What makes fake images detectable? Understanding properties that generalize: Supplementary Materials

Lucy Chai[1], David Bau[1], Ser-Nam Lim[2], and Phillip Isola[1]

[1] MIT CSAIL, Cambridge MA, 02139
{lrchai,davidbau,phillipi}@csail.mit.edu
[2] sernam@gmail.com

In the Supplementary Materials we include additional details on our dataset extraction process for both real and fake images (Sec. 1.1–1.2) and details on training the classifiers and visualization (Sec. 1.3–1.5). We also include a number of additional experiments and visualizations. Namely, classifiers for generated images are extremely sensitive to subtle differences in preprocessing, and minor differences will allow the classifier to easily learn preprocessing artifacts rather than forensic signals, which gives the appearance of high generalization capability – we include results that demonstrate this effect in Sec. 2.1. We also investigate additional training configurations and visualizations in Sec. 2.2–2.7.

1 Supplementary Methods

1.1 Dataset Construction

Real Images

CelebA-HQ: We prepare the CelebA-HQ images following the dataset preparation pipeline in the PGAN repository.[3]. This saves the images in various resolutions from 4x4px to 1024x1024px in a `tfrecord` format, which are then subsequently used to train the PGAN generators. For our real CelebA-HQ dataset, we extract the images from the `tfrecord` format, similar to during PGAN training, at the same resolution of the generator – i.e., if the fake images are generated from a 128px PGAN, then the real images used for comparison are extracted from the 128px `tfrecord`. This is to avoid any resizing operations that are different from the subsampling used during GAN training. We partition the images into training and validation images following the CelebA partitions, which yields 24183 images for training and 2993 for validation, and 2824 for testing. All images are resized to 128x128 resolution using Lanczos interpolation and saved in PNG format.

FFHQ: We download the prepared images in `tfrecord` format directly from the FFHQ repository[4] and extract them at 1024 resolution for comparison with 1024 resolution generator models. Following the suggested partition in the repository,

[3] https://github.com/tkarras/progressive_growing_of_gans
[4] https://github.com/NVlabs/ffhq-dataset

we reserve the first 60000 images as the training set, the next 5000 for validation and the final 5000 for testing. All images are resized to 128x128 resolution using Lanczos interpolation and saved in PNG format.

CelebA: We follow the data transformation pipeline used to train the CelebA GMM model.[5], which includes a crop operation to localize the face and bilinear resize to 64 pixel resolution for comparison to the fake images generated by the GMM. We use the testing split of 19962 images for classifier evaluation. All images are resized to 128x128 resolution using Lanczos interpolation and saved in PNG format. The CelebA dataset is a superset of the faces in CelebA-HQ, although at a lower resolution and with a slightly different facial crop.

Generated Images

PGAN: We download the pretrained 1024px resolution CelebA-HQ generator from the PGAN repository.[3] Separately, we train PGANs on the CelebA-HQ dataset to 128px, 256px, and 512px final resolutions. We change the random initialization seed and train 3 additional PGANs to 128px resolution. We also train a PGAN to 1024px resolution on the FFHQ dataset. All images are sampled from the generator, resized to 128x128 resolution using Lanczos interpolation, and saved in PNG format. We sample the same number of images as used in each split of the corresponding real dataset.

SGAN/SGAN2: We download the pretrained 1024px resolution generators trained on the CelebA-HQ and FFHQ datasets from the StyleGAN repository,[6] and the the pretrained 1024px resolution generator trained on the FFHQ dataset from the StyleGAN2 repository.[7] Images are sampled from the generator, resized to 128x128 resolution using Lanczos interpolation, and saved in PNG format. We sample the same number of images as used in each split of the corresponding real dataset.

GMM: We train the Mixture of Factor Analyzers Gaussian Mixture model based on [7] using the default training settings provided in the source code.[5] Images are sampled from the generator, resized to 128x128 resolution using Lanczos interpolation, and saved in PNG format. We sample the same number of images as used in each split of the corresponding real dataset.

Glow: We sample images from the 1024px resolution pretrained model in the Glow repository.[8] We perform random manipulation of attributes by selecting an attribute tag and a manipulation amount within the range $[-1, 1]$ uniformly at random, and apply this to each sample. Images are then resized to 128x128 resolution using Lanczos interpolation and saved in PNG format. We sample the same number of images as used in each split of the corresponding real dataset.

