Wavelet-Packet Powered Deepfake Image Detection

Moritz Wolter  
Fraunhofer Center for Machine Learning  
and Fraunhofer SCAI  
University of Bonn  
wolter@cs.uni-bonn.de

Felix Blanke  
Fraunhofer SCAI  
University of Bonn  
felix.blanke@scai.fraunhofer.de

Charles Tapley Hoyt  
Harvard Medical School  
thoyt@gmail.com

Jochen Garcke  
Institute for Numerical Simulation, University of Bonn  
Fraunhofer Center for Machine Learning  
and Fraunhofer SCAI  
garcke@ins.uni-bonn.de

Abstract

As neural networks become more able to generate realistic artificial images, they have the potential to improve movies, music, video games and make the internet an even more creative and inspiring place. Yet, at the same time, the latest technology potentially enables new digital ways to lie. In response, the need for a diverse and reliable toolbox arises to identify artificial images and other content. Previous work primarily relies on pixel-space convolutional neural networks or the Fourier transform. To the best of our knowledge, wavelet-based generative adversarial neural network (GAN) analysis and detection methods have been absent thus far. This paper aims to fill this gap and describes a wavelet-based approach to GAN-generated image analysis and detection. We evaluate our method on FFHQ [19], CelebA [22], and LSUN [40] source identification problems and find improved or competitive performance.

1 Introduction

While GANs can improve traffic prediction [1], extract useful representations from data [30], translate textual descriptions into images [42], transfer scene and style information between images [44], and generate original art [33], they can also enable abusive actors to quickly and easily generate potentially damaging, highly realistic fake images, colloquially called deepfakes [27, 12]. As the internet becomes a more prominent virtual public space for political discourse and social media outlet [2], deepfakes present a looming threat to its integrity that must be met with techniques for differentiating the real and trustworthy from the fake.

Previous algorithmic techniques for separating real from computer-hallucinated images of people have relied on identifying fingerprints in either the spatial [41, 25] or frequency [12, 11] domain. However, to the best of our knowledge, no techniques have jointly considered the two domains. Here, we study the joint spatio-frequency distribution of wavelet-packet coefficients on real and computer-hallucinated images and find significant differences, especially at the image edges.

We believe our virtual public spaces and social media outlets will benefit from a growing, diverse toolbox of techniques enabling automatic detection of GAN-generated content. In this paper, we aim to make the following contributions:

Preprint. Under review.
A principled wavelet-based analysis of GAN-generated images. Compared to existing work in the frequency domain, we examine the spatio-frequency properties of GAN-generated content for the first time. We find significant differences in both the wavelet-packet mean and standard deviation, especially at the edges.

As a result of the aforementioned wavelet-based analysis, we build classifiers to identify image sources. We work with fixed seed values for reproducibility and report mean and standard deviations over five runs whenever possible. Our systematic analysis shows slightly improved or competitive performance.

To aid the reproduction and extension of the research project our source code is available at https://github.com/gan-police/frequency-forensics.

2 Related work

2.1 Generative adversarial nets

The advent of GANs [15] heralded several successful image generation projects. For a recent review on GAN theory, architecture, algorithms, and applications, we refer the reader to [16]. We highlight the contribution of progressive growth technique, in which a small network is trained on smaller images then gradually increased in complexity/number of weights and image size, on the ability of optimization to converge at high resolutions of $1024 \times 1024$ pixels [18]. As a supplement, style transfer methods [14] have been integrated into new style-based generators. The resulting style-GANs have increased the statistical variation of the hallucinated faces [19, 20]. Finally, regularization (e.g., using spectral methods), has increased the stability of the optimization process [26]. The recent advances have allowed large-scale training [5], which in turn improves the quality further.

2.2 Wavelets

Originally developed within applied mathematics [24, 8], wavelets are a long-established part of the image analysis and processing toolbox [34, 32, 17]. In the world of traditional image coding, wavelet-based codes have been developed to replace techniques based on the discrete cosine transform (DCT) [34]. Early applications of wavelets in machine learning studied the integration of wavelets into neural networks for function learning [43]. Within this line of work, wavelets have previously served as input features [6], or as early layers of scatter nets [23, 7]. Deeper inside neural networks, wavelets have appeared within pooling layers using static Haar [37, 38] or adaptive wavelets [39].

