Adversarial Attacks on Co-Occurrence Features for GAN Detection

Michael Goebel
UC Santa Barbara
Santa Barbara, California
mgoebel@ucsb.edu

B.S. Manjunath
UC Santa Barbara
Santa Barbara, California
manj@ucsb.edu

Abstract

Improvements in Generative Adversarial Networks (GANs) have greatly reduced the difficulty of producing new, photo-realistic images with unique semantic meaning. With this rise in ability to generate fake images comes demand to detect them. While numerous methods have been developed for this task, the majority of them remain vulnerable to adversarial attacks. In this paper, develop two novel adversarial attacks on co-occurrence based GAN detectors. These are the first attacks to be presented against such a detector. We show that our method can reduce accuracy from over 98% to less than 4%, with no knowledge of the deep learning model or weights. Furthermore, accuracy can be reduced to 0% with full knowledge of the deep learning model details.

1. Introduction

Since the advent of Generative Adversarial Networks (GANs) in 2014 [8], there has been a dramatic increase in the capabilities of GAN networks. They are now able to modify facial attributes in photo-realistic 1 megapixel images [11], perform image-to-image translation [32], and create new images given only a segmentation map [21].

With the vast increase in capability of GANs, there have been a number of detectors developed to distinguish GAN from authentic images [19, 27, 30, 1, 18]. These detectors generally perform well, reporting over 98% test set accuracy. However, the question then arises as to what adversarial methods can be used to fool these detectors.

We focus primarily on attacking the co-occurrence feature based GAN detector proposed by Nataraj et al. [19]. Our paper makes the following contributions:

1. Describes the first targeted attacks against co-occurrence based GAN detectors known to the authors.
2. Generalizes the discrete co-occurrence matrices used for GAN detection to a differentiable function, and shows that gradient descent based attacks on a co-occurrence based detector with known model weights can be used to bring detector accuracy down to 0%.
3. Creates a new attack gray-box against co-occurrence based detectors, which assumes no knowledge of the deep learning weights or architecture. Tests with this method show that it can drop accuracy on GAN images from 98% to less than 4%.
4. Demonstrate that without adversarial training against our method, it can decrease accuracy of other GAN detection methods by approximately 18%.

2. Related Works

2.1. Co-Occurrence Matrices in Steganography

Co-occurrence matrices have a long history in image forensics, as features for detecting steganography, along with works intended to counter such detectors [25, 24]. Though steganography presents a unique challenge in that an adversarial image still must contain the embedded message, changes are often to a small number of pixels, and to a
small degree, making restoring the statistics an easier task. In contrast, GAN manipulations are more pronounced, significantly changing the image’s pixels and semantics.

2.2. GAN Detection Methods

Many methods of GAN detection have been proposed, though the one of primary interest will be that used in Nataraj et al. They propose using co-occurrence matrices as a feature, which will be passed to a binary deep learning classifier. Another variation of the co-occurrence detector was recently posted on arXiv, which includes cross-channel co-occurrence pairs in the feature \[11\]. Abridged results generalizing our attack to this method are in section 6.5.2.

Several methods were proposed in an older work from F. Marra et al. [18]. These included constraining the input layer of a neural network classifier to a high-pass filter, computing co-occurrence matrices on high-pass filtered images, and reusing the original GAN discriminator. A more recent method proposed that simply training a ResNet model directly on a real vs GAN dataset was effective on more recent method proposed that simply training a ResNet images, and reusing the original GAN discriminator. A ter, computing co-occurrence matrices on high-pass filtered put layer of a neural network classifier to a high-pass fil-

3. Detection Methods

In addition to the co-occurrence features, we also investigate the co-occurrence and direct methods. Figure \[2\] shows an abstract outline of these models. This section describes the 3 methods we investigate for the feature extraction step.