[5] `https://github.com/eitanrich/torch-mfa`
[6] `https://github.com/NVlabs/stylegan`
[7] `https://github.com/NVlabs/stylegan2`
[8] `https://github.com/openai/glow`

FaceForensics

We download the Deepfakes, Face2Face, FaceSwap, NeuralTextures, and original videos from the FaceForensics++ dataset [8][9] in compressed format. We use the training and validation splits suggested by the authors. To localize the face, we use the dlib face detector[10] to extract eyes, nose, and mouth landmarks, and based on these detected landmarks, align and crop the the frames using the CelebA-HQ alignment approach (the CelebA images are annotated with these landmarks a priori, which are then used to generate the CelebA-HQ images; here, we automate that process). For training images, we extract all frames in the corresponding training videos. For validation and testing images, we extract 100 frames per video to prevent any video from have more influence than the others.

1.2 Reprojecting Fake Images

Apart from generating fake samples by sampling from a generative model, we can also create hard negative examples by generating fake GAN images that are most similar to a given real target image – i.e., we project real images to the output manifold of the generative model. To create these images, we use the approach in [2], which uses a hybrid encoder and optimization approach. First, given an image x, an encoder E is trained layer-wise such that $G(E(x)) \approx x$. This provides a latent code initialization $z = E(x)$ for the second optimization step, which uses an LBGFS optimizer over z to minimize $|x - G(z)|$. We add these reprojected examples to the fake image dataset when training the classifier, but also conduct experiments in which the classifier is trained 1) without reprojected images and 2) only on reprojected images as fake samples in Sec. 2.4.

Table 1. Model receptive field and parameter count calculations.

Truncated Model	RF	# Params	Full Model	# Params
Resnet Layer 1	43	0.158 M	[4] Resnet-18	11.178 M
Xception Block 1	19	0.055 M	[1] MesoInception4	0.029 M
Xception Block 2	43	0.191 M	[3] Xception	20.811 M
Xception Block 3	91	1.108 M	[9] CNN (p=0.1)	23.510 M
Xception Block 4	187	2.722 M	[9] CNN (p=0.5)	23.510 M
Xception Block 5	263	4.336 M		

1.3 Patch Classifiers Architecture

Truncating a classifier reduces the receptive field and number of parameters of the model. Effectively, it increases the ratio of data to model size, as the same model weights are used across all patches of an input image. In Table 1, we provide calculations on the receptive field size and number of parameters

[9] `https://github.com/ondyari/FaceForensics/`
[10] `https://github.com/davisking/dlib`

for each model used in our experiments. We construct truncated classifiers using Resnet [4] and Xception [3] models as backbones. The Resnet architecture consists of four layers containing residual skip connections – in our experiments we observe that truncating after the first residual layer often performs best. The Xception architecture consists of 12 blocks with residual connections and separable convolutions, we conduct experiments truncating after the first five blocks.

1.4 Additional Training Details

We train all models using the Adam optimizer with default parameters and learning rate (0.001). We use a batch size of 32 images, consisting of 16 real and 16 fake images. After every epoch, we measure raw patch-wise prediction accuracy (without ensembling patch decisions) on validation images corresponding to the training dataset. We stop training when validation accuracy does not improve for a patience parameter of $5 * p$ epochs - we use $p = 50$ for the Xception Block 1 patch classifier, $p = 20$ for the Xception Block 2 patch classifier, and $p = 10$ for the deeper models. We use the checkpoint with the highest raw validation accuracy to evaluate on the test data split. For training, we resize all images to native size for each model – 299 for Xception architectures, 224 for Resnet architectures, and 256 for MesoNet architectures.

When training with reprojected images, we have paired examples of the original image and the reprojected image. When creating batches, we sample both the original and the reprojection within the batch. We use a similar approach for the FaceForensics++ dataset, where we have paired examples of the original and manipulated images. This creates hard negative samples, which we found to improve classification performance. When training with the FaceForensics++ dataset, we do not use information about the manipulated region – rather, the classifier's objective is simply to predict all patches in the real image as real and all patches in the fake image as fake. Because the background is unlearnable, the classifier's predictions in the background region for the FaceForensics++ datasets remain uncertain. However, [6] notes that using the additional mask location as learning supervision improves classification.

