What Makes Fake Images Detectable? Understanding Properties that Generalize

Lucy Chai(✉), David Bau, Ser-Nam Lim, and Phillip Isola

MIT CSAIL, Cambridge, MA 02139, USA
{lrchai,davidbau,phillipi}@csail.mit.edu, sernam@gmail.com

Abstract. The quality of image generation and manipulation is reaching impressive levels, making it increasingly difficult for a human to distinguish between what is real and what is fake. However, deep networks can still pick up on the subtle artifacts in these doctored images. We seek to understand what properties of fake images make them detectable and identify what generalizes across different model architectures, datasets, and variations in training. We use a patch-based classifier with limited receptive fields to visualize which regions of fake images are more easily detectable. We further show a technique to exaggerate these detectable properties and demonstrate that, even when the image generator is adversarially finetuned against a fake image classifier, it is still imperfect and leaves detectable artifacts in certain image patches. Code is available at https://github.com/chail/patch-forensics.

Keywords: Image forensics · Generative models · Image manipulation · Visualization · Generalization

1 Introduction

State-of-the-art image synthesis algorithms are constantly evolving, creating a challenge for fake image detection methods to match the pace of content creation. It is straightforward to train a deep network to classify real and fake images, but of particular interest is the ability of fake image detectors to generalize to unseen fake images. What artifacts do these fake image detectors look at, and which properties can allow a detector released today to work on novel fake images?

Generalization is highly desired in machine learning, with the hope that models work not only on training data, but also on related held-out examples as well. For tasks like object detection and classification, this has been accomplished with successively deeper and deeper networks that incorporate the context of the entire image to learn about global semantics and object characteristics. On the other hand, to learn image manipulation artifacts that are shared across various image generation pipelines, global content is not the only signal that matters. In

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58574-7_7) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12371, pp. 103–120, 2020.
https://doi.org/10.1007/978-3-030-58574-7_7

fact, two identical generators trained on the same training data, differing only in the random initialization seed, can create differences in content detectable by a deep network classifier [35]. Instead of these differences, we seek to identify what image generators have *in common*, so that training on examples generated from one model can help us identify fake images from another model.

Across different facial image generators, we hypothesize that global errors can differ but local errors may transfer: the global facial structure can vary among different generators and datasets, but local patches of a generated face are more stereotyped and may share redundant artifacts. Therefore, these local errors can be captured by a classifier focusing on textures [9] in small patches. We investigate a fully convolutional approach to training classifiers, allowing us to limit the receptive field of the model to focus on image patches. Furthermore, these patch-based predictions offer us a natural way to visualize patterns that are indicative of a real or fake image.

Using a suite of synthetic face datasets that span fully generative models [16,17,19,31] and facial manipulation methods [32], we find that more complex patches, such as hair, are detectable across various synthetic image sources when training on images from a single source. In one of our early experiments, however, we observed that we could obtain misleadingly high generalization simply due to subtle differences in image preprocessing – therefore, we introduce careful preprocessing to avoid simply learning differences in image formatting.

With a fixed classifier, an attacker can simply modify the generator to create adversarial examples of fake images, forcing them to become misclassified. Accordingly, we finetune a GAN to create these adversarial examples. We then show that a newly trained classifier can still detect images from this modified GAN, and we investigate properties of these detected patches. Our results here suggest that creating a coherent fake image without any traces of local artifacts is difficult: the modified generator is still unable to faithfully model certain regions of a fake image in a way that is indistinguishable from real ones.

Detecting fake images is a constant adversarial game with a number of ethical considerations. As of today, no method is completely bulletproof. Better generators, out-of-distribution images, or adversarial attacks [5,11] can defeat a fake-image detector, and our approach remains vulnerable to many of these same shortcomings. Furthermore, we train on widely used standard face datasets, but these are still images of real individuals. To protect the privacy of people in the dataset, we blur all real faces and manipulated real faces used in our figures. Our contributions are summarized as follows:

- To avoid learning image formatting artifacts, we preprocess our images to reduce formatting differences between real and fake images.
- We use a fully-convolutional patch-based classifier to focus on local patches rather than global structure, and test on different model resolutions, initialization seeds, network architectures, and image datasets.
- We visualize and categorize the patches that are most indicative of real or fake images across various test datasets.

– To visualize detectable properties of fake images, we manipulate the generated images to exaggerate characteristic attributes of fake images.
– Finetuned generators are able to overcome a fake-image detector, but a subsequent classifier shows that detectable mistakes still occur in certain image patches.

2 Related Work

Image Manipulation. Verifying image authenticity is not just a modern problem – historical instances of photo manipulation include a well-known portrait of Abraham Lincoln[1] and instances of image censorship in the former Soviet Union[2]. However, recent developments in graphics and deep learning make creating forged images easier than ever. One of these manipulation techniques is image splicing, which combines multiple images to form a composite [12]. This approach is directly relevant to face swapping, where a source face is swapped and blended onto a target background to make a person appear in a falsified setting. The deep learning analogue of face swapping, Deepfakes [1], has been the focus of much recent media attention. In parallel, improvements in generative adversarial networks (GANs) form another threat, as they are now able to create shockingly realistic images of faces simply from random Gaussian noise [16,17].

