CNN-generated images are surprisingly easy to spot... for now – Supplementary Material

Sheng-Yu Wang[1] Oliver Wang[2] Richard Zhang[2] Andrew Owens[1,3] Alexei A. Efros[1]
UC Berkeley[1] Adobe Research[2] University of Michigan[3]

1. Additional Analysis

1.1. Additional ranking visualizations

In we rank ordered the fake images according to how "fake" the classifier deemed them to be. These full ranking results are included in the following link: https://peterwang512.github.io/CNNDetection/ranking/. We randomly select 20 real and 20 fake images from each dataset, and rank all images based on our **(Blur+JPEG (0.1))** model's scores. Note that there is a clear separation between real and fake images, where the real images have lower "fakeness" score and vice versa. Moreover, we observe the synthetic images ranked more "real" are super resolution (SAN) outputs, and the ones ranked more "fake" are CRN and IMLE outputs. However, we observe little noticeable correlation between the model predictions and the visual quality of the synthesized images in each dataset, where BigGAN and StarGAN images are the exceptions.

1.2. Effect of dataset size

We include additional ablation studies on the effect of dataset size, and the results are shown in Table 1. To compare with the dataset diversity ablation in Section 4.3 of the main text, we train 4 additional models with 10%, 20%, 40%, 80% of the entire dataset respectively, while having all 20 LSUN classes included in the training set. Same augmentation scheme as **Blur+JPEG (0.5)** is applied to all models. We observe much less reduction in generalization performance, indicating data diversity, comparing to dataset size, contributes more towards better CNN detection in general.

1.3. Comparison to training on a different model

To evaluate the choice of training architecture, we also include a model that is trained *solely* on BigGAN. To prepare the training data, we generate 400k fake images from an ImageNet-pretrained 256×256 BigGAN model [3], and take 400k ImageNet images with the same class distribution as real images. For comparison, we train the model with the same data augmentation as **Blur+JPEG (0.5)**. We denote this model as **Blur+JPEG (Big)**. We see in Table 1 that this model also exhibits generalization, albeit with slightly lower results in most cases. One explanation for this is that while our ProGANs model was trained on an ensemble (one model per class), BigGAN images were generated with a single model.

1.4. Training with images generated with a deep image prior

Instead of generating fake images with GANs, which have limited representational capacity and hence large synthesis errors, we consider an "oracle" generation method based on the deep image prior (DIP) [22]. We ask what the *very best* reconstruction of an image is achievable via a given network architecture, regardless of the synthesis task. For each synthesized image in our dataset, we train a *different* network to reconstruct it by minimizing ℓ_1 loss:

$$\min_{\theta} ||f(\theta_i) - I_i||_1, \tag{1}$$

where $f(\theta_i)$ is the image generated by a neural network parameterized with weights θ_i and I_i is a real image. We use the reconstructed image $f(\theta_i)$ as an instance of a fake image. During reconstruction, we use the Adam optimizer [12] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and optimized with a decreasing learning rate: $0.01 \rightarrow 0.001 \rightarrow 0.0001$. For each learning rate we optimize for 2000 iterations.

As training data, we take 44k real images randomly sampled from ImageNet [21], and "fake" images are the reconstruction by the generator of ProGAN (and hence 44k different networks). We take DIP images optimized for 1000, 2000, 3000, 4000, 5000, 6000 iterations into our "fake" image set. We then train a classifier on this dataset, and we over-sample the real images by 6 times to balance the classes. All training configurations and augmentations are same as **Blur+JPEG (0.5)**. This model is denoted as **DIP** in Tab. 1.

We note although this model does not perform as well as the model directly trained on ProGAN images, but it is able to detect several datasets, including StarGAN, CRN, SITD,