1.5 Facial Segmentation Model

To visualize the patch-wise model decisions, we use a facial segmentation network and cluster patches by semantic category. Precisely, for each real or fake image, we take the most predictive patch in the image and label it by semantic category. For the segmentation task, we use a BiSeNet pretrained on the CelebAMask-HQ dataset.[11] The network output assigns each input pixel to one of 19 categories: background, skin, left/right brow, left/right eye, eyeglasses, left/right ear, earring, nose, mouth, upper/lower lip, neck, necklace, clothes, hair, and hat. We group together the left/right brow classes into a brow category, left/right eye and

[11] `https://github.com/zllrunning/face-parsing.PyTorch`

eyeglasses into eyes, left/right ear and earring into ears, mouth and upper/lower lip into mouth, and neck and necklace into neck, which yields 11 semantic categories in total.

Given an image, we use the segmentation network to obtain a semantic class prediction for each pixel. To assign the patch to a cluster, we tally the number of pixels in the patch belonging to each segmentation class, and normalize by the total number of pixels labelled as that class in the image. We assign the patch to the cluster with the highest normalized proportion. This normalization step helps to appropriately weight classes of small features – e.g. in a patch containing eyebrows, the most common cluster assignment will be skin, but the most common normalized cluster assignment will be eyebrows since the total number of eyebrow pixels in the full image is lower.

2 Additional Experiments

2.1 Preprocessing

One critical step of the real vs. fake classification task is to make sure that the classifier is not simply learning differences in the preprocessing of real or fake images, especially since we do not know the exact preprocessing steps taken in the real image datasets. The solution we use is to pass the real images through the data transformation used to train the generator, so that we can construct the real image dataset in a manner as similar as possible to the fake images output from the generator. In our early experiments, we found that a few subtle differences in this preprocessing pipeline will allow the model to learn differences in preprocessing. If all real and fake datasets (training and testing) have this discrepancy, then this will lead to artificially high generalization performance.

Table 2. Slight differences in the preprocessing for real or fake images leads to seemingly increases generalization, as the classifier ends up exploiting these differences. We report average precision for a classifier trained on PGAN fake images on the CelebA-HQ dataset and tested on the remaining datasets. AP on the test set corresponding to training images is colored in gray.

		Architectures			**FFHQ dataset**		
Preprocessing	**PGAN**	**SGAN**	**GLOW**	**GMM**	**PGAN**	**SGAN**	**SGAN2**
identical preprocessing	100.00	64.80	47.06	54.69	79.20	51.15	52.37
different interpolation	99.93	99.33	99.76	85.40	99.44	99.08	99.09
different initial size	100.00	100.00	100.00	97.81	100.00	100.00	100.00
different image format	100.00	100.00	100.00	100.00	100.00	100.00	100.00

The standard pipeline that we use in the main text for training the fake-image classifier consists of the following steps for real images:

1. We pass real images through generator data transform.
2. All images are resized to the same size (128px) before saving in PNG format.
3. When loading the image during training, we resize the image to the classifier's native resolution, perform mean centering, and then input to the classifier.

For the fake images, we replace step 1 with sampling and renormalizing the output from the generator.

We experiment with 3 small variations on the preprocessing pipeline, and apply these variations consistently across the CelebA-HQ, FFHQ, and CelebA real-image datasets.

- interpolation: Before saving to file, real and fake images undergo different interpolation methods (bilinear vs. lanczos).
- initial size: Real and fake images are saved to file in different resolutions.
- image format: Real and fake images are saved in different image codecs (JPG vs PNG).

Average precision results using the full Resnet-18 model (generally the weaker model in our generalization experiments) are shown in Table 2. When controlling for image format, interpolation, and size differences, the performance of the classifier decreases when tested on different datasets. However, training the same model in which the real dataset is differently processed from the fake dataset leads to the appearance of very high generalization, when in fact the classifier is just learning the differences in preprocessing. These results show that subtle artifacts in preprocessing are in fact easy to learn, and that these classifiers are very sensitive to the types of preprocessing used during training and inference time. Note that these preprocessing artifacts are still learnable even despite a second resizing step which converts real and fake images to the same size with the same interpolation method prior to being input to the classifier.