3.1. Co-Occurrence

Unless stated otherwise, we will use the horizontal co-occurrence matrix defined by Nataraj et al. For each channel in the RGB input image, represented by the array \(X\), we produce a 2D histogram of horizontal pixel pairs:

\[
C_{i,j} = \sum_{k,l} \delta(X_{k,l} - i) \cdot \delta(X_{k,l+1} - j) \quad (1)
\]

Where \(\delta(\cdot)\) is the Kronecker delta function:

\[
\delta(k) = \begin{cases} 1, & \text{if } k = 0 \\ 0, & \text{otherwise} \end{cases}, k \in \mathbb{Z} \quad (2)
\]

Each co-occurrence matrix is then scaled into the range \([0, 1]\), re-stacked in the channel dimension, and passed to a deep learning classifier.

3.2. DFT

As proposed by Zhang et al., we will use the DFT of the image as an input to a deep learning classifier. The processing steps for the DFT based method are as follows:

1. Get the centered, unitary DFT of the input image

![Figure 2. High-level diagram of the detection architectures. Sections 3.1, 3.2, and 3.3 describe the different feature extraction methods investigated. Section 6.2 describes the different CNN architectures tested.](image)
2. Take the magnitude of the DFT
3. Apply the function \( f(x) = \log(x + 10^{-6}) \)
4. Shift and scale into the range [-1,1]

All steps are consistent with those used in X. Zhang et al., except for the inclusion of a small constant in step 3, to prevent a \( \log(0) \) case. They use a ResNet34 architecture pretrained with ImageNet weights for detection.

3.3. Direct

This method will pass the image directly to a deep learning classifier, with only affine scaling as a preprocessing step. The following ImageNet means and standard deviations are applied, as per the Torchvision documentation [17]:

\[
\begin{align*}
\text{mean} &= 255 \cdot [0.485, 0.456, 0.406], \\
\text{std} &= 255 \cdot [0.229, 0.224, 0.225]
\end{align*}
\]

(3) (4)

For their direct method, Wang et al. used ResNet50 pretrained on ImageNet [27].

4. Co-Occurrence Gray-Box Attack

4.1. Attack Formulation

For our gray-box attack, we assume that it is known that co-occurrence matrices are the only feature used for detection, but we have no knowledge about the deep learning model used on these matrices. We also assume that the adversary has an arbitrary set of real images at their disposal. The goal of the adversary will be to modify each GAN image by some small amount, such that the co-occurrence matrix of the adversarial image is close to, if not exactly equal to, the co-occurrence matrix of a real image.

The adversarial, real, and GAN images will be represented by \( X_A, X_R, \) and \( X_G \) respectively. \( F(\cdot) \) is the co-occurrence function, \( \text{Loss}_1(\cdot, \cdot) \) and \( \text{Loss}_2(\cdot, \cdot) \) are the loss functions to be defined later, and \( \lambda \) is a user-defined constant. The adversarial image will be proposed as:

\[
X_A = \arg \min_{\hat{X}} \text{Loss}_1(F(\hat{X}), F(X_R)) + \lambda \text{Loss}_2(\hat{X}, X_G)
\]

(5)

4.2. Distinctions Between Co-Occurrence and Histograms

Given the formulation in [3] it is worth noting that the Earth Mover’s Distance (EMD) solves a similar optimization problem, and that there exist efficient approximations [22]. For a pair of 2 dimensional histograms and for some pre-defined cost function between bins, this would produce a minimal cost transformation from one histogram to the other. However, non-edge pixels will appear twice in the co-occurrence histogram, once as the left pixel in the pair, and again as the right. Therefore, entries in the co-occurrence matrix cannot be individually manipulated to achieve this optimal transport, without inadvertently changing values at another location in the histogram.

To handle this entanglement between pairs, we use gradient descent to find an approximate minima. However, this will require the functions \( F, \text{Loss}_1, \) and \( \text{Loss}_2 \) to be differentiable. The original definition of the co-occurrence function was over only integer inputs, posing a problem for the differentiability requirement. The next few sections break down the details of how \( F, \text{Loss}_1, \) and \( \text{Loss}_2 \) are selected.

4.3. Differentiable Extension of Co-Occurrence Function

In creating a differentiable extension of the co-occurrence matrix, we impose the following requirements:

1. For integer inputs, \( F(\cdot) \) must be equivalent to the original co-occurrence function.
2. The sum of the histogram bins should equal the number of input elements.
3. For all input elements, the contribution to each bin must be non-increasing with respect to distance from that bin.
4. It must be differentiable over \( \mathbb{R}^2_{[0,255]} \).