Automating Detection of Manipulated Images. Given the ease in creating manipulated images nowadays and the potential to use them for malicious purposes, a number of efforts have focused on automating detection of manipulated images. A possible solution involves checking for consistency throughout the image – examples include predicting metadata [14] or other low-level artifacts [27–29], learning similar embeddings for nearby patches [38], or learning similarity graphs from image patches [25]. Other works have focused on training classifiers for the detection task, using a deep network either directly on RGB images [2,4,32] or alternative image representations [7,26]. [30] uses a combination of both: a CNN to extract features over image patches and a separate classifier for prediction. Here we also take a patch-wise approach, and we use these patches to visualize the network decisions.

Can Detectors Generalize? The class of potential manipulations is so large that it is infeasible to cover all possible cases. Can a detector learn to distinguish real and fake images from one source and transfer that knowledge to a different source? Preprocessing is one way to encourage generalization, such as using spectral features [37] or adding blur and random noise [34]. [33] generalizes across a wide variety of datasets simply by adding various levels of augmentation. Specialized architectures also help generalization: for example [8] uses an

[1] https://www.bbc.com/future/article/20170629-the-hidden-signs-that-can-reveal-if-a-photo-is-fake.
[2] https://en.wikipedia.org/wiki/Censorship_of_images_in_the_Soviet_Union.

autoencoder with a bottleneck that encourages different embeddings for real and fake images. A challenge with generalization is that the classifiers are not explicitly trained on the domain they are tested on. [22] demonstrates that it is possible to simulate the domain of manipulated images; by applying warping to source images, they can detect deepfake images without using manipulated images in classifier training. [21] further studies generalization across different facial manipulation techniques also using a simulated domain of blended real images. However, a remaining question is what features do these models rely on to transfer knowledge among different domains, which we seek to investigate here.

Classification with Local Receptive Fields. We use patch-based classification to visualize properties that generalize. Small receptive fields encourage the classifier to focus on local artifacts rather than global semantics, which is also an approach taken in GAN discriminators to encourage synthesis of realistic detailed textures [15]. A related concept is the Markovian generative adversarial network for texture synthesis [20]; the limited receptive field makes the assumption that only pixels within a certain radius affect the output, and the pixels outside that radius are independent from the output. [24] demonstrate a method for converting deep neural classifiers to fully convolutional networks and use patch-wise training, allowing the model to scale efficiently to arbitrarily-sized inputs, used for the task of semantic segmentation.

3 Using Patches for Image Forensics

Rather than training a network to predict a global "real" or "fake" decisions for an image, we use shallow networks with limited receptive fields that focus on small patches of the image. This approach allows us to localize regions of the image that are detected to be manipulated and ensemble the patch-wise decisions to obtain the overall prediction.

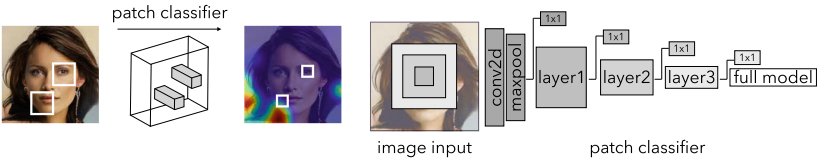

Fig. 1. We use a classifier with small receptive fields to obtain a heatmap over the patch-wise output. To obtain this patch classifier, we truncate various deep learning models after an initial sequence of layers.

3.1 Models for Patch-Based Classification

Modern deep learning architectures typically consist of a series of modular blocks. By truncating the models after an intermediate block, we can obtain model predictions based on a local region of the image, where truncating earlier in the layer sequence results in a smaller receptive field, while truncating after more layers results in a larger receptive field. We then add a 1×1 convolution layer after this truncated backbone to convert the feature representation into a binary real-or-fake prediction. We experiment with Resnet and Xception as our model backbones – generally we observe that Xception blocks perform better than Resnet blocks, however we also report results of the top performing Resnet block. We provide additional details on the model architecture and receptive field calculations in Supplementary Material Sect. 2.3.

The truncation operation reduces the size of the model's receptive field, and yields a prediction for a receptive-field-sized patch of the input, rather than the entire image at once. This forces the models to learn local properties that distinguish between real and fake images, where the same model weights are applied in a sliding fashion over the entire image, and each output prediction is only a function of a small localized patch of the image. We apply a cross entropy loss to each patch; i.e. every real patch should be considered real, and every fake image patch should be considered fake:

$$\mathcal{L}(\mathbf{x}) = \frac{1}{|P|} \sum_{i,j} \sum_{t} t \log f^t(x_{i,j}) \tag{1}$$

where f is the model output after a softmax operation to normalize the logits, t indexes over the real and fake output for binary classification, (i, j) indexes over the receptive field patches, and $|P|$ is the total number of patches per image. We train these models with the Adam optimizer with default learning rate, and terminate training when validation accuracy does not improve for a predetermined number of epochs.

By learning to classify patches, we increase the ratio of data points to model parameters: each patch of the image is treated independently, and the truncated models are smaller. The final classification output is an ensemble of the individual patch decisions rather than a single output probability. To aggregate patches at inference time, we take a simple average after applying a softmax operation to the patch-wise predictions:

$$t^* = \arg\max_t \left(\frac{1}{|P|} \sum_{i,j} f^t(x_{i,j}) \right) \tag{2}$$

The averaging approach can be applied in both cases where the image is wholly generated, or when only part of the image is manipulated. For example, when a generated face is spliced onto a real background, the background patches may not be predicted as fake; in this case, because the same background is present in both real and fake examples, the model remains uncertain in these locations.