2.2 Additional Training Variations

Alignment vs Cropping. In our main experiments, we train with all patches of aligned facial image and ensemble patch-wise predictions to obtain a binary prediction. We also experimented with random cropping and random resized cropping augmentations during training by resizing the image to 333px resolution, and take a random crop (or random resized crop) of 299 pixels for Xception network training. We compare the classification results on aligned faces, random crops, and random resized crops in Tables 3, 4, and 5 respectively. The results with added cropping are somewhat mixed – random cropping boosts Glow generalization, although not as much as training without reprojected images (in Table 9), and random resized cropping helps slightly in the GMM case, while the FFHQ datasets fare better without cropping augmentation.

A possible explanation for this behavior is that faces are naturally structured and can be aligned via facial landmarks – CelebA-HQ and FFHQ datasets are already aligned, whereas we automatically align the FaceForensics dataset using a facial landmark detector (see Table 6 for a comparison). Secondly, truncated

Table 3. Average precision on test datasets when trained on CelebA-HQ PGAN images. AP on the test set corresponding to training images is colored in gray.

		Architectures			FFHQ dataset		
Model	**PGAN**	**SGAN**	**GLOW**	**GMM**	**PGAN**	**SGAN**	**SGAN2**
Xception Block 1	100.00	98.68	**82.39**	76.21	99.68	81.35	77.40
Xception Block 2	100.00	99.99	46.35	**91.38**	**100.00**	90.12	90.85
Xception Block 3	100.00	**100.00**	64.77	80.96	100.00	92.91	**91.45**
Xception Block 4	100.00	99.99	51.80	42.82	100.00	**95.85**	90.62
Xception Block 5	100.00	100.00	58.18	48.92	100.00	93.09	89.08

Table 4. Average precision on test datasets when trained on CelebA-HQ PGAN images. Random cropping is applied during training.

		Architectures			FFHQ dataset		
Model	**PGAN**	**SGAN**	**GLOW**	**GMM**	**PGAN**	**SGAN**	**SGAN2**
Xception Block 1	100.00	97.87	**89.60**	84.00	99.48	79.00	76.64
Xception Block 2	100.00	99.93	47.02	82.47	99.99	88.40	**89.58**
Xception Block 3	100.00	**99.97**	41.24	**86.76**	**100.00**	86.69	87.25
Xception Block 4	100.00	99.85	56.84	66.14	100.00	**91.60**	88.36
Xception Block 5	100.00	99.63	54.39	84.52	99.99	89.56	88.86

Table 5. Average precision on test datasets when trained on CelebA-HQ PGAN images. Random resized cropping is applied during training.

		Architectures			FFHQ dataset		
Model	**PGAN**	**SGAN**	**GLOW**	**GMM**	**PGAN**	**SGAN**	**SGAN2**
Xception Block 1	100.00	93.22	63.99	74.13	98.81	76.83	71.04
Xception Block 2	100.00	99.78	46.35	56.80	99.98	85.93	85.71
Xception Block 3	100.00	**99.91**	50.46	56.34	99.99	**90.10**	**90.11**
Xception Block 4	100.00	99.80	36.94	80.88	**100.00**	88.69	86.80
Xception Block 5	100.00	99.79	**64.22**	**92.75**	99.99	89.99	89.22

models can tolerate minor shifts, as the same model weights are applied over local patches in a sliding fashion over the image.

2.3 Receptive field vs. number of parameters

There are two main differences between a shallow truncated model and a deeper one – receptive field and number of parameters. Hypothetically, both factors could reduce overfitting and improve generalization. In Table 7 we seek disentangle these two components when evaluating generalization on test datasets. We take

Table 6. Average precision on test datasets when trained on Face2Face manipulated images. Images are automatically aligned using facial landmarks. We compare average precision on test images when trained on aligned patches (left four columns) and random crops of patches (right four columns).