Given the original co-occurrence formulation in equation [1] a simple extension would be to define a new one-dimensional function \( f(\cdot) \) which will interpolate the delta function’s integer values:

\[
C_{i,j} = \sum_{k,l} f(X_{k,l} - i) \cdot f(X_{k,l+1} - j)
\]

(6)

Equation [6] consists only of additions, multiplications, and \( f(\cdot) \), making gradient calculation straightforward. From this equation, the previous 4 requirements can be simplified into these requirements on \( f(\cdot) \):

1. \( f(x) = \delta(x), \ x \in \mathbb{Z} \)
2. \( f(x) = 1 - f(x - 1), \ \forall x \in [0,1] \)
3. \( \frac{df}{dx} \leq 0 \) for \( x > 0 \), and \( \frac{df}{dx} \geq 0 \) for \( x < 0 \)
4. \( \frac{df}{dx} \) should be defined for all \( x \in \mathbb{R}_{[-255,255]} \)

The combination of constraints 1 and 3 will require that \( f(x) = 0 \) for \( x \notin (-1,1) \). Therefore, each pixel pair will contribute to at most 4 bins. This fact was taken advantage of in implementation, as opposed to computing the entire
Both the triangle and raise cosine shown below were tested as interpolation functions:

\[
tri(x) = \begin{cases} 
1 - |x|, & \text{if } |x| < 1 \\
0, & \text{otherwise}
\end{cases}
\]  

(7)

\[
\text{raised}_\cos(x) = \begin{cases} 
\frac{1+\cos(\pi x)}{2}, & \text{if } |x| < 1 \\
0, & \text{otherwise}
\end{cases}
\]  

(8)

For the triangle function, derivatives at \(x = -1, 0, 1\) are undefined, so the average of the left and right derivatives is used. Raise cosine gave better results experimentally, and will be used for the remained of the tests.

4.4. Co-Occurrence Loss Function

In this section, we provide intuitive reasoning and experimental justification for our selection of \(\text{Loss}_1\). We provide several motivating examples for the 1D and 2D histogram cases. "Source" will correspond to the GAN input, "target" to the real input, and "solution" to the adversarial solution. For all of these examples, we will assume \(\lambda = 0\).

4.4.1 One-Dimensional Example

Consider the case with a source of \([1, 2, 3]\), a target of \([2, 3, 4]\), and \(\lambda = 0\). We would now like gradient descent to push the source towards the target, making their histograms equal. In this example, all derivatives should be negative.

Shown in Figure 3 are plots of loss for different loss functions, varying one input at a time. Consider the loss values as \(x_1\) is moved from 1 to 4. For \(L_1\), there is a constant loss from \(x_1 = 1\) to \(3\). This is compared to \(L_2\), where loss fluctuates in the same region. Given that the histogram will have a constant \(L_1\) norm, and that \(L_2\) loss is less for the vector \([1/2, 1/2]\) than \([0, 1]\), the \(L_2\) loss function tended to get stuck between integer values. For this reason, we will focus our attention on \(L_1\) loss.

Looking at the top left graph in Figure 3, the loss for \(x_1\) decreases only after passing the threshold of 3. Using point-wise loss, the vacancy at 4 can only pull values which are within the \((-1, 1)\) support region of \(f(\cdot)\). To alleviate this, we instead compute loss on a multiscale pyramid of the histograms, with a downsampling factor of 2 in each step.

To combine the multi-scale losses, a simple weighted sum is used. Weights are set equal to the downsampling factor of 2 in each step. Results using the image pyramid loss are shown in the right column of Figure 3. When gradient descent is run on the different loss functions for the 1D case, the results in Figure 4 are produced.

4.4.2 Two-Dimensional Examples

A 2D example is shown in figure 5. Especially important from this figure is the necessity of random noise. Often we will need points initialized to the same value to arbitrarily split into two different outcomes, which cannot occur with deterministic gradient descent.

