fake images using a classification threshold of 0.25 (Table III). Furthermore, the DIP defense retained at least 90.0% of the classifier's performance on real images using the same threshold value.

While the DIP defense showed success for deepfake detection on the 100 images tested, we emphasize that additional experiments would be required to demonstrate success on adversarial attacks in other domains. For example, deepfake classifiers only need to be robust to adversarial perturbations for one class of images (the fake class), while in other domains, robustness to adversarial attacks on more than one class may be important. Another limitation of the DIP defense is the time it takes to process a single image. As described in section V-C, each image took a little under 30 minutes to process on a NVIDIA Tesla K80 GPU. This may limit the practicality of the defense in situations where there are resource constraints or where many images need to be processed in real time. Future work involves finding more efficient methods for improving deepfake detector robustness to adversarial perturbations.

Acknowledgment

The authors sincerely thank Professor Bart Kosko for his feedback and guidance throughout this work. We also acknowledge the USC Center for AI in Society's Student Branch for providing us with computing resources.

Code Availability

Code and additional architecture details are available at: *https://github.com/ApGa/adversarial_deepfakes*.

References

[1] T. T. Nguyen, C. M. Nguyen, D. T. Nguyen, D. T. Nguyen, and S. Nahavandi, "Deep learning for deepfakes creation and detection," *arXiv: 1909.11573*, 2019.

[2] J. Vincent, "New ai deepfake app creates nude images of women," 2019. [Online]. Available: https://www.theverge.com/2019/6/27/18760896/deepfake-nude-ai-app-women-deepnude-non-consensual-pornography

[3] D. O'Sullivan, "Now fake facebook accounts are using fake faces," 2019. [Online]. Available: https://edition.cnn.com/2019/12/20/tech/facebook-fake-faces/index.html

[4] K. Wagner, "Twitter will label, remove 'deepfake' videos under new policy," 2020. [Online]. Available: https://fortune.com/2020/02/04/twitter-deepfake-videos/

[5] W. Woods, J. Chen, and C. Teuscher, "Adversarial explanations for understanding image classification decisions and improved neural network robustness," *Nature Machine Intelligence*, 2019.

[6] V. Lempitsky, A. Vedaldi, and D. Ulyanov, "Deep image prior," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[7] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *IEEE International Conference on Computer Vision*, 2015.

[8] Shaoanlu, "Fewshot Face Translation GAN," 2019. [Online]. Available: https://github.com/shaoanlu/fewshot-face-translation-GAN

[9] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz, "Few-shot unsupervised image-to-image translation," in *IEEE International Conference on Computer Vision*, 2019.

[10] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[13] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *10th ACM Workshop on Artificial Intelligence and Security*, 2017.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv: 1412.6572*, 2014.

[15] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy*, 2017.

[16] N. Carlini and H. Farid, "Evading deepfake-image detectors with white- and black-box attacks," *arXiv: 2004.00622*, 2020.

[17] P. Neekhara, S. Hussain, M. Jere, F. Koushanfar, and J. McAuley, "Adversarial deepfakes: Evaluating vulnerability of deepfake detectors to adversarial examples," *arXiv: 2002.12749*, 2020.

[18] Kkew3, "Pytorch implementation of carlini-wagner's l2 attack." 2019. [Online]. Available: https://github.com/kkew3/pytorch-cw2

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[20] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv: 1605.07277*, 2016.

[21] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv: 1902.06705*, 2019.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015.