	Face Alignment				**Random Cropping**			
Model	**DF**	**NT**	**F2F**	**FS**	**DF**	**NT**	**F2F**	**FS**
Xception Block 1	77.65	**80.88**	93.84	61.62	76.46	79.11	92.44	60.08
Xception Block 2	**84.04**	79.51	97.40	**63.21**	**83.19**	**80.61**	97.01	**63.59**
Xception Block 3	76.10	74.77	97.33	63.10	76.15	75.23	98.00	63.17
Xception Block 4	67.18	61.72	97.19	63.04	71.91	68.01	98.60	62.01
Xception Block 5	81.25	61.91	96.45	55.15	76.44	62.78	96.10	52.63

Table 7. Comparing the effect on model size and receptive field on test datasets. The Extended block2 model adds two additional Xception blocks modified with 1x1 convolutions to increase parameter count without increasing receptive field. AP on the test set corresponding to training images is colored in gray.

	Architectures				**FFHQ dataset**		
Model	**PGAN**	**SGAN**	**GLOW**	**GMM**	**PGAN**	**SGAN**	**SGAN2**
Xception Block 2	100.00	99.99	46.35	91.38	100.00	90.12	**90.85**
Extended Block 2	100.00	99.99	39.23	**91.54**	100.00	87.82	89.43
Xception Block 4	100.00	99.99	**51.80**	42.82	100.00	**95.85**	90.62

the Xception Block 2 and Xception Block 4 truncated models, and additionally train an Extended Block 2 model by adding 2 additional Xception blocks to the Block 2 model, modified with 1x1 convolutions. This increases the number of parameters of the Block 2 truncated model without increasing the receptive field. However, despite this increase in parameters, we do not see large increases in average precision on the test datasets, suggesting that perhaps the receptive field size contributes more to generalization on unseen faces at test time.

2.4 Training with reprojected image samples.

For detecting fake images created from generative models, we find that adding reprojected fake images – i.e., the GAN-generated image most similar a given real image, helps to improve performance on the test datasets in most cases. Here, we compare those results (Table 8) to models trained only on random samples from the fake image generators (Table 9) and models trained only on reprojected fake images (Table 10). One exception where adding the reprojected fake images harms classification, compared to training only with random samples, in the case of the Glow fake images where there is a difference in AP of over 10%. On the other hand, when training only with reprojected images there is a domain shift between train and test data, as the model is evaluated on random samples from

the generator which are never seen during training. Hence, the test AP is lower in most cases, except for the GMM model where it is similar.

How does training with reprojected images affect the patches that the classifier uses for classification? For this experiment, we take the same classifiers as before, trained on CelebA-HQ PGAN images. In Fig. 1 we show patches grouped by semantic category when the classifier is tested on various FFHQ generators. Training with reprojected images causes the classifier place greater emphasis on background patches, compared to training without reprojections, suggesting that adding the reprojection better allows the classifier to learn artifacts in the background portion of the image.

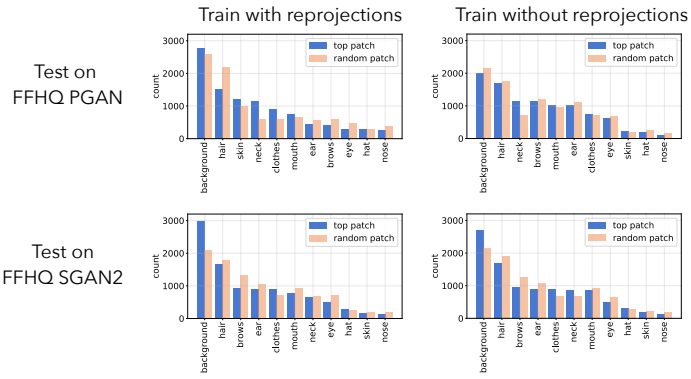

Fig. 1. Top patches categorized by segmentation category for classifiers trained with and with reprojected fake images. Adding reprojections to the training set causes the classifier to place greater emphasis on background patches.

2.5 Investigating biases in the classifiers

To investigate biases in the fake-image classifier, we take the pre-trained detector on male and female faces from [5] and compute average precision conditioned on the male or female classes predicted by the detector. Classifiers are on trained on CelebA-HQ faces and PGAN samples. When there is no domain gap between training and test time, the classifier can solve the task perfectly for both male and female categories. Next we test on two datasets where a domain gap exists – CelebA-HQ faces generated using the Glow model and FFHQ faces generated using the SGAN2 model. In these more difficult cases, the fake-image classifier obtains slightly higher AP on faces categorized as male by the pre-trained detector.