This two-dimensional histogram test was repeated over 100 iterations, with both source and target vectors containing 8 elements uniformly sampled from \(Z^2_{[0,7]}\). For the \(L_1\) pyramid with Gaussian noise, all 100 tests successfully converged from the source to target. This is compared to only
Figure 5. Example applying point-wise and pyramid loss to a 2D histogram gradient descent problem. Horizontal and vertical axes represent location of the 2D vectors. Top left: with $L_1$ loss and no noise, all gradients are 0. Top right: With noise, $L_1$ finds a sub-optimal minima, where not all target points are reached. Bottom left: Without added noise, the pyramid loss almost converges to a global minima. However, the two source points originating from (1,1) need to split and fill different target points. With deterministic gradient descent, this splitting will not occur. Bottom right: A proper solution is found with pyramid loss and noise.

4.4.3 Extension to Co-Occurrence on Images

For an 8-bit image, a 9 layer pyramid is used, with down-sampling factors from 1 (None) to 256. To implement the blurring and downsampling steps in the co-occurrence image pyramid, we rely upon the original interpolation function defined for the co-occurrence. By dividing the input image by the downsampling factor before computing co-occurrence, lower resolution co-occurrence matrices can be produced. The full loss function is shown in equation 9.

$$\text{loss} = \sum_{n=0}^{8} 2^n \left\| F\left( \frac{X_A}{2^n} \right) - F\left( \frac{X_R}{2^n} \right) \right\|_1$$

4.5. Image-Space Loss Function

In equation 5 only the $\text{loss}_2$ and $\lambda$ terms are left to be defined. For consistency with the co-occurrence loss, $L_1$ distance is chosen for the image-space loss. The $\lambda$ parameter remains as a user selected parameter, and several values were tested experimentally.

4.6. Implementation Details

Ideally, we would like to choose source-target pairs with similar color values for optimization. For example, we would not want to force a GAN image with a green grassy background to have the same color distribution as a real image of a blue ocean. To do this, we divide the data into blocks of size approximately 900, and for each GAN image, select the real image whose EMD over the 1D RGB histograms is closest to that of the GAN image.

With the pairs selected, we can then run our gradient descent algorithm. The solution is initialized with the source image. We use a standard gradient descent, with a learning rate of 0.01, and momentum of 0.9. This is done in 3 sequential epochs, with 200, 50, and 50 steps. For the first 2 epochs, Gaussian noise with standard deviation of 0.01 is added to the image. No noise is added in the last epoch. The solution is rounded after each epoch.

When run on an Nvidia 1080 Ti, the algorithm took approximately 30 seconds per 256x256 image. However, up to 3 processes could be placed on a single GPU, so 6 adversarial images could be produced every minute.

Quantitative results are shown in figure 8. For comparison, average $L_1$ loss between the co-occurrence matrices of the source target pairs was 0.90, and $L_1$ loss between source and target images was 52.7. For this real data, it cannot achieve a perfect match between the real and adversarial co-occurrence matrices. Two examples are given in figures 6 and 7. The effect of this slight mismatch is investigated experimentally.
5. Other Adversarial Attacks

5.1. Gray-Box DFT

This method follows a similar formulation to the co-occurrence gray-box attack. A real image is obtained in addition to the GAN image, and the detection feature of the adversarial image is made to be similar to the real image, while minimizing the distance from the original GAN image. We rely upon the intuition that the defining features of the GAN in the DFT domain are concentrated away from the DC axes. To estimate this high-frequency noise signal, we use the same filter as Kirchner in his work on resampling [13], and has a centered DFT given in equation 10:

\[
\mathcal{F}(f) = \frac{1}{4} \begin{bmatrix} 3 & 0 & 3 \\ 0 & 0 & 0 \\ 3 & 0 & 3 \end{bmatrix} \tag{10}
\]

To produce an adversarial image, we solve the following:

\[
X_A = \arg \min_{\tilde{X}} \left\| f \ast \tilde{X} - f \ast X_R \right\|_2^2 + \lambda^2 \left\| \tilde{X} - X_G \right\|_2^2 \tag{11}
\]

Application of the Fourier transform turns this problem into one of weighted least squares, and can be solved as:

\[
\mathcal{F}(X_A) = \frac{\mathcal{F}(f)^2 \cdot \mathcal{F}(X_R) + \lambda^2 \mathcal{F}(X_G)}{\mathcal{F}(f)^2 + \lambda^2} \tag{12}
\]

This is done between randomly selected real and GAN images, for different \( \lambda \) values in section 6.