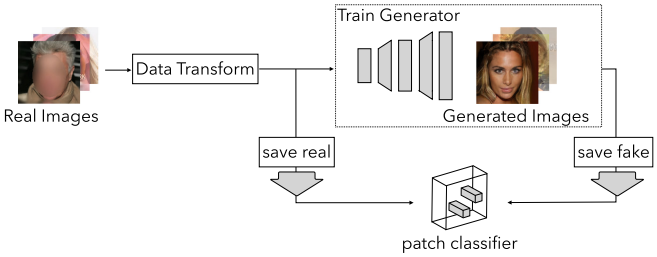

Fig. 2. To minimize the effect of image preprocessing artifacts - in which real and fake images undergo different preprocessing operations – we pass real images through the same data transform as used to train the generator. We then save the real and fake images using identical pipelines.

3.2 Dataset Preparation

Image Preprocessing. A challenge with fully generative images, such as those created by GANs, is that fake images can be saved with arbitrary codecs, e.g., we decide whether we want to save the image in JPG or PNG format. However, the set of real images is saved with a fixed codec when the original dataset is created. When training a classifier on real and fake images with subtly different preprocessing pipelines, the classifier can simply learn to detect the differences in preprocessing. If the test images also have this inconsistency, we would appear to obtain high accuracy on the test set, even though the classifier is really only detecting formatting artifacts. One way to mitigate this disparity is to apply data augmentation to reduce the effect of these differences [33,34].

We preprocess the images to make our real and fake dataset as similar as possible, in an effort to isolate fake image artifacts and minimize the possibility of learning differences in preprocessing. We create the "real" dataset by passing the real images through the generator's data loading pipeline (e.g. resizing) and saving the real images *after* this step in lossless PNG format (Fig. 2). We save the fake images in PNG format from the generator output, so the remaining differences between real and fake images are due to artifacts of the generator. We then resize all images to the same size using Lanczos interpolation before saving to file. Additional details are provided in Supplementary Material Sect. 2.1.

We take these precautions because any minor difference in preprocessing is easily learnt by the fake-image classifier and leads to an illusion of increased generalization capacity (for example, differences in the image codec leads to perfect average precision across various test datasets; see Supplementary Material Sect. 2.1). This approach allows us to focus on the inherent differences between real images and generated ones to minimize any potential confounders due to preprocessing. In the remainder of this section, we briefly detail the image generation and manipulation methods that we investigate in our experiments.

Fully Generative Models. The first class of models we consider are fully generative models which map a random sample from a known distribution (e.g. a

multivariate Gaussian) to an image. *Progressive GAN* (PGAN) [16] is one recent example, which uses a progressive training schedule to increase the output resolution of images during training. We use the publicly available PGAN model trained on the CelebA-HQ face dataset. We also train several other PGANs to various smaller resolutions and on the more diverse FFHQ face dataset. *StyleGAN* (SGAN) [17] introduces an alternative generator architecture which incorporates the latent code into intermediate layers of the generator, resulting in unsupervised disentanglement of high-level attributes, e.g., hair and skin tone. We use the public versions of StyleGAN on the CelebA-HQ and FFHQ datasets, and StyleGAN2 [18] on the FFHQ dataset. In additional to PGAN and SGAN, we also consider the *Glow* generator [19], a flow-based model using modified 1×1 invertible convolutions that optimizes directly for log-likelihood rather than adversarial loss. We use the public Glow generator trained on CelebA-HQ faces. Finally, we also include a face generator based on a *Gaussian Mixture Model* (GMM) rather than convolutional layers [31]; the GMM uses low-rank plus diagonal Gaussians to efficiently model covariance in high-dimensional outputs such as images. We train the GMM model on the CelebA [23] dataset using default parameters.

Facial Manipulation Models. We use the FaceForensics++ dataset [32], which includes methods for identity manipulation and expression transfer. Identity manipulation approaches, such as *FaceSwap*, paste a source face onto a target background; specifically, FaceSwap fits detected facial landmarks to 3D model and then projects the face onto the target scene. The deep learning analogue to FaceSwap is the *Deepfake* technique, which uses a pair of autoencoders with a shared encoder to swap the source and target faces. On the other hand, expression transfer maps the expression of a source actor onto the face of a target. *Face2Face* achieves this by tracking expression parameters of the face in a source video and applying them to a target sequence. *Neural Textures* uses deep networks to learn a texture map and a neural renderer to modify the expression of the target face.

3.3 Baseline Models

We train and evaluate full MesoInception4 [2], Resnet [13], and Xception [6] models on the same datasets that we use to train the truncated classifiers. Following [2], we train MesoInception4 using squared error loss. For the Resnet model and the Xception model, also used in [32], we train with standard two-class cross entropy loss. We train these models from scratch as they are not initially trained for this classification task. Finally, we also compare to a model trained to detect CNN artifacts via blurring and compression augmentations [33]. For this model, we finetune at a learning rate of 1e-6 using similar augmentation parameters as the original paper to improve its performance specifically on face datasets. We use the same stopping criteria based on validation accuracy for all baseline models as we use for the truncated models.