Within the subfield of generative networks, ongoing concurrent research [13] is exploring the use of wavelets to improve GAN generated image content.

2.3 Deepfake detection

Previous work on deepfake image detection broadly falls into two categories. The first group works in the frequency domain. Projects include the study of natural and deep network generated images in the frequency space created by the DCT, as well as detectors based on Fourier features [11, 12, 10, 9]. In particular, [12] visualizes the DCT transformed images and identifies artifacts created by different upsampling methods. We learn that the transformed images are efficient classification features, which allow significant parameter reductions in GAN identification classifiers. Instead of relying on the DCT, [11] studied the distribution of Fourier coefficients for real and GAN-generated images. After noting significant deviations of the mean frequencies for GAN generated- and real-images, classifiers are trained. Finally, [11] spoofed the newly trained Fourier-classifiers by manually adapting the coefficients in Fourier-space.

The second group of classifiers works in the spatial or pixel domain, among others, [41] and [36] train convolutional neural networks directly on the raw images to identify various GAN architectures. According to [41], the classifier features in the final layers constitute the fingerprint for each GAN and are interesting to visualize. Instead of relying on deep-learning, [25] proposed GAN-fingerprints working exclusively with denoising-filter and mean operations.
To the best of our knowledge, we are the first to combine both schools of thought by proposing to work with wavelet packets. As we demonstrate in the next section, wavelet packets allow visualization of frequency information while, at the same time, preserving local information to some degree.

3 Methods

“Wavelets are localized waves, instead of oscillating forever, they drop to zero.” This observation, in the words of [32], leads to the first motivation for using wavelet transforms. In contrast to the periodic sine and cosine waves in the Fourier-basis, wavelets are local and end eventually.

Images often contain sharp borders, where pixel values change rapidly. Such short pulses cause many slowly decaying Fourier coefficients, which fill up the spectrum. Reconstruction of the original pulse depends on the cancellation of most coefficients. The locality of wavelets makes it easier to deal with pulses. If the basis functions end eventually, no cancellation is required.

While previous approaches based on the fast Fourier transform (FFT) [11] and DCT [12] provide abundant frequency space information, the global nature of these bases means all spatial relations are missing. The desire to extend our knowledge by conserving spatial information to some degree is our second reason to work with wavelets. Before discussing the mechanics of the fast wavelet transform (FWT) and its packet variant, we present a proof of concept experiment. We compare the 3° Haar wavelet packets of 5K 1024 × 1024 pixel images from Flickr Faces High Quality (FFHQ) [19] to 5K 1024 × 1024 pixel images generated by StyleGAN. We re-scale the absolute values of the packet representation using the natural logarithm \( \log_e \) for better visibility. This example has previously been studied in [12]. Following [11], we expect to find differences in the coefficient distributions in the frequency domain. Figure 1 shows a sample image from both FFHQ and a StyleGAN generated one, as well as the mean wavelet packets and their standard deviation over 5K samples each. For example, we find that the mean wavelet packets of GAN-generated images are significantly brighter and therefore larger in many filter packets, especially in the third and fourth rows. We confirm the differences previously observed by [11] in Fourier space. Additionally, however, we now have an intuition regarding the local origin of the differences. Figure 2 plots the absolute differences of the
mean absolute packet difference

Figure 2: Absolute distance between the FFHQ and StyleGAN log_e-scaled wavelet packet coefficients mean (and standard deviation), shown on the left and right side, respectively. Especially the third and fourth rows are different. Differences in the facial areas at the center exist but are significantly more pronounced at the edges. The labels for each packet appear in Figure 3 on the right.

mean- and standard deviation images. Face-like shapes appear, suggesting differences there, but the image edges stand out, indicating the most significant differences in this area. We note that Haar wavelet packets do not use padding and conclude that the differences stem from GAN generation. For the standard deviations, the picture is reversed. Now the FFHQ images appear brighter. The StyleGAN packets do not deviate as much as the data in the original dataset. The observation suggests that the GAN did not capture the complete variance in the dataset. Our evidence indicates that GAN-generated backgrounds are of particular monotony.

