Leveraging Frequency Analysis for Deep Fake Image Recognition

Joel Frank [1] **Thorsten Eisenhofer** [1] **Lea Schönherr** [1] **Asja Fischer** [1] **Dorothea Kolossa** [1] **Thorsten Holz** [1]

Abstract

Deep neural networks can generate images that are astonishingly realistic, so much so that it is often hard for humans to distinguish them from actual photos. These achievements have been largely made possible by Generative Adversarial Networks (GANs). While *deep fake* images have been thoroughly investigated in the image domain—a classical approach from the area of image forensics—an analysis in the *frequency domain* has been missing so far. In this paper, we address this shortcoming and our results reveal that in frequency space, GAN-generated images exhibit severe artifacts that can be easily identified. We perform a comprehensive analysis, showing that these artifacts are consistent across different neural network architectures, data sets, and resolutions. In a further investigation, we demonstrate that these artifacts are caused by upsampling operations found in all current GAN architectures, indicating a structural and fundamental problem in the way images are generated via GANs. Based on this analysis, we demonstrate how the frequency representation can be used to identify deep fake images in an automated way, surpassing state-of-the-art methods.

1. Introduction

GANs produce sample outputs—images, audio signals, or complete video sequences—that are astonishingly effective at fooling humans into believing their veracity (Fried et al., 2019; Karras et al., 2019; Kumar et al., 2019; Song et al., 2020). The difficulty of distinguishing these so-called *deep fakes* from real media is for example demonstrated at *whichfaceisreal.com* (West & Bergstrom, 2019), a website on which a user can see two different images: one from the Flicker-Faces-HQ data set and one generated by StyleGAN (Karras et al., 2019). The task is to decide which of these two images is real. Even though humans generally do better than random guessing, players' performance reportedly peaked at around 75% accuracy (Simonite, 2019).

At a time where fake news have become a practical problem and Internet information campaigns might have influenced democratic processes (Thompson & Lapowsky, 2017), developing automated detection methods is a crucial task. A worrying reminder is the example of Gabon's president Ali Bongo: In late 2018, the president fell ill, not appearing in public for months. As the public grew weary, the government released a video of the president, only to be immediately labeled as a deep fake. Albeit never to be confirmed as such, one week later the military launched an unsuccessful coup, citing the video as part of the motivation (Hao, 2019).

Previous research on detecting GAN-generated images has either utilized large, complex convolutional networks directly trained in the image domain (Mo et al., 2018; Yu et al., 2019a; Tariq et al., 2019) or used hand-crafted features from the frequency domain (Marra et al., 2019; Valle et al., 2018). In contrast, we provide in this paper a comprehensive analysis of the frequency spectrum across multiple different GAN architectures and data sets. The surprising finding is that *all* GAN architectures exhibit severe artifacts in the frequency domain. Speculating that these artifacts stem from upsampling operations, we experiment with different upsampling techniques and identify patterns which are consistent with our earlier observations.

Based on these insights, we demonstrate that the frequency domain can be utilized for (i) efficiently separating real from fake images, as well as, (ii) identifying by which specific GAN a sample was generated. In the first case, we demonstrate that the artifacts are so severe that a linear separation of the data is possible in the frequency space. In the second case, we demonstrate that we can achieve higher accuracy, while simultaneously utilizing significantly less complex models, than state-of-the-art approaches (Yu et al., 2019a) (using roughly 1.9% of their parameters). Additionally, we demonstrate that classifiers trained in the frequency domain are more robust against common image perturbations (e.g., blurring or cropping). The code to reproduce our experi-

[1]Ruhr-University Bochum, Horst Görtz Institute for IT-Security, Bochum, Germany. Correspondence to: Joel Frank <joel.frank@rub.de>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

ments and plots as well as all pre-trained models are available online at github.com/RUB-SysSec/GANDCTAnalysis.

In summary, our key contributions are as follows:

- We perform a comprehensive frequency-domain analysis of images generated by various popular GANs, revealing severe artifacts common across different neural network architectures, data sets, and resolutions.
- We show in several experiments that these artifacts arise from upsampling operations employed in all current GAN architectures, hinting towards a structural problem on how generative neural networks that map from a low-dimensional latent space to a higher-dimensional input space are constructed.
- We demonstrate the effectiveness of employing frequency representations for detecting GAN-generated deep fake images by an empirical comparison against state-of-the-art approaches. More specifically, we show that frequency-representation-based classifiers yield higher accuracy, while simultaneously needing significantly fewer parameters. Additionally, these classifiers are more robust to common image perturbations.

2. Related Work

In the following, we present an overview of related work and discuss the connections to our approach.

Generative Adversarial Networks GANs (Goodfellow et al., 2014) have essentially established a new sub-field of modern machine learning research. As generative models, they aim at estimating the probability density distribution underlying the training data. Instead of employing standard approaches like likelihood maximization for training, they are based on the idea of defining a game between two competing models (usually neural networks): a generator and a classifier (also called discriminator). The generator is tasked with producing samples that look like training data, while the discriminator attempts to distinguish real from fake (i. e., generated) samples. These tasks can be translated into a min-max problem: a joint objective which is minimized w.r.t. the parameters of the generator and maximized w.r.t. the parameters of the discriminator.

Image Synthesis While there exists a huge variety of models for image generation (e.g. see van den Oord et al., 2017; Razavi et al., 2019), we will focus on images generated by GANs. The earliest breakthrough in generating images with GANs was the switch to Convolutional Neural Network (CNN) (Radford et al., 2016). While this might seem trivial today, it allowed GANs to outperform similar image synthesis methods at the time. In follow-up work, GAN research yielded a steady stream of innovations, which pushed the state-of-the-art further: training with labeled data (Mirza & Osindero, 2014; Salimans et al., 2016), utilizing the Wasserstein distance (Arjovsky et al., 2017; Gulrajani et al., 2017; Petzka et al., 2018), spectral normalization (Miyato et al., 2018), progressive growing (Karras et al., 2018) or style mixing (Karras et al., 2019), and employing very large models (Brock et al., 2019), just to name a few examples.

Image Forensics Traditional image forensics uses the natural statistics of images to detect tampered media (Fridrich, 2009; Lyu, 2013). A promising approach is steganalysis (Lukáš et al., 2006; Fridrich, 2009; Bestagini et al., 2013), where high-frequency residuals are used to detect manipulations. These traditional methods have recently been expanded by CNN-based methods (Bayar & Stamm, 2016; Bappy et al., 2017; Cozzolino et al., 2017; Zhou et al., 2018), which learn a more complex feature representation, improving the state-of-the-art for tampered media detection.

Prior work has utilized these findings for identifying GAN-generated images: Marra et al. (2018) provide a comparison of different steganalysis and CNN-based methods, several approaches use CNNs in the image domain (Mo et al., 2018; Yu et al., 2019a; Tariq et al., 2019), others use statistics in the image domain (McCloskey & Albright, 2018; Nataraj et al., 2019). Another group of systems employs handcrafted features from the frequency domain, namely, steganalysis-based features (Marra et al., 2019) and spectral centroids (Valle et al., 2018). In contrast, our method explores the entire frequency spectrum and we link our detection capabilities to fundamental shortcomings in the construction of modern generative neural networks.

Concurrently and independently to our research, both Wang et al. (2020) and Durall et al. (2020) made similar observations: Wang et al. demonstrate that a deep fake classifier trained with careful data augmentation on the images of only one specific CNN generator is able to generalize to unseen architectures, data sets, and training methods. They suggest that their findings hint at systematic flaws in today's CNN-generated images, preventing them from achieving realistic image synthesis. Durall et al. show that current CNN-based generative models fail to reproduce spectral distributions. They utilize the discrete Fourier Transform to analyze generated images and propose a spectral regularization term to tackle these issues.