Table 8. Average precision on datasets when trained on CelebaHQ PGAN images and reprojected images as the fake image dataset. AP on the test set corresponding to training images is colored in gray.

		Architectures			FFHQ dataset		
Model	**PGAN**	**SGAN**	**GLOW**	**GMM**	**PGAN**	**SGAN**	**SGAN2**
Xception Block 1	100.00	98.68	**82.39**	76.21	99.68	81.35	77.40
Xception Block 2	100.00	99.99	46.35	**91.38**	**100.00**	90.12	90.85
Xception Block 3	100.00	**100.00**	64.77	80.96	100.00	92.91	**91.45**
Xception Block 4	100.00	99.99	51.80	42.82	100.00	**95.8**5	90.62
Xception Block 5	100.00	100.00	58.18	48.92	100.00	93.09	89.08

Table 9. Average precision on datasets when trained on only CelebA-HQ PGAN samples as the fake image dataset.

		Architectures			FFHQ dataset		
Model	**PGAN**	**SGAN**	**GLOW**	**GMM**	**PGAN**	**SGAN**	**SGAN2**
Xception Block 1	100.00	93.78	**95.48**	**78.91**	89.29	67.84	66.74
Xception Block 2	100.00	99.90	67.49	77.34	99.89	84.27	84.56
Xception Block 3	100.00	**99.92**	74.98	71.29	**99.97**	**88.49**	**87.78**
Xception Block 4	100.00	98.81	66.79	68.06	99.79	84.67	79.50
Xception Block 5	100.00	95.25	60.44	68.47	98.95	71.75	70.83

Table 10. Average precision on datasets when trained on only images reprojected via PGAN as the fake image dataset.

		Architectures			FFHQ dataset		
Model	**PGAN**	**SGAN**	**GLOW**	**GMM**	**PGAN**	**SGAN**	**SGAN2**
Xception Block 1	97.74	90.57	31.30	77.49	99.60	70.92	71.90
Xception Block 2	99.98	**99.34**	39.91	**92.03**	**99.97**	**84.00**	**84.27**
Xception Block 3	99.86	99.23	45.53	89.89	99.95	79.90	78.57
Xception Block 4	99.02	97.13	48.06	43.24	99.21	66.00	66.60
Xception Block 5	87.30	79.86	48.94	50.47	96.25	53.31	56.65

Table 11. We take a pre-trained classifier on male and female faces and calculate AP on the test images to investigate biases in the fake-image classifier.

Test Set	**AP Overall**	**# Total**	**AP male**	**# Male**	**AP female**	**# Female**
CelebAHQ PGAN	100.0	5986	100.0	2024	100.0	3962
CelebAHQ Glow	94.9	5986	96.6	2034	93.9	3952
FFHQ SGAN2	91.7	10000	93.3	4480	90.4	5520

2.6 Additional FaceForensics Visualizations

In the main text we show patch-wise visualizations and statistics for training on unmanipulated images and Face2Face images and testing on Neural Textures and Deepfakes images. Here we show similar visualization when trained on Deepfakes images in Fig. 2. In addition, we show examples of local classifier predictions and heatmaps of the top 100 most predictive images in Fig 3. While the heatmap of test images corresponding to the training set capture the general face area, heatmaps of images corresponding to different manipulation methods highlight more local features, such as lower face or eye regions.

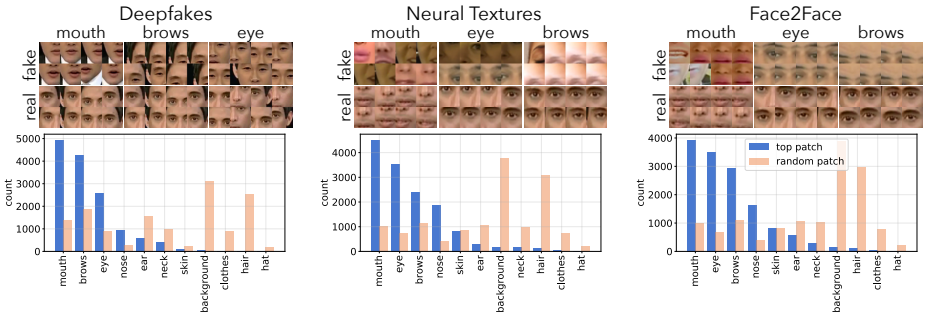

Fig. 2. Histograms of the most predictive patches from a classifier trained on Deepfakes and un-manipulated images, and tested on the Neural Textures and Face2Face manipulation methods