In the next section, we discuss how the fast wavelet transform (FWT) and wavelet packets in Figure 1 are usually computed.

### 3.1 Fast wavelet transform

Similar to CNNs, the FWT [24, 32, 17] utilizes convolutions to decompose an input signal into its frequency components. The 2D transformation requires the input \( x_q \) as well as a filter quadruple \( f_k \) for \( k \in \{ll, lh, hl, hh\} \) and is computed by

\[
 x_q * f_k = k_{q+1}, 
\]  

(1)

where \(*\) denotes stride two convolution. We denote the representation level by \( q \). The input image is at level zero, i.e. \( x_0 = I \). For later composition stages the low pass result is used as input. Even though 2D wavelets such as the Mexican hat wavelet exist, one dimensional wavelet pairs are often transformed to 2D quadruples using the outer products [35]:

\[
 f_{ll} = f_l f_l^T, \quad f_{lh} = f_l f_h^T, \quad f_{hl} = f_h f_l^T, \quad f_{hh} = f_h f_h^T. 
\]  

(2)

In the equations above, the subscripts \( l \) denote the low-pass and \( h \) the high pass filter, respectively. The wavelet transform is invertible. The forward or analysis transform relies on the analysis filters often denoted as \( H_k \) [32]. The inverse or synthesis transform filters are typically written as \( F_k \). The down- \( \downarrow_2 \) and up- \( \uparrow_2 \) arrow symbols describe the sampling operations in what is known as stride two convolution and transposed convolution in today’s machine learning literature.

In order to make the transform invertible and plots interpretable, not just any filter will do. The perfect reconstruction and anti-aliasing conditions must hold. We refer the interested reader to [32] and [17] for an excellent further discussion of these conditions. For transform to preserve all information and be invertible, higher-order wavelets require padding or special boundary-filters [32]. In a process analogous to CNNs, for higher-order wavelets, the boundary values must be extended with zeros or reflected versions of the boundary values. That way, the wavelets touch every pixel during the convolution and produce invertible filters. To avoid padding-related boundary artifacts, we rely on first-degree-Haar wavelets in this project.
We show a schematic drawing of the process on the left of Figure 3. The upper half shows the analysis which allows inversion for completeness. Finally, for the correct interpretation of Figures 1 and 2 the wavelet packets. For a wavelet packet representation, one recursively continues to filter the low- and high-pass results. Each recursion leads to a new level of filter coefficients, starting with an input image $I \in \mathbb{R}^{h \times w}$, and using $n_{0,0} = I$. A node $n_{q,j}$ at position $j$ of level $q$, is convolved with all filters $f_k$, $k \in [l, lh, hl, hh]$: 

$$n_{q,j} * f_k = n_{q+1,k}.$$  

Figure 3: Visualization of the 2D wavelet packet transform analysis and synthesis packet transform (left). The analysis filters are written as $H_k$ synthesis filters as $F_k$. We show all first level coefficients $a, h, v, d$, as well as some second level coefficients $aa, ah, av, ad$. The dotted lines indicate omission of further possible analysis and synthesis steps. The transform is invertible in principle, $\hat{I}$ denotes the reconstructed original input. The right shows the labels for the $3^q$ transforms we showed previously. In the right plot, $a$ denotes the low-low (LL) coefficients, $h$ denotes the low-high (LH) coefficients, $v$ denotes high-low (HL) coefficients, and $d$ denotes the high-high (HH) coefficients. On the right, boxes highlight the first level pattern of the filter arrangement. Sub-boxes appear for the second and third levels as well but are not highlighted for better visibility.

### 3.2 Wavelet packets

Presuming to find the most essential information in the lower frequencies, standard wavelet transformations decompose only the low-pass or $ll$ coefficients further. The $lh$, $hl$, and $hh$ coefficients are left untouched. While this is often a reasonable assumption [32], previous work [11] found the higher frequencies equally relevant for deepfake detection. Our analysis, therefore, relies on wavelet packets. For a wavelet packet representation, one recursively continues to filter the low- and high-pass results. Each recursion leads to a new level of filter coefficients, starting with an input image $I \in \mathbb{R}^{h \times w}$, and using $n_{0,0} = I$. A node $n_{q,j}$ at position $j$ of level $q$, is convolved with all filters $f_k$, $k \in [l, lh, hl, hh]$: 