3. Frequency Artifacts

While early GAN-generated images were easily distinguishable from real images, newer generations fool even human observers (Simonite, 2019). To facilitate the development of automated methods for recognizing fake images, we take

FFHQ Spectrum FFHQ StyleGAN StyleGAN Spectrum

Figure 1: **A side-by-side comparison of real and generated faces in image and frequency domain.** The left side shows an example and the mean DCT spectrum of the FFHQ data set. The right side shows an example and the mean DCT spectrum of a data set sampled from StyleGAN trained on FFHQ. We plot the mean by averaging over 10,000 images.

inspiration from traditional image forensics (Lyu, 2013) and examine GAN-generated images in the frequency domain.

3.1. Preliminaries

We transform images into the frequency domain using the discrete cosine transform (DCT). The DCT expresses, much like the discrete Fourier transform (DFT), a finite sequence of data points as a sum of cosine functions oscillating at different frequencies. The DCT is commonly used in image processing due to its excellent energy compaction properties and its separability, which allows for efficient implementations. Together with a circular convolution-multiplication relationship (Wen-Hsiung Chen et al., 1977), it enables fast filtering. We use the type-II 2D-DCT, which is, for example, also used in JPEG compression (Fridrich, 2009).

More formally, let an input image be given by the matrix[1] $I \in \mathbb{R}^{N_1 \times N_2}$, where the entries (specifying the pixel values) are denoted by $I_{x,y}$, and its DCT-transformed representation by the matrix $D \in \mathbb{R}^{N_1 \times N_2}$. The 2D-DCT is given by a function $\mathcal{D} : \mathbb{R}^{N_1 \times N_2} \to \mathbb{R}^{N_1 \times N_2}$ that maps an image $I = \{I_{x,y}\}$ to its frequency representation $D = \{D_{k_x,k_y}\}$, with

$$D_{k_x,k_y} = w(k_x)w(k_y) \sum_{x=0}^{N_1-1} \sum_{y=0}^{N_2-1} I_{x,y} \cos\left[\frac{\pi}{N_1}\left(x+\frac{1}{2}\right)k_x\right] \cos\left[\frac{\pi}{N_2}\left(y+\frac{1}{2}\right)k_y\right],$$

for $\forall k_x = 0, 1, 2, \ldots, N_1 - 1$ and $\forall k_y = 0, 1, 2, \ldots, N_2 - 1$, and where $w(0) = \sqrt{\frac{1}{4N}}$ and $w(k) = \sqrt{\frac{1}{2N}}$ for $k > 0$.

When we plot the DCT spectrum, we depict the DCT coefficients as a heatmap. Intuitively, the magnitude of each coefficient is a measure of how much the corresponding spatial frequency contributed to the overall image. The horizontal direction corresponds to frequencies in the x direction, while the vertical direction corresponds to frequencies in the y direction. In practice, we compute the 2D-DCT as a product of two 1D-DCTs, i.e., for images we first compute a DCT along the columns and then a DCT along the rows. This results in the top left corner of the heatmap corresponding to low frequencies (k_x and k_y close to zero), while the right bottom corner corresponds to high frequencies (k_x and k_y close to $N_1 - 1$ and $N_2 - 1$, respectively). Due to the energy compaction property of the DCT, the coefficients drop very quickly in magnitude when moving to high frequencies, thus, we log-scale the coefficients before plotting. All plots are computed for gray-scale images, produced using standard gray-scale transformations (i.e., a weighted average over the color channels). We also computed statistics for each color channel separately, which are consistent with the findings for gray-scale images. These can be found in the supplementary material, where we also provide plots of the absolute difference between the spectra of real and fake images.

[1] For simplicity, we omit the color channels and treat images as matrices, not as tensors.

3.2. Investigating Generated Images in the Frequency Domain

We start with examining our introductory example, i.e., images from the website *whichfaceisreal.com*. The images are either from the Flickr-Faces-HQ (FFHQ) data set or from a set generated by StyleGAN (Karras et al., 2019). In Figure 1, we visualize the frequency statistics of the data set by plotting the means of the corresponding spectra over the sets. As a reference, we include a sample from each data set. In the image domain, both samples look similar, however, in the frequency domain, one can easily spot multiple clearly visible artifacts for the generated images.

The spectrum of the FFHQ images represents a regular spectrum of DCT coefficients. Multiple studies (e.g. see Burton & Moorhead, 1987; Tolhurst et al., 1992; Field, 1987; 1999) have observed that the average power spectra of natural images tend to follow a $\frac{1}{f^\alpha}$ curve, where f is the frequency along a given axis and $\alpha \approx 2$ (see Figure 2 (a) from Torralba & Oliva, 2003). The low frequencies (in

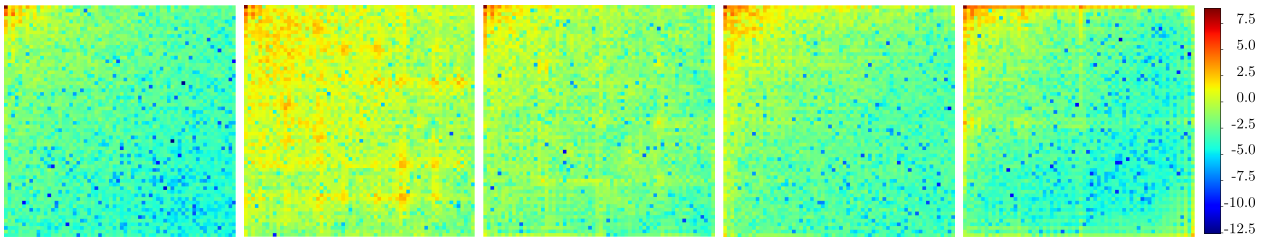


Figure 2: **The spectra of images generated by different neural networks trained on the Stanford dog data set.** The left-most heatmap depicts the mean spectrum of the Stanford dog data set. The rest depicts the mean spectra of images generated by different GANs. We plot the mean of the DCT spectra by averaging over 10,000 images.

the upper left corner of the heatmap) contribute the most to the image, and the contribution is gradually decreasing as we approach the higher frequencies (lower right corner). Intuitively, if a group of neighboring pixels contains similar values, i.e., they form an isochromatic area in the image, one can approximate those with a sum of low frequency functions. However, if there is a sudden change in the values, e.g., corresponding to an edge in the images, one has to use higher frequency functions to achieve a good approximation. Therefore, since most pixels in images are correlated to each other, i.e., colors mostly change gradually, large parts of the image can be approximated well by using low-frequency functions.

The images generate by StyleGAN, however, exhibit artifacts throughout the spectrum. In comparison to the spectra of natural images, StyleGAN-generated images contain strong high frequencies components (visible as high values in the lower right corner), as well as generally higher magnitudes throughout the spectrum. Especially notably is the grid-like pattern scattered throughout, also clearly noticeable in the top right (highest frequencies in x direction) and lower left corner (highest frequencies in y direction).

To analyze if this pattern in the spectral domain is a common occurrence for different GAN types and implementations, or simply a fault specific to the StyleGAN instance we studied, we selected four different architectures, namely BigGAN (Brock et al., 2019), ProGAN (Karras et al., 2018), StyleGAN (Karras et al., 2019), and SN-DCGAN (Miyato et al., 2018). Note that all of them were under the top-ten entries in the recent Kaggle competition on generating images of dogs (Kaggle, 2019). In this competition, the participants were required to upload 10,000 samples from their respective GAN instance. For each architecture, we downloaded and analyzed the corresponding samples and the training data (Khosla et al., 2011).

We show the mean of the spectra of the samples of each GAN and the training data in Figure 2. As discussed, statistics of natural images have been found to follow particular regularities. Like natural images, the DCT coefficients of the GAN-generated images decrease rapidly towards high frequencies. However, the spectral images often show a grid-like pattern. StyleGAN seems to better approximate spectra of natural images than the other GANs, but still contains high coefficients along the upper and left side of their spectra. These findings indicate a structural problem of the way GANs generate images that we explore further.