4 Experiments

4.1 Classification via Patches

Nowadays with access to public source code, it becomes easy for anyone to train their own image generators with slight modifications. We conduct two experiments to test generalization across simple changes in (1) generator size and (2) the weight initialization seed. In addition to the public 1024px PGAN, we train PGANs for 512, 256, and 128px resolutions on the CelebA-HQ dataset and sample images from each generator. We then train a classifier using only images from the 128px generator.

We test the classifier on generated images from the remaining resolutions, using average precision (AP) as a metric (Table 1; left). Here, the full-model baselines tend to perform worse on the unseen test resolutions compared to the truncated models. However, adding blur and JPEG augmentations in [33] helps to overcome the full-model limitations, likely hiding the resizing artifacts. Of the truncated models, the AP tends to decrease on the unseen test images as the receptive field increases, although there is a slight decline when the receptive field is too small with the Xception Block 1 model. On average across all resolutions, the Xception Block 2 model obtains highest AP.

Table 1. Average precision across PGANs trained to different resolutions or with different random initialization seeds. The classifier is trained on a fake images from a 128px GAN and real images at 128px resolution. AP on the test set corresponding to training images is colored in gray.

Model depth	Resolution				Model seed			
	128	256	512	1024	0	1	2	3
Resnet Layer 1	100.0	99.99	99.60	96.95	100.0	100.0	100.0	100.0
Xception Block 1	100.0	100.0	99.87	98.53	100.0	100.0	100.0	100.0
Xception Block 2	100.0	100.0	100.0	99.98	100.0	100.0	100.0	100.0
Xception Block 3	100.0	100.0	100.0	99.92	100.0	100.0	100.0	100.0
Xception Block 4	100.0	100.0	99.92	99.34	100.0	100.0	100.0	100.0
Xception Block 5	100.0	100.0	98.90	91.18	100.0	100.0	100.0	100.0
[2] MesoInception4	100.0	99.59	98.15	87.00	100.0	99.99	99.82	99.95
[13] Resnet-18	99.99	96.85	91.75	80.17	99.99	98.41	95.20	95.02
[6] Xception	100.0	99.94	99.84	97.28	100.0	100.0	99.99	100.0
[33] CNN (p = 0.1)	100.0	99.99	99.97	99.78	100.0	100.0	100.0	100.0
[33] CNN (p = 0.5)	100.0	100.0	99.99	99.83	100.0	100.0	100.0	100.0

Next, we train four PGANs to 128px resolution with different weight initialization seeds. We train the classifier using fake images drawn from one of the generators, and test on the remaining generators (Table 1; right). Surprisingly, even when the only difference between generators is the random seed, the full Resnet-18 model makes errors when classifying fake images generated by the three other GANs. This suggests that fake images generated by different PGANs differ slightly between the different initialization seeds (as also noted in [35]). The MesoInception4 and Xception architectures are more robust to model seed, and so is blur/JPG augmentation. The truncated models with reduced receptive field are also robust to model seed differences.

Table 2. Average precision on different model architectures and an alternative dataset (FFHQ). The classifier is trained on 1024px PGAN random samples and reprojected PGAN images on the CelebA-HQ dataset. For the Glow model (*) we observe better performance when classifier training does not include reprojected images for the truncated models; additional results in Supplementary Material Sect. 2.4. AP on the test set corresponding to training images is colored in gray.

Model	PGAN	Architectures			FFHQ dataset		
		SGAN	Glow*	GMM	PGAN	SGAN	SGAN2
Resnet Layer 1	100.0	97.22	72.80	80.69	99.81	72.91	71.81
Xception Block 1	100.0	98.68	95.48	76.21	99.68	81.35	77.40
Xception Block 2	100.0	99.99	67.49	**91.38**	**100.0**	90.12	90.85
Xception Block 3	100.0	**100.0**	74.98	80.96	100.0	92.91	**91.45**
Xception Block 4	100.0	99.99	66.79	42.82	100.0	**95.85**	90.62
Xception Block 5	100.0	100.0	60.44	48.92	100.0	93.09	89.08
[2] MesoInception4	100.0	97.90	49.72	45.98	98.71	80.57	71.27
[13] Resnet-18	100.0	64.80	47.06	54.69	79.20	51.15	52.37
[6] Xception	100.0	99.75	55.85	40.98	99.94	85.69	74.33
[33] CNN (p = 0.1)	100.0	98.41	90.46	50.65	99.95	90.48	85.27
[33] CNN (p = 0.5)	100.0	97.34	**97.32**	73.33	99.93	88.98	84.58

We then test the ability of patch classifiers to generalize to different generator architectures (Table 2; left). To create a training set of PGAN fake images, we combine two datasets – random samples from the generator, as well as images obtained by reprojecting the real images into the GAN following [3]. Intuitively, this reprojection step creates fake images generated by the GAN that are as close as possible to their corresponding real images, forcing the classifier to focus on the remaining differences (also see Supplementary Material Sects. 1.2 and 2.4). We then test the classifier on SGAN, Glow, and GMM face generators. We show additional results training on only PGAN fake samples, as well as only on reprojected images as the fake dataset in Supplementary Material Sect. 2.4 (on the Glow model, AP is substantially better when trained without the reprojected

images). Generalizing to the SGAN architecture is easiest, due to the many similarities between the PGAN and SGAN generators. With the exception of the Glow generator, the truncated models obtain higher AP compared to the larger classifiers with a fraction of the number of parameters.