$$n_{q,j} * f_k = n_{q+1,k}.$$  

Therefore every node at level $q$ will spawn four nodes at the next level $q + 1$. The result at the final level $Q$, assuming Haar or boundary wavelets without padding, will be a $4^Q \times \frac{h}{2^Q} \times \frac{w}{2^Q}$ tensor, i.e. the number of coefficients is the same as before, and is denoted by $Q^2$. Thereby wavelet packets provide filtering of the input into progressively finer equal-width blocks, with no redundancy. Note that for multi-channel color images, we transform each color separately and consider only single channels here for simplicity. For excellent presentations of the one-dimensional case we again refer to [32] and [17].

Instead of numbering the nodes, we record the chain of filters that where used to produce each set of coefficients. To save space, $a$ denotes the low-low (LL) coefficients, $h$ denotes the low-high (LH) coefficients, $v$ denotes high-low (HL) coefficients, and $d$ denotes the high-high (HH) coefficients.

We show a schematic drawing of the process on the left of Figure 3. The upper half shows the analysis transform, which leads to the coefficients we are after. The lower half shows the synthesis transform, which allows inversion for completeness. Finally, for the correct interpretation of Figures 1 and 2 the right of Figure 3 lists the filter combinations for all previously shown plots.

### 4 Image Source Separation Experiments

In a first experiment, we will attempt to separate FFHQ from StyleGAN generated images using a linear classifier. We work with 126K images per class for training, keep 4K for validation, and 10K for testing. All images have a $128 \times 128$ pixel resolution. In Figure 2 the absolute differences of the
Figure 4: The mean 3° Haar wavelet packet coefficient values for 63K 128 × 128 pixel FFHQ (blue) and StyleGAN (orange) images are shown on the left. The shaded area indicates a single standard deviation σ. We find higher mean coefficient values for the StyleGAN samples across the board, with the exception of the approximation packet labeled aaa, shown on the very left. The right side plots the source identification validation accuracy for two linear regression networks during training and testing. The blue line shows the pixel and the orange line the training accuracy using loge-scaled absolute wavelet packet coefficients. For both lines, shaded areas indicate a single standard deviation over five runs with differently seeded Mersenne-twister initializations. We find that working with loge-scaled absolute packets allowed linear separation and significantly increased the convergence speed in this case.

mean wavelets coefficients and their standard deviations appeared significantly different. We saw differences in the row, starting with the avv filter packet. To put the differences in relation to each other, we remove the spatial dimensions by averaging over these as well. The result is and visualized in Figure 4 on the left. We observe mean differences across the board with the exception of the aaa approximation coefficients. In comparison to the FFHQ standard deviation, the variance produced by the StyleGAN, shown in red, appears smaller.

4.1 FFHQ

Encouraged by the differences in mean values and standard deviations across the board, we attempt to linearly separate the loge-scaled 3° Haar wavelet packets by training a linear regression model. We plot the mean validation and test accuracy over 5 runs with identical seeds of 0, 1, 2, 3, 4 in Figure 4 on the right. The shaded areas indicate a single σ-deviation. Using mean and standard deviations computed on the train set both coefficients and raw pixels are normalized by mean subtraction and division by σ. On both normalized-features, we train identical networks using the Adam optimizer [21] with a step size of 0.001 in PyTorch [28]. On the test set we observe a mean accuracy of 99.77 ± 0.07% for the loge-scaled packets and 81.93 ± 1.02 for the raw pixels. We conclude that the packet coefficient allowed us to separate both classes linearly in this case.