3.3. Upsampling

We hypothesize that the artifacts found for GAN-generated images in the frequency domain stem from their employed upsampling operations. We make this more concrete in the following: The generator of a GAN forms a mapping from a low-dimensional latent space to the higher-dimensional data space. In practice, the dimensionality of the latent space is much lower than the dimensionality of the data space, e.g., the generator of the StyleGAN-instance which generated the images presented in Figure 1 defines a mapping $G : \mathbb{R}^{100} \rightarrow \mathbb{R}^{1024\times1024}$. In typical GAN architectures, the latent vector gets successively upsampled until it reaches the final output dimension.

Previous work has already linked upsampling operations to causing grid-like patterns in the image domain (Odena et al., 2016). Recognizing this, the architecture of both the generator-network and the discriminator-network shifted from using strided transposed convolution (e.g., employed in DCGAN (Radford et al., 2016), Cramer-GAN (Bellemare et al., 2017), CycleGAN (Zhu et al., 2017), MMDGAN (Bińkowski et al., 2018), and SN-DCGAN (Miyato et al., 2018)) to using traditional upsampling methods—like nearest neighbor or bilinear upsampling (Gonzalez & Woods, 1992)—followed by a convolutional layer (e.g., employed in ProGAN (Karras et al., 2018), BigGAN (Brock et al., 2019), and StyleGAN (Karras et al., 2019)). While these changes addressed the problem in the image domain, our results show that the artifacts are still detectable in the frequency domain.

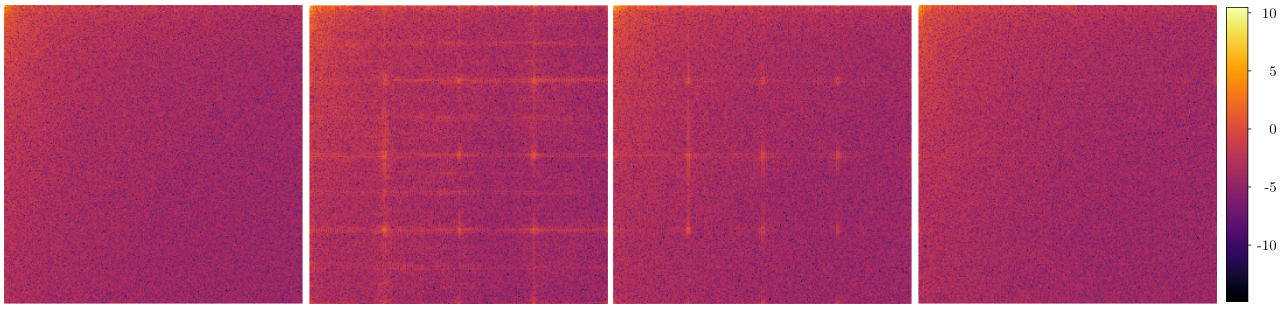


Figure 3: **The frequency spectrum resulting from different upsampling techniques.** We plot the mean of the DCT spectrum. We estimate $\mathbb{E}[\mathcal{D}(I)]$ by averaging over 10,000 images sampled from the corresponding network or the training data.

Moreover, both upsampling and downsampling operations have recently been linked to compromising shift invariance in neural networks, i.e., they cause classifier predictions to vary dramatically due to a simple one-pixel shift in the input image (Azulay & Weiss, 2018). Recently, Zhang (2019) proposed to use low-pass filtering after convolution and pooling layers to mitigate some of these effects.

We investigate how different upsampling strategies affect the DCT spectrum. Typically, we want to double the dimensionality of an image. When doing so, we have to fill in the blanks. However, this is known to cause artifacts. Thus, a range of different techniques have been invented throughout the past years to minimize these detrimental effects (Gonzalez & Woods, 1992). In our analysis, we investigate the effect of three different upsampling techniques:

- ***Nearest Neighbor***: The missing pixels in the upsampled image are approximated by copying the nearest pixel (nearest-neighbor upsampling; for a visual representation see the work of Odena et al., 2016).
- ***Bilinear***: Similar to nearest neighbor, but *after* copying the pixel values, the upsampled image is convolved with an anti-aliasing kernel (the filter $[1, 2, 1]$)[2]. This strategy is employed in the original StyleGAN.
- ***Binomial***: We follow Zhang (2019) and test the Binomial-5 kernel (i. e., the application of the filter $[1, 4, 6, 4, 1]$) as a replacement for the bilinear kernel.

We trained three different versions of StyleGAN on the LSUN bedrooms (Yu et al., 2015) dataset: in one, we kept the standard bilinear upsampling strategy, and in two, we replaced it by nearest-neighbor upsampling and binomial upsampling, respectively. We train at a resolution of 256×256, using the standard model and settings provided in the StyleGAN repository (Karras et al., 2019).

[2]Note that for brevity, we list the 1D-variant of the anti-aliasing kernel; in practice, we generate the 2D-variant as the outer product: $m\, m^T$, where m is the corresponding kernel.

The results are shown in Figure 3. As expected, with more elaborated upsampling techniques and with a larger size of the employed kernel, the spectral images become smoother and the artifacts less severe. These findings are in line with our hypothesis that the artifacts are caused by upsampling operations.

4. Frequency-Based Deep-Fake Recognition

In the following, we describe our experiments to demonstrate the effectiveness of investigating the frequency domain for differentiating GAN-generated images. In a first experiment, we show that DCT-transformed images are fully linearly separable, while classification on raw pixels requires non-linear models. Further, we verify that our model utilizes the artifacts discovered in Section 3.2. Then, we recreate the experiments by Yu et al. (2019a) and show how we can utilize the frequency domain to match generated images to their underlying architecture, demonstrating that we can achieve higher accuracy while utilizing fewer parameters in our models. Finally, we investigate how our classifier behaves when confronted with common image perturbations.

All experiments in this chapter were performed on a server running Ubuntu 18.04, with 192 GB RAM, an Intel Xeon Gold 6230, and four Nvidia Quadro RTX 5000.

4.1. Detecting Fake Images

First, we want to demonstrate that using frequency information allows to efficiently separate real from fake images. We consider our introductory example, i. e., aim at distinguishing real images from the FFHQ data set and fake images generated by StyleGAN. As discussed in Section 3, in the frequency domain, the images show severe artifacts. These artifacts makes it easy to use a simple linear classifier. To demonstrate this, we perform a ridge regression on real and generated images, after applying a DCT. For compari-

Table 1: **Ridge regression performed on FFHQ data set.** We report the accuracy on the test set. We also report the gain in accuracy when training in the frequency domain instead of using raw pixels. Best score is highlighted in **bold**.

Method	Accuracy	Gain
Ridge-Regression-Pixel	75.78 %	
Ridge-Regression-DCT	**100.00** %	+ 24.22 %

son, we also perform ridge regression on the original data representation.

Experiment Setup We sample 16,000 images from both the training data and the generator of the StyleGAN, respectively. We split each set into 10,000 training, 1,000 validation and 5,000 test images, resulting in a training set of 20,000, a validation set of 2,000, and a test set of 10,000 samples. For training a model on samples in the image domain, we normalize the pixel values to the range $[-1, 1]$. In the frequency domain, we first convert the images using DCT, then log-scale the coefficients and, finally, normalize them by removing the mean and scaling to unit variance. We optimize the linear regression models using the Adam optimizer (Kingma & Ba, 2015) with an initial learning rate of 0.001, minimizing the binary cross-entropy with l_2 regularization. We select the regularization factor λ via grid search from $\lambda \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ on the validation data, picking the one with the best score.

Results The results of our experiments are listed in Table 1, where we report the classification accuracy on the test set. As depicted in Figure 1, StyleGAN generates convincing face images, which are able to fool humans. However, they seem to exhibit consistent patterns, since even the simple linear classifier reaches a non-trivial accuracy in the image domain. In the frequency domain, however, we can perfectly separate the data set, with a classification accuracy of $100\,\%$ on the test set.