Lastly, we test the classifiers' ability to generalize to a different face dataset (Table 2; right). Using the same classifiers trained on CelebA-HQ faces and PGAN samples and reprojections, we measure AP on real images from the FFHQ dataset and fake images from PGAN, SGAN, and SGAN2 trained on FFHQ faces. The truncated classifiers improve AP, particularly on the Style-based generators. The FFHQ dataset has greater diversity in faces than CelebA-HQ; however, small patches, such as hair, are likely similar between the two datasets. Using small receptive fields allows models to ignore global differences between images from different generators and datasets and focus on shared generator artifacts, perhaps explaining why truncated classifiers perform better than full models.

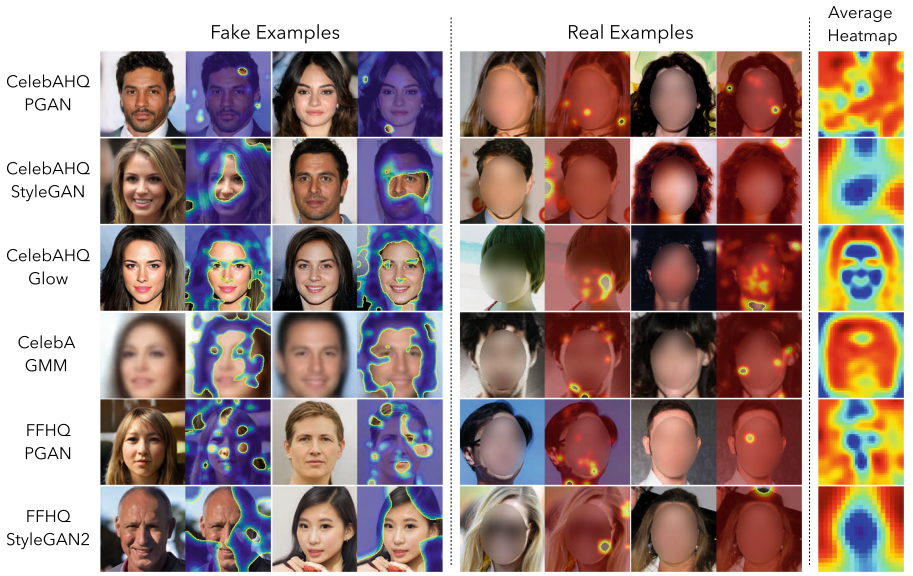

Fig. 3. Heatmaps based on the patch-wise predictions on real and fake examples from each dataset and fake image generator. We normalize all heatmaps between 0 and 1 and show fake values in blue and real values in red. We also show the average heatmap over the 100 easiest and fake examples, where red is most indicative of the correct class.

4.2 What Properties of Fake Images Generalize?

What artifacts do classifiers learn that allow them to detect fake images generated from different models? Since the patch-based classifiers output real-or-fake

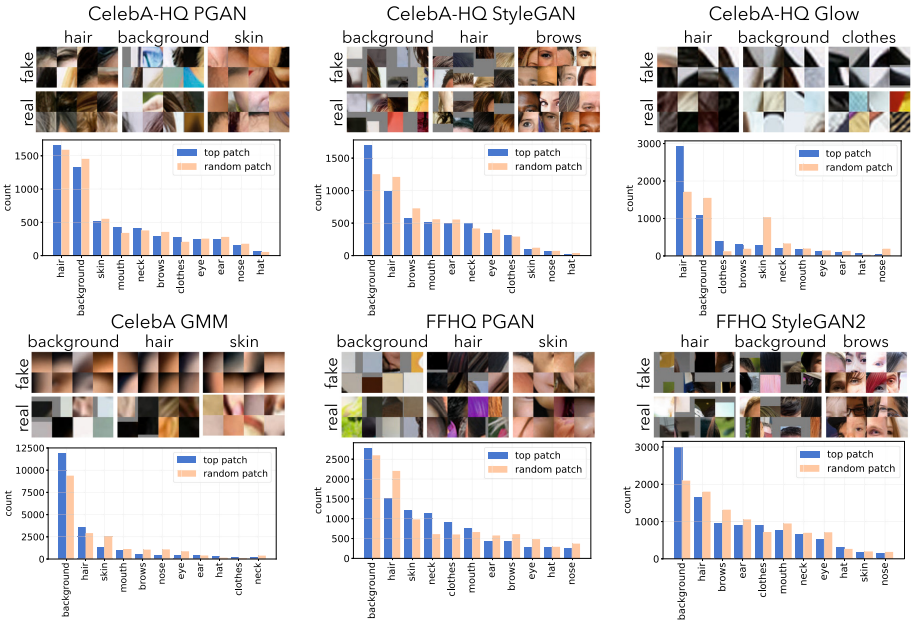

Fig. 4. We take a pretrained segmentation network to assign the most predictive patch in real and fake images to a semantic cluster. We find that the fake-image classifier (which was only trained on the CelebA-HQ dataset with PGAN fake images) relies on patches such as hair, background, clothing, and mouths to make decisions.

predictions over sliding patches of a query image, we use these patch-wise predictions to draw heatmaps over the images and visualize what parts of an image are predicted as more real or more fake (Fig. 3). Using the classifiers trained on CelebA-HQ PGAN images, we show examples of the prediction heatmaps for the other face generators and on the FFHQ dataset, using the best performing patch model for each column in Table 2. We also show an averaged heatmap over the 100 most real and most fake images, where the red areas indicate regions most indicative of the correct class (Fig. 3; right). The average heatmaps highlight predominately hair and background areas, indicating that these are the regions that patch-wise models rely on when classifying images from unseen test sources.