4.2 Large-scale Celeb Faces Attributes and Large-scale Scene UNDERstanding

In the previous experiment, only two image sources, authentic FFHQ and StyleGAN generated had to be identified. In this section, we will study an upscaled problem with more image sources. In other words, we evaluate the ability of a multi-label classifier to differentiate between images from a source dataset and images generated from one of several GANs. We work with the experimental setup as described by [12] and cite their DCT-results for comparison. Additionally, we benchmark against the photoresponse non-uniformity (PRNU) [25] approach, as well as the eigenfaces [31], and straightforward regression as baselines. As we are interested in comparing our results to [12], each of our experiments use four GANs: CramerGAN [3], MMDGAN [4], ProGAN [18], and SN-DCGAN [26].
Table 1: Eigenface [31], linear-regression, PRNU, and CNN source identification results on the LSUN and CelebA datasets. We report the test set accuracy. Whenever possible we report the mean test set accuracy and the standard deviation is given for 5 runs. All convolutional networks have 10961 parameters.

| Method                              | Accuracy | LSUN         |
|-------------------------------------|----------|--------------|
|                                      | CelebA   | max  | µ ± σ | max  | µ ± σ |
| Eigenfaces (ours)                   | 68.56 %  | -    |      | 62.24 % | -    |
| Eigenfaces-DCT [12]                 | 88.39 %  | -    | 94.31 % | -    |
| Eigenfaces-Wavelets (ours)          | 68.54 %  | -    | 62.26 % | -    |
| Eigenfaces-log_e-Wavelets (ours)    | 92.67 %  | -    | 87.91 % | -    |
| PRNU (ours)                         | 83.13 %  | -    | 66.10 % | -    |
| Regression-Pixel (ours)             | 97.30 %  | 95.60 ± 1.62 % | 80.50 % | 77.31 ± 2.65 % |
| Regression-Wavelets (ours)          | 99.20 %  | 99.03 ± 0.20 % | 86.66 % | 82.63 ± 2.15 % |
| Regression-log_e-Wavelets (ours)    | 95.44 %  | 94.91 ± 0.53 % | 91.50 % | 91.20 ± 0.18 % |
| CNN-Pixel [12]                      | 97.80 %  | -    | 98.95 % | -    |
| CNN-DCT [12]                        | 99.07 %  | -    | 99.64 % | -    |
| CNN-Pixel (ours)                    | 98.87 %  | 98.74 ± 0.14 % | 97.06 % | 95.02 ± 1.14 % |
| CNN-Wavelets (ours)                 | 99.35 %  | 99.11 ± 0.18 % | 93.61 % | 92.34 ± 1.49 % |
| CNN-log_e-Wavelets (ours)           | 97.09 %  | 96.79 ± 0.29 % | 97.14 % | 96.89 ± 0.19 % |

150K images were randomly selected from the Large-scale Celeb Faces Attributes (CelebA) [22] and Large-scale Scene Understanding (LSUN) [40] datasets. Our pre-processing is identical for both datasets. The real-images are cropped and resized to 128 × 128 pixels each. With each of the four GANs an additional 150K images are generated at the same resolution. The 750K total images are split into 500K training, 100K validation, and 150K test images. To ensure stratified sampling, we draw the same amount of samples from each generator or the original dataset. As a result, the train, validation, and test sets contain equally many images from each source. As discussed previously, we compute the 3° Haar wavelet packet for each image. For a fair comparison, we normalize both the raw pixels and the wavelet coefficients. Normalization is always the last step before the models. We normalize by subtracting the training-set color-channel mean and dividing each color channel by its standard deviation. The log_e-scaled coefficients are normalized after the rescaling. We attempt to separate the wavelet packet coefficients for each class linearly by regression, eigenfaces, as well as non-linearly using a CNN. Our wavelet packet transform, regression, and convolutional models are implemented using PyTorch [29]. We use Adam [21] to optimize the regression and convolutional models. The batch size is set to 512 and the learning rate to 0.001. For reference, see the supplementary material for convergence plots and the exact network layouts. Table 1 lists the results for comparison. We find that the wavelet representations is descriptive enough to deliver competitive results. Simpler methods, in particular, tend to benefit more from the representation.

Results on the CelebA identification problem appear in the left column of Table 1. In this case, the eigenface approach, in particular, was able to classify 24.11% more images correctly when run on log_e-scaled wavelet packages instead of pixels. A simple regression on unscaled packets was able to outperform the more complex CNN on other features. We note that log_e-scaling was not always beneficial. We elaborate on this point further in Section 6. The CNN outperforms the DCT-based version proposed in [12] with the same amount of parameters on CelebA.