4.2. Different Upsampling Techniques

We want to investigate if more elaborate upsampling techniques can thwart the detection. To this end, we examine the different StyleGAN instances described in Section 3.3 (i.e., those who utilize nearest neighbour/bilinear/binomial upsampling). Again, we train a ridge-regression on both the raw pixel values as well as DCT coefficients. The experiment setup is the same as in the experiment described in Section 4.1

Results In Table 2, we report the classification accuracy of the test set. As one would intuitively suspect, heavier anti-aliasing reduces the accuracy of the classifier. Interestingly, this also seems to remove the consistent patterns exploited in the image domain. Note that these samples are generated at a much lower resolution (256×256) than the ones examined in Section 4.1 (1024×1024). Thus, they require less upsampling operations and exhibit less artifacts, making a full linear separation impossible. However, we can still reach $100\,\%$ test accuracy on the different data sets when we utilize a non-linear method like the CNN used in Section 4.3.

The small accuracy drop for the images generated by StyleGAN using binomial upsampling in comparison to the one using bilinear upsampling is surprising. When we examine the corresponding mean spectrum (Figure 3), it contains less artifacts obvious to humans. To verify that the classifier indeed utilizes the upsampling artifacts for its decision, we perform an additional experiment: we train a logistic regression classifier with l_1 penalty (LASSO) on the data sets. We then extract the corresponding weight vector and map it back to the corresponding frequencies. Since the l_1 penalty forces weights to zero which play a minor role in the classification, the remaining weights show that the high coefficients correspond to the frequencies which impact the decision the most. The results are depicted in Figure 4 and reveal that the binomial classifier still utilizes a grid-like structure to separate these images. Additionally, we present the absolute difference of the mean spectra with respect to the training data in the supplementary material.

4.3. Source Identification

The experiments described in this section are based on Yu et al. (2019a)'s approach, and investigate how precisely a generated image's underlying architecture can be classified: Yu et. al. trained four different GANs (ProGAN (Karras et al., 2018), SN-DCGAN (Miyato et al., 2018), CramerGAN (Bellemare et al., 2017), and MMDGAN (Bińkowski et al., 2018)) on the CelebA (Liu et al., 2015) and LSUN bedrooms dataset (Yu et al., 2015) and generated images from all models. Then, based on these images, a classifier was trained that assigned an image to the corresponding subset class it belongs to, i.e., either being real, or being generated by ProGAN, SN-DCGAN, CramerGAN, or MMDGAN.

Experimental Setup The experiments are conducted on images of resolution 128×128. We converted both data sets (i. e., celebA and LSUN bedrooms) specified by Yu et al. (2019b). For each data set, we utilize their pre-trained models to sample 150,000 images from each GAN, and take another 150,000 real images randomly sampled from the underlying training data set. We then partition these samples into 100,000 training, 20,000 validation and 30,000 test images, resulting in a combined set of 500,000 training, 100,000 validation and 150,000 test images. We analyze the performance of different classifiers trained both on the images in their original representation (i. e., raw pixels) and af-

Table 2: **Ridge regression performed on data samples generated by StyleGAN for different upsampling techniques**. More elaborate upsampling techniques seem to remove artifacts in the image domain.

Method	Nearest Neighbor	Gain	Bilinear	Gain	Binomial	Gain
Ridge-Regression-Pixel	74.77 %		62.13 %		52.64 %	
Ridge-Regression-DCT	**98.24** %	+ 23.47 %	**85.96** %	+ 23.83 %	**84.20** %	+ 31.56 %

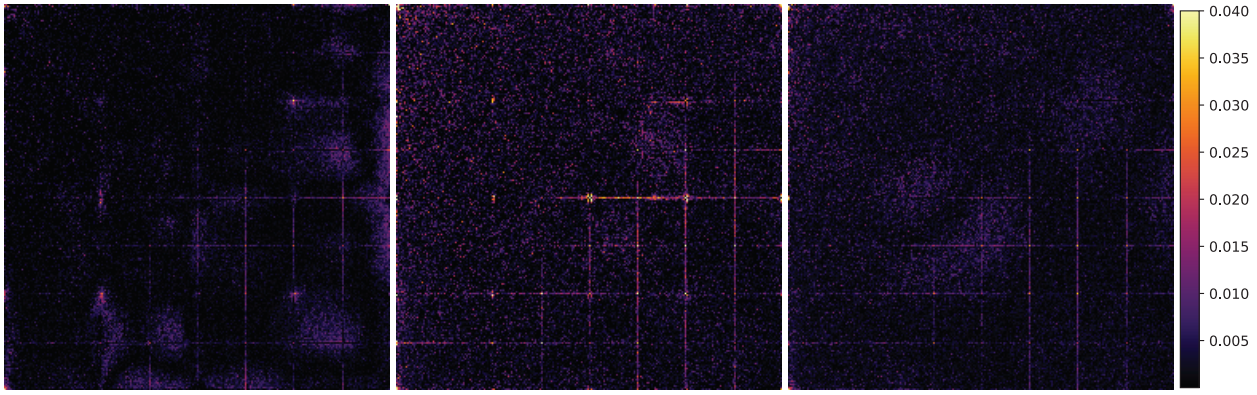


Figure 4: **A heatmap of which frequencies are used for classification.** We extracted the weight vector of the regression classifier trained on the different upsampling techniques. Then, we mapped it back to the corresponding frequencies. We plot the absolute value of the individual weights and clip their maximum value to 0.04 for better visibility.

ter applying DCT: K-nearest-neighbor, Eigenfaces (Sirovich & Kirby, 1987)), a CNN-based classifier developed by Yu et al. (2019a), and a steganalysis method based on photo-response non-uniformity (PRNU) patterns by Marra et al. (2019). These patterns also build on frequency information and utilizes high-pass filtered images to extract residuals information common to specific generators.

Moreover, we trained a shallow CNN[3], with only four convolution layers, to demonstrate that frequency information can significantly reduce the needed computation resources. Details on the architecture can be found in the supplementary material. Yu et. al. used a very large CNN with roughly 9 million parameters. In contrast, our CNN only utilizes around 170,000 parameters ($\sim 1.9\,\%$). During training, we utilize the validation set for hyperparameter tuning and employ early stopping We trained on log-scaled and normalized DCT coefficients. For raw pixel, we scaled the values to the range $[-1, 1]$, except for the PRNU-based method, which operates directly on image data.

For training our CNN, we use the Adam optimizer with an initial learning rate of 0.001 and a batch size of 1024, minimizing the cross-entropy loss of the model. For Yu et. al.'s CNN, we used the parameters specified in their repository (Yu et al., 2019b). We train their network for 8,000 mini-batch steps. During first tests, we discovered their network usually converges at around 4,000 steps. Hence, we doubled the number of steps for the final evaluation and picked the best performing instance (measured on the validation set). We also trained a variant of their classifier on DCT-transformed images. This version usually converges around step 300, however, we are conservative and train for 1,000 steps.

For the PRNU classifier, we utilize the implementation of Bondi & Bonettini (2019). We perform a grid search for the wavelet decomposition level $l \in \{1, 2, 3, 4\}$ and the estimated noise power $\sigma \in \{0.05, \dots, 1\}$ with step size 0.05. For the Eigenfaces-based classifier, we use Principal Component Analysis (PCA) to reduce the dimensionality to a variance threshold v and train a linear Support Vector Machine (SVM) on the transformed training data. We select the variance threshold $v \in \{0.25, 0.5, 0.95\}$, the SVMs regularization parameter $C \in \{0.0001, 0.001, 0.01, 0.1\}$, and the number of considered neighbors $k \in \{1, 2^\kappa + 1\}$ with $\kappa \in \{1, \dots, 10\}$, for the kNN classifier, via grid search over the validation set.

As the PRNU fingerprint algorithm does not require a large amount of training data (Marra et al., 2019) and scaling more traditional methods to such large data sets is notoriously hard (i. e., kNN and Eigenfaces), we use a subset of 100,000

[3]Using a CNN enabled a direct comparison to the performance of an analogous model on the raw-pixel input, where a CNN is the model of choice. In preliminary experiments, we also tried utilizing simple feed forwad NNs, however, only a CNN was able to fully separate the data.