Next, we take a pretrained facial segmentation network to partition each image into semantic classes. For the most predictive patch in each image, we assign the patch to a cluster from the segmentation map, and plot the distribution of these semantic clusters (Fig. 4). We also sample a random patch in each image and assign it to a semantic cluster for comparison. Using the segmentation model, the predominant category of patches tends to be hair or background, with clothes, skin, or brows comprising the third-largest category. Qualitatively, many of the fake patches contain boundary edges such as those between hair and background or hair and skin, suggesting that creating a realistic boundary

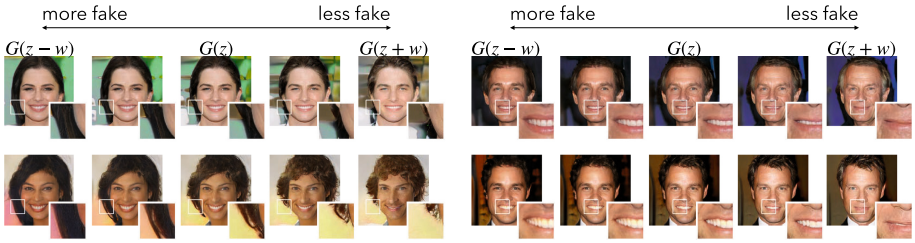

Fig. 5. We shift the latent space of the PGAN generator to exaggerate the fake features of an image, which accentuates the hair and smile of the fake images.

is difficult for image generators to imitate, whereas crisp boundaries naturally exist in real images.

To further understand what makes fake images look fake, we modify the latent space of the PGAN generator to accentuate the features that the classifier detects (Fig. 5). We parametrize a shift in latent space by a vector w, and optimize:

$$w^* = \arg\min_w \mathbb{E}_z \left[L_{\text{fake}}(G(z-w)) + L_p(G(z),\ G(z-w)) \right] \tag{3}$$

where L_{fake} refers to the classifier loss on fake images [10], and L_p is a perceptual loss regularizer [36] to ensure that the modified image does not deviate too far from the original. Applying this vector to latent space samples accentuates hair and smiling with teeth, which are both complex textures and likely difficult for generators to recreate perfectly (Fig. 5). By applying the shift in the opposite direction, $G(z+w)$, we see a reduction these textures, in effect minimizing the presence of textures that are more challenging for the generator to imitate.

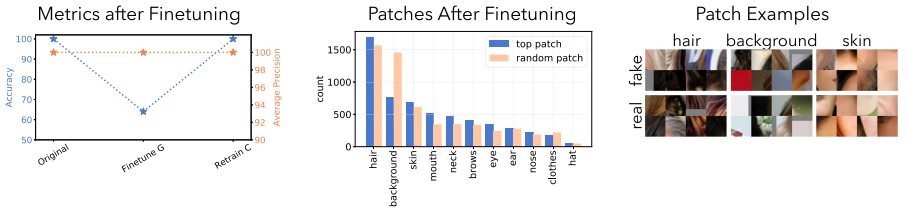

Fig. 6. We finetune the PGAN generator to evade detection by the fakeness classifier. When we subsequently train a new classifier, we find that the finetuned generator still has detectable artifacts, but now predominantely less in background patches.

4.3 Finetuning the Generator

With access to gradients from the classifier, an easy adversarial attack is to modify the generator to evade detection by the classifier. Will this now make the

previously identified fake patches undetectable? To investigate this, we finetune a PGAN to create adversarial fake samples that are classified as real. To ensure that the images remain realistic, we jointly optimize the classifier loss and GAN loss on the CelebA-HQ dataset:

$$\mathcal{L} = \min_G \max_D \left[\mathcal{L}_{\text{GAN}}(G, D) + \mathcal{L}_{\text{real}}(G, C) \right]; \tag{4}$$

i.e., we optimize both the generator and discriminator with the added constraint that the generator output should be predicted as real by the classifier C. Finetuning the generator does not drastically change the generated output (see Supplementary Material Sect. 2.7), but it decreases the classifier's accuracy from 100% to below 65% (Fig. 6). Using a variable threshold (AP) is less sensitive to this adversarial finetuning. We train a second classifier using images from the finetuned generator, which is able to recover in accuracy. We then compute the most predictive image patches for the retrained classifier and cluster them according to semantic category. Compared to the patches captured by the first classifier, this retrained classifier relies less on background patches and more on facial features, suggesting that artifacts in typically solid background patches are easiest for the generator to hide, while artifacts in more textured regions such as hair still remain detectable.

4.4 Facial Manipulation

Unlike the fully-generative scenario, facial manipulation methods blend content from two images, hence only a portion of the image is manipulated. Here, we train on each of the four FaceForensics++ datasets [32], and test generalization to the remaining three datasets (Table 3). We compare the effect of different receptive fields using truncated models, and investigate which patches are localized.