The rightmost column in Table 1 lists our results on the LSUN data-set. While we see gains over the raw pixel representation for log_e-scaled wavelets in the eigenface- and regression-cases, these gains are less pronounced that in the DCT-setting described in [12]. In particular, the CNN did not profit from the unscaled packet representation. The log_e-scaled mean, however, was consistently higher in comparison to the pixel-based CNN trained in our setting. The wavelet packet approach does not perform as well for LSUN as it does for CelebA. We speculate this may be due to the more unstructured and diverse images from scene understanding. The face images are typically
Table 2: Confusion matrix for our best CNN using wavelet features on CelebA. The confusion matrix shows classification errors for the original data-set as well as the CramerGAN [3], MMDGAN [4], ProGAN [18], and SN-DCGAN [26] architectures. The test set contains 30,000 entries per label. Of the GAN-architectures we considered in this experiment the ProGan was the hardest to detect.

| Predicted label | True label | CelebA | CramerGAN | MMDGAN | ProGAN | SN-DCGAN |
|----------------|------------|--------|-----------|--------|--------|----------|
| CelebA         | 29,609     | 8      | 9         | 358    | 16     |          |
| CramerGAN      | 0          | 29,810 | 189       | 1      | 0      |          |
| MMDGAN         | 0          | 193    | 29,804    | 3      | 0      |          |
| ProGAN         | 147        | 1      | 1         | 29,851 | 0      |          |
| SN-DCGAN       | 44         | 0      | 1         | 1      | 29,954 |          |

Table 3: CNN GAN detection results on the LSUN dataset. The CNN was trained on the real data-samples as well as the CramerGAN [3], MMDGAN [4] and ProGAN [18] architectures to differentiate real and generated images. The SN-DCGAN [26] was only added to the test set, which contains 10,000 entries per label. We report the accuracy on the whole test set as well as its subsets with known and unknown labels respectively. The mean test set accuracy and the standard deviation is given for 5 runs.

| CNN-input | LSUN classification test set accuracy |
|-----------|---------------------------------------|
|           | on unknown GAN | on known labels | overall |
|           | max | μ ± σ | max | μ ± σ | max | μ ± σ |
| Pixel     | 61.7 % | 46.5 ± 12.1 % | 95.1 % | 91.9 ± 2.1 % | 87.1 % | 82.8 ± 3.9 % |
| log_e-Wavelets | **81.7 %** | **78.8 ± 1.8 %** | **98.6 %** | **98.4 ± 0.1 %** | **95.2 %** | **94.5 ± 0.4 %** |

centered and foreground and background are more easily distinguished. A comparison of Figure 1 and supplementary Figure 6 illustrates this intuition.

We show the confusion matrix for our best performing network on CelebA in Table 2. Among the GANs we considered the ProGAN-architecture was the hardest to identify, followed by the SN-DCGAN-architecture. Images drawn from both architectures were almost exclusively misclassified as original data and rarely attributed to another GAN. The CramerGAN and MMDGAN generated images were often misattributed to each other, but rarely confused with the original dataset.

5 Detection of images from an unknown generator

To simulate an unknown GAN, we remove the SN-DCGAN generated images from our LSUN training and validation sets. SN-DCGAN images now appear exclusively in the test set, where they never influence the gradients. The training hyperparameters are identical to those discussed in the previous section 4.2. We re-balance the data to contain equally many real and gan-generated images. The task now is to produce a real or fake label in the presence of fake images which were not present during the optimization process. We show the test results in Table 3. We find that our approach does allow detection of the SN-DCGAN samples on LSUN-bedrooms, without their presence in the training set. Please note, that the data re-balancing shifts the naïve baseline accuracy achieved by simply producing the same label all the time from 20 % to 50 %.