Family	Name	Training settings					Individual test generators											Total
		Train	Input	No. Class	Augments Blur	Augments JPEG	Pro-GAN	Style-GAN	Big-GAN	Cycle-GAN	Star-GAN	Gau-GAN	CRN	IMLE	SITD	SAN	Deep-Fake	**mAP**
Nataraj *et al*. [17]	–	CycleGAN	Co-occur. mtx	–			76.4	96.5	56.4	100.	88.2	56.2	58.7	83.1	39.6	46.1	55.1	68.8
Cozzolino *et al*. [7]	ForensicTransfer	ProGAN	HF residual	–			88.9	77.9	79.5	77.2	91.7	83.3	**99.9**	31.3	72.8	90.8	79.2	79.3
Ours	DIP	ProGAN-DIP	RGB	–	✓	✓	62.0	52.3	61.7	62.4	**100.**	49.0	98.2	38.6	92.8	**93.1**	63.1	70.3
	Blur+JPEG (Big)	BigGAN	RGB	1000	✓	✓	**85.1**	82.4	100.	86.2	87.4	96.7	79.7	82.6	91.2	71.9	60.3	83.9
	2-class	ProGAN	RGB	2	✓	✓	98.8	78.3	66.4	88.7	87.3	87.4	94.0	97.3	85.2	52.9	58.1	81.3
	4-class	ProGAN	RGB	4	✓	✓	99.8	87.0	74.0	93.2	92.3	94.1	95.8	97.5	87.8	58.5	59.6	85.4
	8-class	ProGAN	RGB	8	✓	✓	99.9	94.2	78.9	94.3	91.9	95.4	98.9	99.4	91.2	58.6	63.8	87.9
	16-class	ProGAN	RGB	16	✓	✓	100.	98.2	87.7	96.4	95.5	**98.1**	99.0	**99.7**	**95.3**	63.1	**71.9**	**91.4**
	10% data	ProGAN	RGB	20	✓	✓	100.	93.2	82.3	94.1	93.2	97.1	96.8	99.4	88.2	58.1	63.5	87.8
	20% data	ProGAN	RGB	20	✓	✓	100.	96.8	85.9	95.9	93.6	97.9	98.7	99.5	90.2	61.8	65.2	89.6
	40% data	ProGAN	RGB	20	✓	✓	100.	97.8	87.5	96.0	95.3	98.1	98.2	99.3	91.2	61.4	67.9	90.2
	80% data	ProGAN	RGB	20	✓	✓	100.	98.1	88.1	96.4	95.4	98.0	98.9	99.4	93.0	63.8	65.1	90.6
	Blur+JPEG (0.5)	ProGAN	RGB	20	✓	✓	100.	**98.5**	**88.2**	**96.8**	95.4	**98.1**	98.9	99.5	92.7	63.9	66.3	90.8

Table 1: **Additional evaluations.** We evaluate other baseline models, classifiers trained with DIP and BigGAN images, respectively, and classifiers trained with various dataset size. Same as Table 2 in the main text, we show the average precision (AP) of the models tested across 11 generators. For comparison, we include the ablations on the number of classes and the **Blur+JPEG (0.5)** model's results, which are presented in the main text. Symbols ✓ means the augmentation is applied with 50% or probability at training. The color coding scheme is identical to that of Table 2 in the main text. We note that when only the dataset size is reduced, AP dropped less comparing to reducing the number of classes. Also, the model trained with ProGAN outperforms the baselines, **DIP** and **Blur+JPEG (Big).**

and SAN. This indicates that low-level artifacts shares across different methods, but just leveraging on those may not be sufficient for a general detection.

1.5. Comparison to other baselines

In the main text, we compared with Zhang *et al*. [24], a state-of-the-art in GAN detection, and outperform it across different synthesis methods. In addition, we include the performance of Nataraj *et al*. [17], another GAN detection method trained on co-occurrence matrices of images, and Cozzolino *et al*. [7], a few-shot single-target domain adaption method trained on HF filtered images. For Cozzolino *et al*., we evaluate the ProGAN/CycleGAN model. Both methods are evaluated on 256×256 images in a zero-shot setting, and if the image is larger than 256 pixels, it is center-cropped to 256 pixels. The results are in Tab. 1.

1.6. Other evaluation metrics

To help clarify the threshold-less AP evaluation metric, we also computed several other metrics (Table 2). We provide the precision and recall curve on each dataset from our **(Blur+JPEG (0.1))** model in Figure 1. We give the *uncalibrated* generalization accuracy of the model on the test distribution, by simply using the classifier threshold we learned during training, and *oracle* accuracy that chooses the threshold that maximizes accuracy on the test set. We also consider a *two-shot* regime where we have access to one real and one fake image from each dataset, and only the model's *threshold* is adjusted during the two-shot calibration process.

We calibrate the model by a single random real and fake pair, and we augment the image pair by taking 224×224 random crops 128 times. The images are passed into the model to get the logits, which are then fitted by a logistic regression (the method is also known as Platt scaling [19]). We take the bias learned from the logistic regression to adjust the base rate of our model. Specifically, we apply the bias to our model's logit and then take the sigmoid to get the calibrated probability.