5.2. White-Box PGD

With the co-occurrence, DFT, and direct methods all being differentiable pytorch functions, we run the \( L_{\infty} \) PGD algorithm on all GAN images on each method [16]. This is done using the advertorch library [6]. Default parameters are used, with a maximum distortion of 1, maximum step size of 2/40, and 40 total iterations, running on pixels in the range \([0, 255]\). Pixels are rounded after completion of PGD.

6. Experiments

6.1. Datasets

Our dataset consists of 4 different GANs, drawing from a variety of image datasets and tasks. Image counts are given in table 1. The each group is divided into a 70/15/15 train/val/test split. These groups are then further split in half; the first for training and testing of models, and the second for generating adversarial samples. All images were center-cropped to 256x256.

| Architecture | Dataset                      | Total Count |
|--------------|------------------------------|-------------|
| CycleGAN [32]| apple2orange [5]             | 3,000       |
|              | horse2zebra [5]              | 3,000       |
|              | summer2winter [32]           | 3,000       |
|              | cityscapes [4]               | 3,000       |
|              | cezanne [32]                 | 3,000       |
|              | monet [32]                   | 3,000       |
|              | ukyoe [32]                   | 3,000       |
|              | vangogh [32]                 | 3,000       |
| ProGAN [10]  | CelebHQ [15]                 | 24,000      |
| SPADE [21]   | ADE20K [31]                  | 12,000      |
|              | COCO-Stuff [2]               | 12,000      |
| StyleGAN [11]| LSUN Bedroom [29]            | 8,000       |
|              | LSUN Cat [29]                | 8,000       |
|              | LSUN Cat [29]                | 8,000       |
| Total        |                              | 96,000      |

Table 1. Combined number of real and fake samples from each data subset. For all subsets, the number of real and fake examples is equal. In all, there were 48k real and 48k GAN images used.

\[
X_A = \arg \min_{\tilde{X}} \left\| f \ast \tilde{X} - f \ast X_R \right\|_2^2 + \lambda^2 \left\| \tilde{X} - X_G \right\|_2^2 \tag{11}
\]
### Table 2. Overall accuracy of different networks on a balanced test set of real and GAN. Each row represents using a different deep learning architecture. Three detection methods are shown as the first column headers. The second shows results with either ImageNet weights or random initialization.

| Method | Co-Occurrence | DFT | Direct |
|--------|---------------|-----|--------|
| Initialization | ImNet | rand | ImNet | rand | ImNet | rand |
| ResNet18 | 0.979 | 0.977 | 0.904 | 0.888 | 0.980 | 0.829 |
| ResNet50 | 0.979 | 0.977 | 0.864 | 0.900 | 0.976 | 0.866 |
| ResNet101 | 0.424 | 0.569 | 0.503 | 0.500 | 0.519 | 0.503 |
| ResNet152 | 0.500 | 0.668 | 0.495 | 0.500 | 0.500 | 0.499 |
| ResNetX105 | 0.978 | 0.935 | 0.882 | 0.907 | 0.986 | 0.853 |
| InceptionV3 | 0.944 | 0.950 | 0.948 | 0.708 | 0.990 | 0.949 |
| MobileNet | 0.978 | 0.974 | 0.949 | 0.919 | 0.996 | 0.989 |

Table 3. Test set accuracy of co-occurrence based detectors, without adversarial retraining. Gray-box (GB) co-occurrence (CO) samples are generated as described in section 4. PGD co-occurrence examples are produced using ResNet18.

| Method | Real | GAN | GB CO \( \lambda = 0 \) | PGD CO |
|--------|-----|-----|----------------|--------|
| ResNet18 | 0.979 | 0.984 | 0.030 | 0.000 |
| MobileNet | 0.976 | 0.981 | 0.039 | 0.083 |

### Table 4. Results for only the ResNet18 co-occurrence detector. The rows show results with and without adversarial retraining.