We study the confusion matrix in Table 4. Of the 10K test-images from the unknown generator, 8,172 are recognized correctly. In comparison to Table 2, we observe a miss-classification increase for the unseen SN-DCGAN, but also for the other architectures. This is probably because the data re-balancing causes the network to see fewer gan generated images overall, but could also hint that including additional generators increases classification accuracy across the board.
Table 4: Confusion matrix for our best GAN detection CNN using log<sub>e</sub>-scaled wavelet features on LSUN. The CNN was trained on the original data-set as well as the CramerGAN [3], MMDGAN [4], and ProGAN [18] architectures to differentiate real and generated images. The SN-DCGAN [26] was only added to the test set, which contains 10K entries per label.

| True label     | Predicted label | Real | Generated |
|----------------|-----------------|------|-----------|
| LSUN (real)    | 9,814           | 186  |           |
| CramerGAN      | 53              | 9,947|           |
| MMDGAN         | 36              | 9,964|           |
| ProGAN         | 324             | 9,676|           |
| SN-DCGAN (unseen) | 1,828         | 8,172|           |

6 Limitations of the wavelet packet-approach

Unlike DCT-transformed images, the concatenated packets cannot directly be used as input to off-the-shelf CNN-architectures. Figure 1 for example shows mean-packet-representations concatenated into images with dimensions equal to the time domain representation. Running a convolutional network directly on the concatenated packets does not work well. This is a direct consequence of the fast wavelet transforms’ downsampling operations. The packet representation adds additional channels. This change prevents us from using highly tuned, perhaps even pre-trained CNN-architectures. It is possible to use CNN on top of the packets, as we show in Table 1, but the smaller high-channel-inputs require new CNN designs.

Absolute log<sub>e</sub>-scaled coefficients break invertibility and should therefore be used with caution. In our experiments, the re-scaling lead to improvements, especially for smaller networks, in terms of convergence early in the training process, and for the unknown generator classification experiment, we presented in section 5. But taking the absolute value cannot be undone, and the information carried by the sign is lost. On CelebA, for example, log<sub>e</sub>-scaling did not help the linear-regression and CNN models. We, therefore, recommend pixel or raw wavelet control experiments for verification when working with log<sub>e</sub> scaled wavelets. Only the raw wavelet packets are invertible and therefore guaranteed to contain all information that was present in the original images.

7 Conclusion

This paper presented a wavelet packet-based approach for deepfake analysis and detection. We considered the FFHQ-StyleGAN pair consisting of real and artificial face-centered images for our initial analysis. We saw in Figure 1 that wavelet-packets allow the visualization of frequency domain differences while preserving some spatial information. Enough variance existed in the central areas to make out faces. At the same time, the bulk of the standard deviation differences were at the edges and within the background portions of the images. This observation suggests that while contemporary GAN architectures have become quite good at modeling faces, they still fail to capture the surroundings in full detail. The proposed wavelet packet representations allowed linear separation of the FFHQ-StyleGAN source identification problem. Further analysis of the packet representation revealed improved or competitive performance in comparison to the DCT approach. Furthermore, we found our wavelet-CNN classifier able to recognize previously unseen SN-DCGAN images in the LSUN setting. Even though releasing our detection code will allow potential bad actors to test against it, we hope our approach will complement the publicly available deepfake identification toolkit. We plan to investigate the resilience of multi-classifier ensembles in future work.

Acknowledgments and Disclosure of Funding

We thank Daniel Domingo-Fernández and Andre Gemünd for insightful discussions and support. Moritz would like to thank the University of Bonn for access to its Teufelskapelle and Auersberg GPU-Clusters. This work was supported by the Fraunhofer Cluster of Excellence Cognitive Internet
Technologies (CCIT) and the DARPA Young Faculty Award W911NF2010255 (PI: Benjamin M. Gyori).

References

[1] Sandra Aigner and Marco Körner. Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing gans. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-2/W16, 2019, 2019.
[2] Anne Applebaum. Twilight of democracy: the seductive lure of authoritarianism, 2020.
[3] Marc G. Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. CoRR, abs/1705.10743, 2017.
[4] Mikolaj Binkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD gans. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.
[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019.
[6] Lan-lan Chen, Yu Zhao, Jian Zhang, and Jun-zhong Zou. Automatic detection of alertness/drowsiness from physiological signals using wavelet-based nonlinear features and machine learning. Expert Systems with Applications, 42(21):7344–7355, 2015.
[7] Fergal Cotter. Uses of Complex Wavelets in Deep Convolutional Neural Networks. PhD thesis, University of Cambridge, 2020.
[8] Ingrid Daubechies. Ten Lectures on Wavelets. SIAM, 1992.
[9] Ricard Durall, Margret Keuper, and Janis Keuper. Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7890–7899, 2020.
[10] Ricard Durall, Margret Keuper, Franz-Josef Pfreundt, and Janis Keuper. Unmasking deepfakes with simple features. arXiv preprint arXiv:1911.00686, 2019.
[11] Tarik Dzanic, Karan Shah, and Freddie D. Witherden. Fourier spectrum discrepancies in deep network generated images. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.
[12] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 3247–3258. PMLR, 2020.
[13] Rinon Gal, Dana Cohen, Amit Bermano, and Daniel Cohen-Or. Swagan: A style-based wavelet-driven generative model. arXiv preprint arXiv:2102.06108, 2021.
[14] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 2414–2423. IEEE Computer Society, 2016.
[15] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets. In NIPS, 2014.
[16] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A review on generative adversarial networks: Algorithms, theory, and applications. arXiv preprint, 2001.06937, 2020.
[17] Arne Jensen and Anders la Cour-Harbo. Ripples in mathematics: the discrete wavelet transform. Springer Science & Business Media, 2001.
[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8110–8119, 2020.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[22] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