Table 3: **The results of the source identification.** We report the test set accuracy, the gain in the frequency domain and highlight the best score in **bold**.

Method	LSUN	Gain	CelebA	Gain
kNN	39.96 %		29.22 %	
kNN-DCT	81.56 %	+ 41.60 %	71.58 %	+ 42.36 %
Eigenfaces	47.07 %		57.58 %	
Eigenfaces-DCT	94.31 %	+ 47.24 %	88.39 %	+ 30.81 %
PRNU Marra et al.	64.28 %		80.09 %	
CNN Yu et al.	98.33 %		99.70 %	
CNN Yu et al.-DCT	99.61 %	+ 1.28 %	**99.91** %	+ 0.21 %
CNN-Pixel	98.95 %		97.80 %	
CNN-DCT	**99.64 %**	+ 0.69 %	99.07 %	+ 1.27 %

Table 4: **The results of using only 20 % of the original data.**. We report the accuracy on the test set and the accuracy loss compared to the corresponding network trained on the full data set.

Method	LSUN	Loss	CelebA	Loss
CNN Yu et al.	92.30 %	-6.03 %	98.57 %	-1.13 %
CNN Yu et al.-DCT	99.39 %	-0.22 %	99.58 %	-0.33 %
CNN-Pixel	85.82 %	-13.13 %	96.33 %	-1.47 %
CNN-DCT	99.14 %	-0.50 %	98.47 %	-0.60 %

training samples and report the accuracy on 25,000 test samples.

Results The results of our experiments are presented in Table 3. We report the accuracy computed over the test set. For each method, we additionally report the gain in accuracy when trained on DCT coefficients.

The use of the frequency domain significantly improves the performance of all tested classifiers, which is in line with our findings from the previous sections. The simpler techniques improve the most, with kNN gaining a performance boost of roughly 42 % and Eigenfaces improving by 47.24 % and 30.81 %, respectively. Our shallow CNN already achieves high accuracy when it is trained on raw pixels (CNN-Pixel), but it still gains a performance boost (+0.69 % and 1.27 %, respectively). The CNN employed by Yu et al. (2019a) mirrors the result of our classifier. Additionally, it seems to be important to utilize the entire frequency spectrum, since the PRNU-based classifier by Marra et al. (2019) achieves much lower accuracy.

The performance gains of classifiers trained on the frequency domain are difficult to examine by solely looking at the accuracy. Thus, we consider the error rates instead: Examining our shallow CNN, we decrease the error rate from 1.05 % and 2.20 % (without the use of the frequency domain) to 0.36 % and 0.93 %, which corresponds to a reduction by 66 % and 50 %, respectively. These results are mirrored for the CNN by Yu et al., where the rates drop from 1.76 % and 0.3 % to 0.39 % and 0.09 %, i.e., a reduction by 75 % and 66 %.

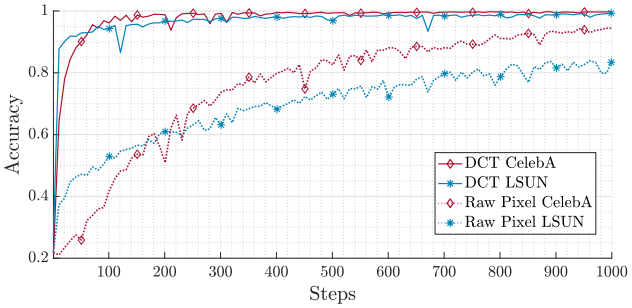


Figure 5: **Validation accuracy for the CCN-classifier by Yu et al.**. We report the validation accuracy during training for the first 1,000 gradient steps, while dropping off in the pixel domain.

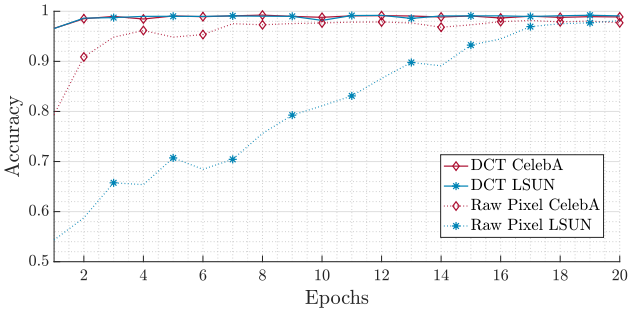


Figure 6: **Validation accuracy for our CNN-classifier**. We report the validation accuracy during training for the first 20 epochs.

4.4. Training in the Frequency Domain

During our experiments we noticed two additional benefits when performing classification in the frequency domain. First, the models trained in the frequency domain require significantly less training data to achieve a high accuracy. Second, training in the frequency domain is substantially easier.

We retrained the network by Yu et al. and our classifier with only 20 % of the original data and reevaluate them. The results are presented in Table 4. In the frequency domain, both classifiers retain a high accuracy. However, both drop significantly when trained on raw pixels. Note that the bigger classifier is able to compensate better for the lack of training data.

We have plotted the validation accuracy during training for both the CNN classifier by Yu et al., as well as for our own in Figure 5 and Figure 6, respectively. For both classifiers and both datasets (CelebA and LSUN), the DCT variant of the classifiers converges significantly faster.

Table 5: **Results of common image perturbations on LSUN bedrooms.** We report the accuracy on the test set. Best score is highlighted in **bold**. The column *CD* refers to the performance of the corresponding classifier trained on clean data and evaluated on perturbed test data. In the column *PD*, we depict the performance when trained on training data which is also perturbed.

	Blur		Cropped		Compression		Noise		Combined	
	CD	PD	CD	PD	CD	PD	CD	PD	CD	PD
CNN-Pixel	60.56 %	88.23 %	74.49 %	97.82 %	68.66 %	78.67 %	**59.51** %	78.18 %	65.98 %	83.54 %
CNN-DCT	**61.42** %	**93.61** %	**83.52** %	**98.83** %	**71.86** %	**94.83** %	48.99 %	**89.56** %	**67.76** %	**92.17** %

CD: Clean Data; PD: Perturbed Data

4.5. Resistance Against Common Image Perturbations

We want to examine if the artifacts persist through post-processing operations when the images get uploaded to websites like Facebook or Twitter. Thus, we evaluate the resistance of our classifier against common image perturbations, namely: blurring, cropping, compression, adding random noise, and a combination of all of them.

Experimental Setup When creating the perturbed data with one kind of perturbation, for each data set we iterate through all images and apply the perturbation with a probability of 50 %. This creates new data sets with about half the images perturbed, these again get divided into train/validation/tests sets, accuracy is reported for the test set. For generating a data set with a combination of different perturbations, we cycle through the different corruptions in the order: blurring, cropping, compression, noise; and apply these with a probability of 50 %. The corruptions are described in the following:

- **Blurring** applies Gaussian filtering with a kernel size randomly sampled from $(3,\ 5,\ 7,\ 9)$.
- **Cropping** randomly crops the image along both axes. The percentage to crop is sampled from $U(5, 20)$. The cropped image is upsampled to its original resolution.
- **Compression** applies JPEG compression, the remaining quality factor is sampled from $U(10, 75)$.
- **Noise** adds i.i.d. Gaussian Noise to the image. The variance of the Gaussian distribution is randomly sampled from $U(5.0, 20.0)$.

We additionally evaluate how well we can defend against common image perturbations by augmenting the training data with perturbed data, i.e., we train a network with a training set that has also been altered by image perturbations.

Results The results are presented in Table 5. Overall, our results show that the DCT variants of the classifiers are more robust w.r.t. all perturbations except of noise.

While the robustness to perturbations is increased, it is still possible to attack the classifier with adversarial examples. In fact, subsequent work has demonstrated that it is possible to evade our classifier with specifically crafted perturbations (Carlini & Farid, 2020).