16 real and 16 GAN images per batch, for 16 epochs. An Adam optimizer was used, with default parameters of 0.001 for the learning rate, and (0.9,0.999) for the betas [12]. The final model weights for testing are selected from the epoch number on which validation loss was the lowest.

The ImageNet pretrained networks did better than those with random initializations in almost all cases. As the larger networks did not provide noticeable improvements on this dataset, we will use pretrained ResNet18 and pretrained MobileNet for the remainder of tests. We chose ResNet18 to generate the PGD samples, given that ResNets were used by both Zhang et al. and Wang et al.

### 6.3. Testing on Adversarial Samples

We then evaluated the detectors chosen in the previous section on the co-occurrence adversarial examples, with results shown in Table 3. For both detectors, the gray-box co-occurrence attack drops the GAN detection rate from approximately 98% to less than 4%, with no knowledge of the deep-learning model used. As expected with the PGD attack, accuracy on the exact model being attacked drops to 0. However, accuracy on MobileNet drops to only 8%; more than twice what was achieved with the gray-box attack.

### 6.4. Adversarial Training

Next we adversarially trained the same networks using different subsets of the adversarial samples. The labels remain binary, with real images in one class, and all GAN images, including adversarial GAN images, in the other.

For data balancing, we maintain an equal number of positive and negative samples. Within the positive sample class, each of the sub-types is sampled equally. For example, in the test using all gray-box co-occurrence adversarial examples, we used 16 real, 4 GAN, and 4 from each of the 3 gray-box co-occurrence classes. For the set of all adversarial images, a batch size of 40 is used so the batch can be evenly divided. For all other cases, batch size remains 32.

### 6.4.1 Full Results

All results are shown in table 5 with a summary of just the co-occurrence results in table 4. These results show test accuracy on only one group at a time. When comparing across rows, it is important to note any changes in accuracy on real images in addition to changes in GAN performance.

From this table, we summarize the results as follows:

1. Without adversarial retraining, all adversarial attacks would generally decrease performance on all detectors.
2. Adversarially training on one attack method does not generally improve performance against other methods.
3. After adversarial training, the models which are most different from the assumption in the adversarial attacks performed best. Notably, MobileNet trained on all adversarial images got over 98% on all subsets.

Of particular relevance to this paper is that for all co-occurrence based detectors which were not trained on the gray-box co-occurrence based attack, accuracy was less than 5% for \( \lambda = 0 \). This included models which were trained on all other adversarial attacks. This would seem to indicate that most co-occurrence detectors not trained on this particular attack would remain highly vulnerable. After retraining, accuracy on the \( \lambda = 0 \) class was only slightly lower than regular GAN class. The difference was more significant in the MobileNet co-occurrence detector.

Also noteworthy is that the DFT and direct classifiers performed, on average, 18% worse on the GB-CO \( \lambda = 0 \) set than the original GAN images, if they were not trained against the GB-CO attack. Performance generally improved back to baseline levels after retraining.

### 6.5. Other Tests

#### 6.5.1 Reversed Gray-Box Co-Occurrence Attack

Though less useful as a real-world attack, the GB-CO method can also be used to generate real images which will be classified as GAN. With the target and source switched, we produced a test set of adversarial real images, and tested on the regular ResNet18 co-occurrence detector (row 1 in Table 5). 95.9% of the images were misclassified as GAN.
6.5.2 Cross Channel Co-Occurrence Detector

Recently a paper was posted on arXiv from M. Barni et al. claiming to have improved the original co-occurrence GAN detector by including cross-channel co-occurrence matrices [1]. Their cross-band co-occurrence matrices for a red-green pair are defined in equation [33] assuming HWC convention on X. This is repeated for the red-blue and green-blue pairs. For spatial co-occurrence, they instead use diagonal pairs, shown in equation [14]. After producing these 6 co-occurrence matrices, they are stacked in the channel dimension, and passed to a ResNet18 classifier.