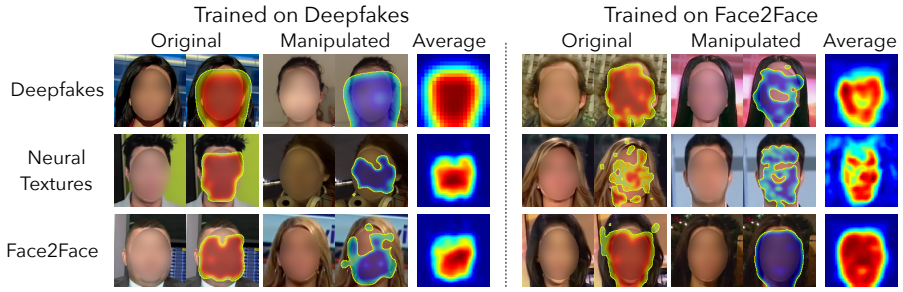

Fig. 3. Heatmaps showing examples of the patch-wise prediction output of a classifier on one FaceForensics method and tested on other methods. We also show the average heatmap over the 100 most predictive real and manipulated images for each dataset.

2.7 Example Images after Finetuning

In the main text, we finetune a face PGAN generator to evade classification by a fakeness detector, which drops detection accuracy to from 100% to below 65%. Fig. 4 shows random samples from the generator before and after finetuning. The samples remain visually similar despite finetuning but are misclassified by the detector. We further show in the main text that a secondary classifier trained on these finetuned images can recover in classification accuracy, which suggests that finetuning does not completely remove the detectable artifacts and the finetuned images are still distinguishable from real faces.

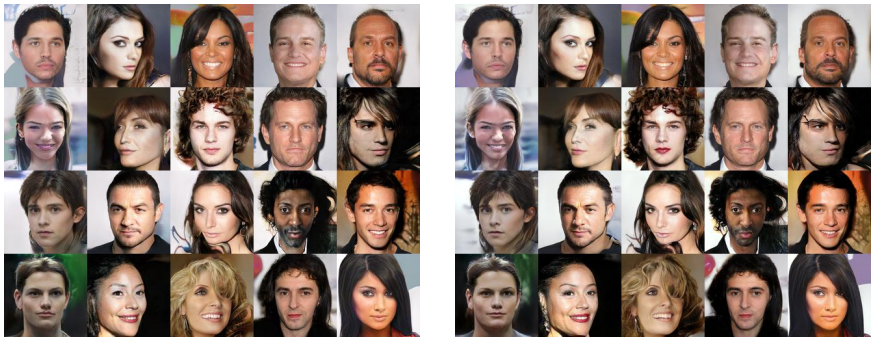

Fig. 4. Samples from PGAN generator before (left) and after (right) finetuning to evade a fakeness classifier.

References

1. Afchar, D., Nozick, V., Yamagishi, J., Echizen, I.: Mesonet: a compact facial video forgery detection network. In: 2018 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1–7. IEEE (2018)
2. Bau, D., Zhu, J.Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., Torralba, A.: Seeing what a gan cannot generate. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4502–4511 (2019)
3. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1251–1258 (2017)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
5. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)
6. Li, L., Bao, J., Zhang, T., Yang, H., Chen, D., Wen, F., Guo, B.: Face x-ray for more general face forgery detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5001–5010 (2020)
7. Richardson, E., Weiss, Y.: On gans and gmms. In: Advances in Neural Information Processing Systems. pp. 5847–5858 (2018)
8. Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., Nießner, M.: Faceforensics++: Learning to detect manipulated facial images. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1–11 (2019)
9. Wang, S.Y., Wang, O., Zhang, R., Owens, A., Efros, A.A.: Cnn-generated images are surprisingly easy to spot... for now. arXiv preprint arXiv:1912.11035 (2019)