5. Discussion and Conclusion

In this paper, we have provided a comprehensive analysis on the frequency spectrum exhibited by images generated from different GAN architectures. Our main finding is that *all* spectra contain artifacts common across architectures, data sets, and resolutions. Speculating that these artifacts stem from upsampling operations, we experimented with different upsampling techniques which confirm our hypothesis. We than demonstrated that the frequency spectrum can be used to efficiently and accurately separate real from deep fake images. We have found that the frequency domain both helps in enhancing simple (linear) models, as well as, more complex CNN-based methods, while simultaneously yielding better resistance against image perturbations. Compared to hand-crafted, frequency-based methods, we discovered that the entire frequency spectrum can be utilized to achieve much higher performance. We argue that our method will remain usable in the future because it relies on a fundamental property of today's GANs, i.e., the mapping from low dimensional latent space to a higher dimensional data space.

One suitable approach to mitigate this problem could be to remove upsampling methods entirely. However, this implies two problems. First, this would remove the advantages of having a compact latent space altogether. Second, an instance of StyleGAN used to generate the pictures depicted in Figure 1 already needs 26.2M (Karras et al., 2019) parameters. Removing the low-dimensional layers and only training at full resolution seems infeasible, at least for the foreseeable future.

Another approach could be to train GANs to generate consistent image spectra. We experimented with both introducing a second DCT-based discriminator, as well as, regularizing the generator's loss function with a DCT-based penalty. Unfortunately, neither approach led to better results. Either the penalty or the second discriminator were weighted too weak and had no effect, or the DCT-based methods dominated and led to training collapse. We leave the exploration of these methods as an interesting question for future work.

Acknowledgements

We would like to thank our colleagues Cornelius Aschermann, Steffen Zeiler, Sina Däubener, Philipp Görz, Erwin Quiring, Konrad Rieck, and our anonymous reviewers for their valuable feedback and fruitful discussions. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2092 CASA – 390781972.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. In *International Conference on Machine Learning (ICML)*, 2017.

Azulay, A. and Weiss, Y. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, 2018.

Bappy, J. H., Roy-Chowdhury, A. K., Bunk, J., Nataraj, L., and Manjunath, B. Exploiting spatial structure for localizing manipulated image regions. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Bayar, B. and Stamm, M. C. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *ACM Workshop on Information Hiding and Multimedia Security*, 2016.

Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017.

Bestagini, P., Milani, S., Tagliasacchi, M., and Tubaro, S. Local tampering detection in video sequences. In *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying mmd gans. In *International Conference on Learning Representations (ICLR)*, 2018.

Bondi, L. and Bonettini, N. polimi-ispl/prnu-python: v.1.2. *Zenodo*, 2019.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

Burton, G. J. and Moorhead, I. R. Color and spatial structure in natural scenes. *Applied optics*, 1987.

Carlini, N. and Farid, H. Evading deepfake-image detectors with white-and black-box attacks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

Cozzolino, D., Poggi, G., and Verdoliva, L. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In *ACM Workshop on Information Hiding and Multimedia Security*, 2017.

Durall, R., Keuper, M., and Keuper, J. Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Field, D. J. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America. A, Optics and image science*, 1987.

Field, D. J. Wavelets, vision and the statistics of natural scenes. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 1999.

Fridrich, J. Digital image forensics. *IEEE Signal Processing Magazine*, 2009.

Fried, O., Tewari, A., Zollhöfer, M., Finkelstein, A., Shechtman, E., Goldman, D. B., Genova, K., Jin, Z., Theobalt, C., and Agrawala, M. Text-based editing of talking-head video. *ACM Transactions on Graphics (TOG)*, 2019.

Gonzalez, R. C. and Woods, R. E. *Digital Image Processing*. Pearson, 1992.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Hao, K. The biggest threat of deepfakes isn't the deepfakes themselves. *MIT Technology Review*, 2019.

Kaggle. Generative dog images - experiment with creating puppy pics. https://www.kaggle.com/c/generative-dog-images, 2019.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Khosla, A., Jayadevaprakash, N., Yao, B., and Fei-Fei, L. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brébisson, A., Bengio, Y., and Courville, A. C. Melgan: Generative adversarial networks for conditional waveform synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Lukáš, J., Fridrich, J., and Goljan, M. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security*, 2006.

Lyu, S. Natural image statistics in digital image forensics. In *Digital Image Forensics*, pp. 239–256. Springer, 2013.

Marra, F., Gragnaniello, D., Cozzolino, D., and Verdoliva, L. Detection of gan-generated fake images over social networks. In *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2018.

Marra, F., Gragnaniello, D., Verdoliva, L., and Poggi, G. Do gans leave artificial fingerprints? In *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2019.

McCloskey, S. and Albright, M. Detecting gan-generated imagery using color cues. *arXiv preprint arXiv:1812.08247*, 2018.

Mirza, M. and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations (ICLR)*, 2018.

Mo, H., Chen, B., and Luo, W. Fake faces identification via convolutional neural network. In *ACM Workshop on Information Hiding and Multimedia Security*, 2018.

Nataraj, L., Mohammed, T. M., Manjunath, B., Chandrasekaran, S., Flenner, A., Bappy, J. H., and Roy-Chowdhury, A. K. Detecting gan generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019.

Odena, A., Dumoulin, V., and Olah, C. Deconvolution and checkerboard artifacts. *Distill*, 2016.

Petzka, H., Fischer, A., and Lukovnicov, D. On the regularization of wasserstein gans. In *International Conference on Learning Representations (ICLR)*, 2018.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Razavi, A., van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Simonite, T. Artificial intelligence is coming for our faces. *Wired*, 2019.

Sirovich, L. and Kirby, M. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America. A, Optics and image science*, 1987.

Song, L., Wu, W., Qian, C., He, R., and Loy, C. C. Everybody's talkin': Let me talk as you want. *arXiv preprint arXiv:2001.05201*, 2020.

Tariq, S., Lee, S., Kim, H., Shin, Y., and Woo, S. S. Gan is a friend or foe? a framework to detect various fake face images. In *ACM/SIGAPP Symposium on Applied Computing*, 2019.

Thompson, N. and Lapowsky, I. How russian trolls used meme warfare to divide america. *Wired*, 2017.

Tolhurst, D., Tadmor, Y., and Chao, T. Amplitude spectra of natural images. *Ophthalmic and Physiological Optics*, 1992.

Torralba, A. and Oliva, A. Statistics of natural image categories. *Network: computation in neural systems*, 2003.

Valle, R., Cai, W., and Doshi, A. Tequilagan: How to easily identify gan samples. *arXiv preprint arXiv:1807.04919*, 2018.

van den Oord, A., Vinyals, O., et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Wang, S.-Y., Wang, O., Zhang, R., Owens, A., and Efros, A. A. Cnn-generated images are surprisingly easy to spot... for now. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Wen-Hsiung Chen, Smith, C., and Fralick, S. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on Communications*, 1977.

West, J. and Bergstrom, C. Which face is real? http://www.whichfaceisreal.com, 2019.

Yu, F., Zhang, Y., Song, S., Seff, A., and Xiao, J. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Yu, N., Davis, L. S., and Fritz, M. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *IEEE International Conference on Computer Vision (ICCV)*, 2019a.

Yu, N., Davis, L. S., and Fritz, M. Attributing fake images to gans: Learning and analyzing gan fingerprints. https://github.com/ningyu1991/GANFingerprints, 2019b.

Zhang, R. Making convolutional networks shift-invariant again. In *International Conference on Machine Learning (ICML)*, 2019.

Zhou, P., Han, X., Morariu, V. I., and Davis, L. S. Learning rich features for image manipulation detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2018.