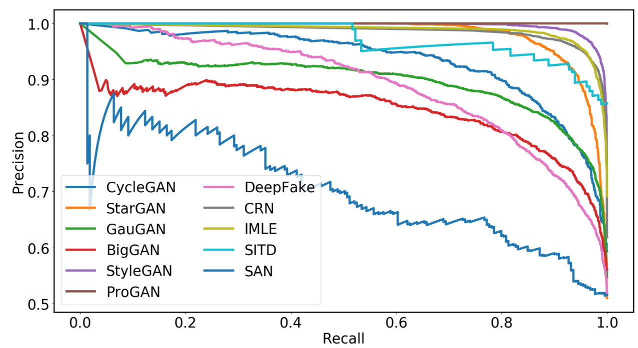

Figure 1: **Precision and recall curves.** The PR curves on each dataset from the **(Blur+JPEG (0.1))** model are shown. Note that AP is defined as the area under the PR curve. Higher AP indicates better trade-off between precision and recall, and vice versa.

1.7. Detecting GAN images from the internet

Unfortunately, there are currently no collections of "in-the-wild" CNN-generated image datasets which we can evaluate with our model. As a "proxy" testcase, we scraped 1k real face and 1k fake faces from `whichfaceisreal.com` [2]. This is a website containing StyleGAN-generated faces and real faces in 1024 pixels, with all images compressed into JPEG. We tested our **Blur+JPEG (0.1)** model on this testset in two scenarios: (1) directly center crop images to 224 pixels without resizing (matching how we test StyleGAN) or (2) resize to 256 pixels then center crop to

	StyleGAN	BigGAN	CycleGAN	StarGAN	GauGAN	CRN	IMLE	SITD	SAN	DeepFake
Uncalibrated	87.1	70.2	85.2	91.7	78.9	86.3	86.2	90.3	50.5	53.5
Oracle	96.8	81.1	86.3	92.8	85.5	95.3	95.4	92.8	68.0	80.7
Two-shot	91.9	74.0	82.4	86.0	79.1	91.6	91.2	88.7	54.8	65.7

Table 2: **Two-shot classifier calibration.** We show the accuracy of the classifiers directly trained from ProGAN ("uncalibrated"), after calibrating the threshold given two examples in the test distribution ("two-shot") and an upper bound, given a perfect calibration ("oracle").

Horse	Zebra	Summer	Winter	Apple	Orange	Facades	Cityscape	Map	Ukiyoe	Vangogh	Cezanne	Monet	Photo	**Avg.**
62.1	87.5	83.2	88.0	90.5	87.7	100.	66.6	78.0	85.4	76.9	82.8	56.2	86.8	80.8

Table 3: **CycleGAN testcase.** We evaluate the uncalibrated accuracy of **Blur+JPEG (0.1)** model tested on each CycleGAN category. We note that our model is still able to perform well above chance (50%) even if not directly trained on any CycleGAN images.

224 pixels. Without resizing, the model gets 83.6% accuracy and 93.2% AP. With resizing, the model drops to 74.9% accuracy and 82.6% AP, still well above chance (50%). This indicates our model can be robust to resizing and in-the-wild JPEG compression. However, maintaining similar performance after significant post-processing (e.g., heavy resizing) remains challenging.

1.8. CycleGAN testcase

While prior works on GAN detection [16, 17, 24] train on CycleGAN images and evaluate generalization across CycleGAN *categories*, our method is not trained on any CycleGAN images and tests generalization across *methods* (a significantly harder task). Nonetheless, we still observe comparable performance in terms of AP (Tab. 2 in the main text) when compared to Zhang *et al*. [24]. For a further comparison, we include our **Blur+JPEG (0.1)** model's accuracy on each CycleGAN category in Tab. 3.

2. Implementation Details

2.1. Dataset Collection

ProGAN [9] [1] We take 20 officially released ProGAN models pretrained on LSUN [23] airplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train, tv-monitor respectively. Following the official code, we sample the synthetic images with $z \sim N(0, I)$, and generate real images by center cropping the images just on the long edge (center crop length is exactly the length of the short edge) and then resizing to 256×256

StyleGAN [10] [2] We take officially released StyleGAN models pretrained on LSUN [23] bedroom, cat and car, with size 256×256, 256×256 and 512×384 respectively. We download the released synthesized images, all of which are generated with 0.5 truncation, and following the code, we generate real images by resizing to the according size of each category.

StyleGAN2 [11] [3] We take officially released StyleGAN2 config-F models pretrained on LSUN [23] church, cat, horse and car, with size 256×256, 256×256, 256×256 and 512×384 respectively. We download the released synthesized images, all of which are generated with 0.5 truncation, and following the code, we generate real images by resizing to the according size of each category.