\[
C_{i,j} = \sum_{k,l} \delta(X_{k,l,1} - i) \cdot \delta(X_{k,l,2} - j) \tag{13}
\]

\[
C_{i,j} = \sum_{k,l} \delta(X_{k,l} - i) \cdot \delta(X_{k+1,l+1} - j) \tag{14}
\]

We also modified the co-occurrence gray-box attack equation in [6] to accept the 6 different pairs used in M. Barni et al. We ran the algorithm to produce an adversarial test set for the cross-band co-occurrence detector. Results are shown in table 5. The gray-box attack still seems effective against detectors using this other co-occurrence feature.

7. Conclusion

In this paper, we presented two new attacks against co-occurrence based GAN detectors. We also demonstrate pre-

8. Acknowledgements

The real and GAN images used in this work were collected or generated from public sources during the first author’s internship at Mayachitra Inc.

This work was partially supported by NSF SI2-SSI award #1664172 and NSF HDR IDEAS2 award #1934641. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
References

[1] Mauro Barni, Kassem Kallas, Ehsan Nowroozi, and Benedetta Tondi. Cnn detection of gan-generated face images based on cross-band co-occurrences analysis. arXiv preprint arXiv:2007.12909, 2020.

[2] Holger Caesar, Japser Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1209–1218, 2018.

[3] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy (sp), pages 39–57. IEEE, 2017.

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3213–3223, 2016.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[6] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6572, 2014.

[7] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4401–4410, 2019.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[9] Matthias Kirchner. Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue. In Proceedings of the 10th ACM workshop on Multimedia and security, pages 11–20, 2008.

[10] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236, 2016.

[11] 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.

[12] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083, 2017.

[13] Sébastien Marcel and Yann Rodriguez. Torchvision: the machine-vision package of torch. In Proceedings of the 18th ACM international conference on Multimedia, pages 1485–1488, 2010.

[14] F. Marra, D. Gragnaniello, D. Cozzolino, and L. Verdone. Detection of gan-generated fake images over social networks. In 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), pages 384–389, 2018.

[15] Lakshmanan Nataraj, Tajuddin Manhar Mohammed, BS Manjunath, Shivkumar Chandrasekaran, Arjuna Fenner, Jawadul H Bappy, and Amit K Roy-Chowdhury. Detecting gan generated fake images using co-occurrence matrices. Electronic Imaging, 2019(5):32–1, 2019.

[16] J Neves, Ruben Tolosana, Ruben Vera-Rodriguez, Vasco Lopes, Hugo Proença, and Julian Fierrez. Ganprint: Improved fakes and evaluation of the state-of-the-art in face manipulation detection. arXiv preprint arXiv:1911.05351, 2019.

[17] Taisung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[18] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271), pages 59–66. IEEE, 1998.

[19] Sara Sabour, Yanshuai Cao, Farshad Faghri, and David J Fleet. Adversarial manipulation of deep representations. arXiv preprint arXiv:1511.05122, 2015.

[20] Kenneth Sullivan, Upamanyu Madhow, Shivkumar Chandrasekaran, and BS Manjunath. Steganalysis for markov cover data with applications to images. IEEE Transactions on Information Forensics and Security, 1(2):275–287, 2006.

[21] Kenneth Sullivan, Upamanyu Madhow, Shivkumar Chandrasekaran, and Bangalore S Manjunath. Steganalysis of spread spectrum data hiding exploiting cover memory. In Security, Steganography, and Watermarking of Multimedia Contents VII, volume 5681, pages 38–46. International Society for Optics and Photonics, 2005.

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Cnn-generated images are surprisingly easy to spot... for now. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, volume 7, 2020.

[23] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. arXiv preprint arXiv:1801.02612, 2018.
[29] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365, 2015.

[30] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In 2019 IEEE International Workshop on Information Forensics and Security (WIFS), pages 1–6. IEEE, 2019.

[31] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. International Journal of Computer Vision, 127(3):302–321, 2019.

[32] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE international conference on computer vision, pages 2223–2232, 2017.