In these experiments, training on Face2Face images yields the best generalization to remaining datasets. On the other hand, generalization to FaceSwap images is the hardest – training on the other manipulation methods does not generalize well to FaceSwap images, and training on FaceSwap does not generalize well to other manipulation methods. Compared to the full-model baselines, we find that truncated patch classifiers tend to generalize when trained on the Face2Face or Deepfakes domains. Adding augmentations to training [33] can also boost results in some domains. While we do not use mask supervision during training, [21] notes that using this additional supervision signal improves generalization.

Next, we seek to investigate which patches are identified as predictive using the truncated classifiers in the facial manipulation setting. Unlike the fully generative scenario in which the classifiers tend to focus on the background, these classifiers trained on facial manipulation focus on the face region (without explicit

supervision of the face location). In particular, when trained on the Face2Face manipulation method, the classifiers use predominately the mouth region to classify Deepfakes and NeuralTextures manipulation, with eyes or nose as a secondary feature depending on the manipulation method (Fig. 7). We show additional visualizations in Supplementary Material.

Table 3. Average precision on FaceForensics++ [32] datasets. Each model is trained on one dataset and evaluated on the remaining datasets.

Model depth	Train on Deepfakes				Train on Neural Tex.			
	DF	NT	F2F	FS	DF	NT	F2F	FS
Resnet Layer 1	98.97	**74.99**	**71.74**	57.15	**70.32**	86.93	65.04	52.37
Xception Block 1	92.95	70.52	65.94	52.83	66.30	80.72	62.65	52.05
Xception Block 2	98.04	70.28	67.48	56.04	69.61	85.75	64.27	52.70
Xception Block 3	99.41	67.58	63.62	57.97	67.62	85.44	60.71	52.07
Xception Block 4	99.14	68.91	70.36	**58.74**	73.65	90.97	60.72	52.79
Xception Block 5	99.27	68.25	66.68	43.20	83.52	92.23	63.75	49.94
[2] MesoInception4	97.28	59.27	60.17	47.24	65.75	83.27	62.92	**54.03**
[13] Resnet-18	93.90	53.22	53.45	53.69	69.98	85.40	54.77	50.89
[32] Xception	98.60	60.15	56.84	46.12	70.07	93.61	56.79	48.55
[33] CNN (p = 0.1)	97.78	60.08	59.73	50.87	68.67	95.16	68.15	47.43
[33] CNN (p = 0.5)	98.16	54.02	56.06	55.99	66.98	95.03	**71.50**	51.93
Model depth	Train on Face2Face				Train on FaceSwap			
	DF	NT	F2F	FS	DF	NT	F2F	FS
Resnet Layer 1	**84.39**	79.72	97.66	60.53	59.49	52.56	62.00	97.13
Xception Block 1	77.65	**80.88**	93.84	61.62	53.14	49.24	56.89	82.89
Xception Block 2	84.04	79.51	97.40	63.21	58.39	51.65	61.73	92.58
Xception Block 3	76.10	74.77	97.33	63.10	**61.77**	53.44	61.34	96.06
Xception Block 4	67.18	61.72	97.19	63.04	61.33	52.02	59.45	96.56
Xception Block 5	81.25	61.91	96.45	55.15	57.14	47.39	54.68	95.57
[2] MesoInception4	67.53	55.17	92.27	54.06	50.64	48.87	56.15	93.81
[13] Resnet-18	55.43	52.57	93.27	53.39	61.03	51.66	52.56	91.49
[6] Xception	66.12	56.07	97.41	53.15	53.86	50.00	56.55	96.84
[33] CNN (p = 0.1)	65.76	64.81	98.40	59.48	59.19	**53.50**	**63.07**	99.02
[33] CNN (p = 0.5)	65.43	60.36	97.94	**63.52**	60.19	52.11	59.81	98.25

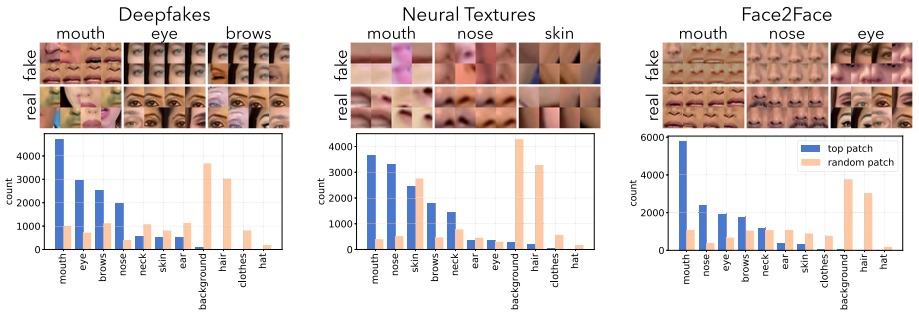

Fig. 7. Histograms of the most predictive patches from a classifier trained on Face2Face and un-manipulated images, and tested on the Neural Textures and Deepfakes manipulation methods. Unlike the fully-generative model setup, the classifier in this case localizes patches within the face.