BigGAN [3] [4] We take officially released BigGAN-deep model pretrained on 256×256 ImageNet images. Following the official code, we sample the images with uniform class distribution and with 0.4 truncation; also, we generate real images by center cropping the images just on the long edge (center crop length is exactly the length of the short edge) and then resizing to 256×256.

CycleGAN [25] [5] We take officially released CycleGAN models: apple2orange, orange2apple, horse2zebra, zebra2horse, summer2winter, winter2summer, and generate real and fake image pairs out of all six categories. Pre-processed real images and synthetic images are directly generated from the released code.

StarGAN [6] [6] We take officially released StarGAN model pretrained on CelebA [15], and generate real and fake image pairs. Pre-processed real images and synthetic images are directly generated from the released code.

GauGAN [18] [7] We take officially released GauGAN model pretrained on COCO [14], and generate real and fake

[1] https://github.com/tkarras/progressive_growing_of_gans
[2] https://github.com/NVlabs/stylegan
[3] https://github.com/NVlabs/stylegan2
[4] https://tfhub.dev/s?q=biggan
[5] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
[6] https://github.com/yunjey/stargan
[7] https://github.com/NVlabs/SPADE

image pairs. Pre-processed real images and synthetic images are directly generated from the released code.

CRN [5] [8] We take officially released CRN model pretrained on GTA, and generate synthesized images from preprocessed segmentation maps. Pre-processed real images and segmentation maps are downloaded from the IMLE repository.

IMLE [13] [9] We take officially released IMLE model pretrained on GTA, and generate synthesized images from preprocessed segmentation maps. Pre-processed real images and segmentation maps are downloaded from the official repository.

SITD [4] [10] We take officially released pretrained model and the dataset by Sony and Fuji cameras from the repository. Pre-processed real images and synthetic images are directly generated from the released code.

SAN [8] [11] We take both the ground truth and the officially released 4x super-resolution predictions on the standard benchmark datasets: Set5, Set14, BSD100 and Urban100. The synthetic images are directly downloaded from the repository.

DeepFake [20] [12] We download raw manipulated and original image sequences in the validation and test split of the Deepfakes dataset. We extracted all frames from the videos, and in each frame a face is detected and cropped using Faced [1]. Similar to [20], our dataset is comprised entirely of cropped faces.

2.2. Training details

To train the classifiers, we use the Adam optimizer [12] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, batch size 64, and initial learning rate 10^{-4}. Learning rate are dropped by $10\times$ if after 5 epochs the validation accuracy does not increase by 0.1%, and we terminate training at learning rate 10^{-6}. One exception is that, in order to balance training iterations with the size of the training set, for the $\{2, 4, 8, 16\}$-class models and the $\{10, 20, 40, 80\}\%$-data models, the learning rate is dropped if the validation accuracy plateaus for $\{50, 25, 13, 7\}$ epochs instead.

References

[1] Faced. https://github.com/iitzco/faced. 4

[2] Which face is real? http://www.whichfaceisreal.com/. 2

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019. 1, 3

[4] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *CVPR*, 2018. 4

[5] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017. 4

[6] Yunjey 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 *CVPR*, 2018. 3

[7] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*, 2018. 2

[8] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *CVPR*, 2019. 4

[9] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 3

[10] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 3

[11] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 3

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015. 1, 4

[13] Ke Li, Tianhao Zhang, and Jitendra Malik. Diverse image synthesis from semantic layouts via conditional imle. In *ICCV*, 2019. 4

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 3

[15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. 3

[16] Francesco Marra, Diego Gragnaniello, Davide Cozzolino, and Luisa Verdoliva. Detection of gan-generated fake images over social networks. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2018. 3

[17] Lakshmanan Nataraj, Tajuddin Manhar Mohammed, BS Manjunath, Shivkumar Chandrasekaran, Arjuna Flenner, Jawadul H Bappy, and Amit K Roy-Chowdhury. Detecting gan generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019. 2, 3

[18] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 3

[19] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. 1999. 2

[20] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics++: Learning to detect manipulated facial images. In *ICCV*, 2019. 4

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy,

[8] https://github.com/CQFIO/PhotographicImageSynthesis

[9] https://github.com/zth667/Diverse-Image-Synthesis-from-Semantic-Layout

[10] https://github.com/cchen156/Learning-to-See-in-the-Dark

[11] https://github.com/daitao/SAN

[12] https://github.com/ondyari/FaceForensics

Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 1

[22] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, 2018. 1

[23] 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. 3

[24] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *WIFS*, 2019. 2, 3

[25] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 3