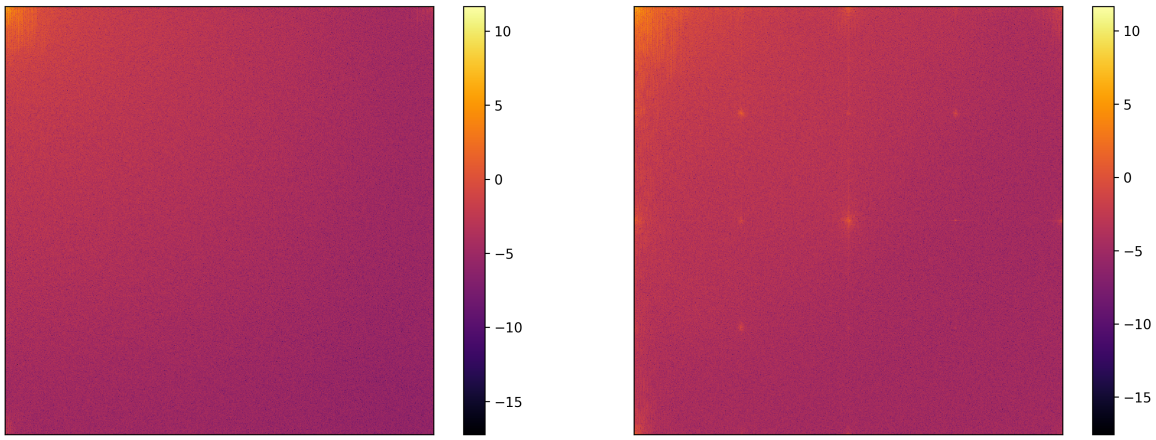
Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Supplementary Material

In this supplementary material, we present all plots in full size, additional statistics, as well as details on our classifier architecture. Note we depict statistics split into color channels only for the Kaggle dataset, since they are consistent with the ones computed over gray-scale images.

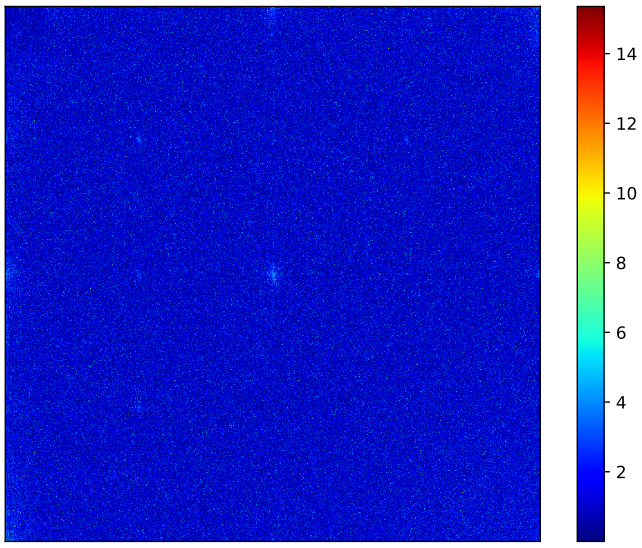
A. FFHQ

We plot the mean of the DCT spectrum of the Flicker-Faces-HQ (FFHQ) data set and an instance of StyleGAN. We estimate $\mathbb{E}[\mathcal{D}(I)]$ by averaging over 10,000 images. Additionally, we plot the absolute difference between the two spectra, notice the additional artifacts scattered throughout the spectrum which are not on the grid.



FFHQ StyleGAN

Figure 7: **The frequency spectrum for real and generated faces (grayscale)**



$$| \mathbb{E}[\mathcal{D}(\text{FFHQ})] - \mathbb{E}[\mathcal{D}(\text{StyleGAN})] |$$

Figure 8: **The absolute difference between the spectra (grayscale)**

Here we also present a plot of a LASSO-regression trained on the FFHQ data set. In context with Figure 1, this makes sense, since compared to the real spectrum, the generated images diverge most in the higher frequencies (real images contain very little energy here). Note that the high frequencies can also be attributed to upsampling operations Durall et al. (2020).

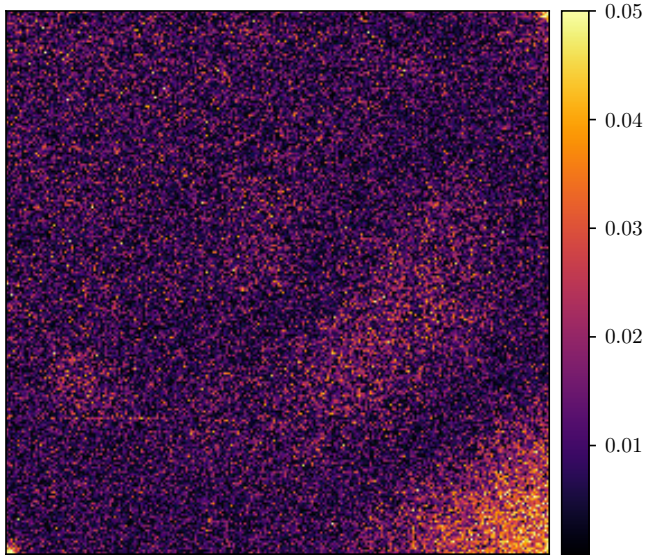
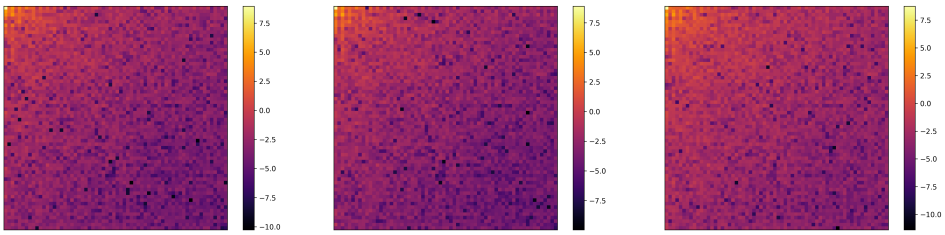


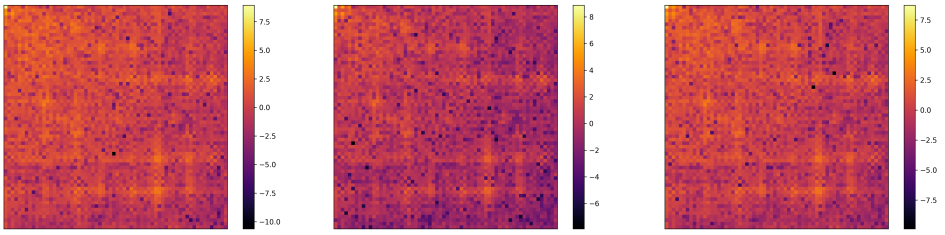
Figure 9: **A heatmap of which frequencies the LASSO-regression uses.** We extracted the weight vector of the regression classifier and mapped it back to the corresponding frequencies. We plot the absolute value of the individual weights and clip their maximum value to 0.05 for better visibility. Note the general focus towards higher frequencies, as well as the top right and lower left corner.

B. Kaggle

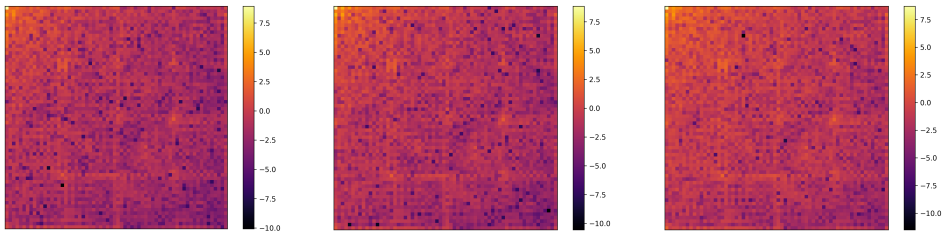
We plot the mean of the DCT spectrum of the Standford dog data set and images generated by different instances of GANs (BigGAN, ProGAN, StyleGAN, SN-DCGAN) trained upon it. We estimate $\mathbb{E}[\mathcal{D}(I)]$ by averaging over 10,000 images.