5 Conclusion

Identifying differences between real and fake images is a constantly evolving problem and is highly sensitive to minor preprocessing details. Here, we take the approach of equalizing the preprocessing of the two classes of images to focus on the inherent differences between an image captured from a camera and a doctored image either generated entirely from a deep network, or partially manipulated in facial regions. We investigate using classifiers with limited receptive fields to focus on local artifacts, such as textures in hair, backgrounds, mouths, and eyes, rather than the global semantics of the image. Classifying these small patches allows us to generalize across different model training parameters, generator architectures, and datasets, and provides us with a heatmap to localize the potential areas of manipulation. We show a technique to exaggerate the detectable artifacts of the fake images, and demonstrate that image generators can still be imperfect in certain patches despite finetuning against a given classifier. While progress on detecting fake images inevitably creates a cat-and-mouse problem of using these results to create even better generators, we hope that understanding these detectors and visualizing what they look for can help people anticipate where manipulations may occur in a facial image and better navigate potentially falsified content in today's media.

Acknowledgements. We thank Antonio Torralba, Jonas Wulff, Jacob Huh, Tongzhou Wang, Harry Yang, and Richard Zhang for helpful discussions. This work was supported by a National Science Foundation Graduate Research Fellowship under Grant No. 1122374 to L.C. and DARPA XAI FA8750-18-C000-4 to D.B.

References

1. Deepfakes.https://github.com/deepfakes/faceswap
2. 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)
3. Bau, D., et al.: Seeing what a GAN cannot generate. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4502–4511 (2019)
4. Bayar, B., Stamm, M.C.: A deep learning approach to universal image manipulation detection using a new convolutional layer. In: Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security, pp. 5–10 (2016)
5. Carlini, N., Farid, H.: Evading deepfake-image detectors with white-and black-box attacks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 658–659 (2020)
6. 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)
7. Cozzolino, D., Poggi, G., Verdoliva, L.: Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In: Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security, pp. 159–164 (2017)
8. Cozzolino, D., Thies, J., Rössler, A., Riess, C., Nießner, M., Verdoliva, L.: ForensicTransfer: weakly-supervised domain adaptation for forgery detection. arXiv preprint arXiv:1812.02510 (2018)
9. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231 (2018)
10. Goetschalckx, L., Andonian, A., Oliva, A., Isola, P.: GANalyze: toward visual definitions of cognitive image properties. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5744–5753 (2019)
11. Gragnaniello, D., Marra, F., Poggi, G., Verdoliva, L.: Analysis of adversarial attacks against CNN-based image forgery detectors. In: 2018 26th European Signal Processing Conference (EUSIPCO), pp. 967–971. IEEE (2018)
12. Hays, J., Efros, A.A.: Scene completion using millions of photographs. ACM Trans. Graph. (TOG) **26**(3), 4-es (2007)
13. 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)
14. Huh, M., Liu, A., Owens, A., Efros, A.A.: Fighting fake news: image splice detection via learned self-consistency. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11215, pp. 106–124. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01252-6_7
15. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1125–1134 (2017)

16. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
17. 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)
18. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. arXiv preprint arXiv:1912.04958 (2019)
19. Kingma, D.P., Dhariwal, P.: Glow: generative flow with invertible 1x1 convolutions. In: Advances in Neural Information Processing Systems, pp. 10215–10224 (2018)
20. Li, C., Wand, M.: Precomputed real-time texture synthesis with Markovian generative adversarial networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 702–716. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_43
21. 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)
22. Li, Y., Lyu, S.: Exposing deepfake videos by detecting face warping artifacts. arXiv preprint arXiv:1811.00656 (2018)
23. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV), December 2015
24. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
25. Mayer, O., Stamm, M.C.: Exposing fake images with forensic similarity graphs. arXiv preprint arXiv:1912.02861 (2019)
26. Mo, H., Chen, B., Luo, W.: Fake faces identification via convolutional neural network. In: Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security, pp. 43–47 (2018)
27. Popescu, A.C., Farid, H.: Exposing digital forgeries by detecting duplicated image regions. Dept. Comput. Sci., Dartmouth College, Technical report TR2004-515 pp. 1–11 (2004)
28. Popescu, A.C., Farid, H.: Exposing digital forgeries by detecting traces of resampling. IEEE Trans. Signal Process. **53**(2), 758–767 (2005)
29. Popescu, A.C., Farid, H.: Exposing digital forgeries in color filter array interpolated images. IEEE Trans. Signal Process. **53**(10), 3948–3959 (2005)
30. Rahmouni, N., Nozick, V., Yamagishi, J., Echizen, I.: Distinguishing computer graphics from natural images using convolution neural networks. In: 2017 IEEE Workshop on Information Forensics and Security (WIFS), pp. 1–6. IEEE (2017)
31. Richardson, E., Weiss, Y.: On GANs and GMMs. In: Advances in Neural Information Processing Systems, pp. 5847–5858 (2018)
32. 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)
33. 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)
34. Xuan, X., Peng, B., Wang, W., Dong, J.: On the generalization of GAN image forensics. In: Sun, Z., He, R., Feng, J., Shan, S., Guo, Z. (eds.) CCBR 2019. LNCS, vol. 11818, pp. 134–141. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31456-9_15

35. Yu, N., Davis, L.S., Fritz, M.: Attributing fake images to GANs: learning and analyzing GAN fingerprints. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 7556–7566 (2019)
36. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 586–595 (2018)
37. Zhang, X., Karaman, S., Chang, S.F.: Detecting and simulating artifacts in GAN fake images. arXiv preprint arXiv:1907.06515 (2019)
38. Zhou, P., Han, X., Morariu, V.I., Davis, L.S.: Two-stream neural networks for tampered face detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1831–1839. IEEE (2017)