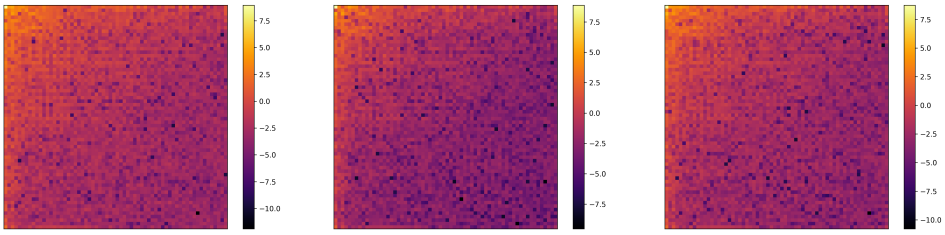
Stanford dogs (red) Stanford dogs (green) Stanford dogs (blue)



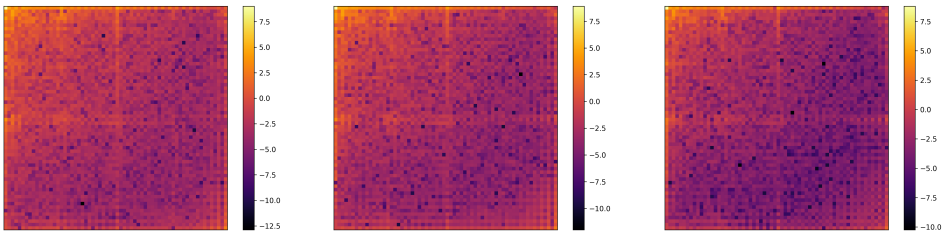
BigGAN (red) BigGAN (green) BigGAN (blue)



ProGAN (red) ProGAN (green) ProGAN (blue)



StyleGAN (red) StyleGAN (green) StyleGAN (blue)

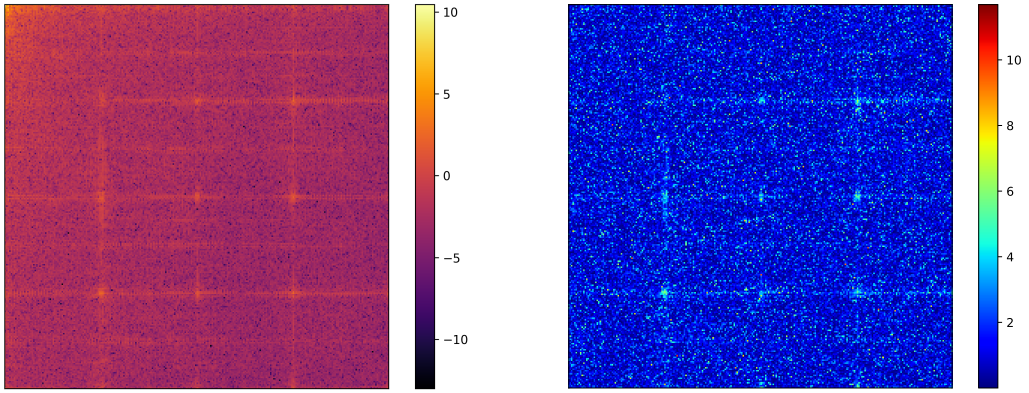


SN-DCGAN (red) SN-DCGAN (green) SN-DCGAN (blue)

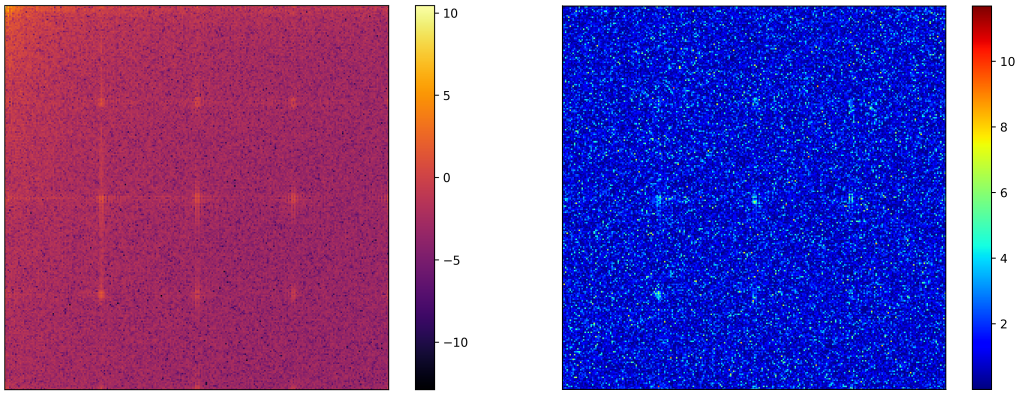
Figure 10: **The frequency spectrum of sample sets generated by different types of GANs trained on the Stanford dog data set (split into color channel)**

C. Upsampling

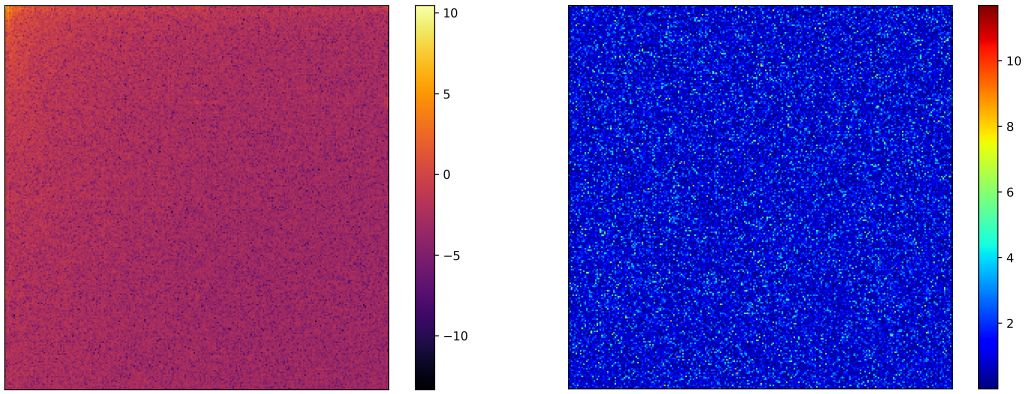
The frequency spectrum resulting from different upsampling techniques. We plot the mean of the DCT spectrum. We estimate $\mathbb{E}[\mathcal{D}(I)]$ by averaging over 10,000 images sampled from the corresponding network or the training data. We additionally plot the absolute difference to the mean spectrum of the training images. Note that, while there is less of a grid, the binomial upsampling still leaves artifacts scattered throughout the spectrum.



Nearest Neighbor | $\mathbb{E}[\mathcal{D}(\text{LSUN})] - \mathbb{E}[\mathcal{D}(\text{Nearest Neighbor})]$ |



Bilinear | $\mathbb{E}[\mathcal{D}(\text{LSUN})] - \mathbb{E}[\mathcal{D}(\text{Bilinear})]$ |



Binomial | $\mathbb{E}[\mathcal{D}(\text{LSUN})] - \mathbb{E}[\mathcal{D}(\text{Binomial})]$ |

Figure 11: **The frequency spectrum resulting from different upsampling techniques, as well as the absolute difference (grayscale)**

D. Network Architecture

For training our CNN we use the Adam optimizer, with an initial learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1^{-7}$, which are the standard parameters for a TensorFlow implementation. We did some experiments with different settings, but none seem to influence the training substantially, so we kept the standard configuration. We train with a batch size of 1024. Again, we experimented with lower batch sizes, which did not influence the training. Thus, we simply picked the largest batch size our GPUs allowed for.

Input (128x128x3)
Conv 3x3 (128x128x3)
Conv 3x3 (128x128x8)
Average-Pool 2x2 (64x64x8)
Conv 3x3 (64x64x16)
Average-Pool 2x2 (32x32x16)
Conv 3x3 (32x32x32)
Dense (5)

Table 6: **The network architecture for our simply CNN.** We report the size of each layer in (brackets).