[23] Stéphane Mallat. Group invariant scattering. Communications on Pure and Applied Mathematics, 65(10):1331–1398, 2012.

[24] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. IEEE transactions on pattern analysis and machine intelligence, 11(7):674–693, 1989.

[25] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do gans leave artificial fingerprints? In 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), pages 506–511. IEEE, 2019.

[26] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[27] Simon Parkin. The rise of the deepfake and the threat to democracy. https://www.theguardian.com/technology/ng-interactive/2019/jun/22/the-rise-of-the-deepfake-and-the-threat-to-democracy. Accessed: 2021-05-10.

[28] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[29] David S Taubman and Michael W Marcellin. Jpeg2000: Standard for interactive imaging. Proceedings of the IEEE, 90(8):1336–1357, 2002.

[30] Aparna Vyas, Soohwan Yu, and Joonki Paik. Multiscale transforms with application to image processing. Springer, 2018.
[36] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8695–8704, 2020.

[37] Yu Guang Wang, Ming Li, Zheng Ma, Guido Montufar, Xiaosheng Zhuang, and Yanan Fan. Haar graph pooling. In *International Conference on Machine Learning*, pages 9952–9962. PMLR, 2020.

[38] Travis Williams and Robert Li. Wavelet pooling for convolutional neural networks. In *International Conference on Learning Representations*, 2018.

[39] Moritz Wolter and Jochen Garcke. Adaptive wavelet pooling for convolutional neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1936–1944. PMLR, 2021.

[40] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[41] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7556–7566, 2019.

[42] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[43] Jun Zhang, Gilbert G Walter, Yubo Miao, and Wan Ngai Wayne Lee. Wavelet neural networks for function learning. *IEEE transactions on Signal Processing*, 43(6):1485–1497, 1995.

[44] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.
Figure 5: Comparison of CelebA (left) and LSUN (right) and CramerGAN wavelet mean and std log_{e}-scaled wavelet coefficients over 200K images.

8 Supplementary

In this supplementary, we show the full architectures for all networks. In terms of training convergence, we give the validation accuracy mean and standard deviation during training. Furthermore, we show additional visualizations of the wavelet packets.

Acronyms

CelebA  Large-scale Celeb Faces Attributes
CNN  convolutional neural network
DCT  discrete cosine transform
FFHQ  Flickr Faces High Quality
FFT  fast Fourier transform
FWT  fast wavelet transform
GAN  generative adversarial neural network
LSUN  Large-scale Scene UNderstanding
PRNU  photoresponse non-uniformity
Figure 6: The mean wavelet packet coefficients for the LSUN-data set, as well as cramerGAN generated images are shown in the two plots in the leftmost column. The center column shows their standard deviation. Mean and standard deviations are computed using 200k images each. The rightmost column plots differences for both mean coefficients and their standard deviations.

Figure 7: Regression results on the LSUN (left) and the CelebA (right) datasets. All plots show mean accuracy and standard deviation over 5 runs.
Figure 8: CNN results on the LSUN (left) and the CelebA (right) datasets. All plots show mean accuracy and standard deviation over 5 runs.

8.1 Network Architectures

| Pixel-CNN                  | Wavelet-Packt-CNN          |
|---------------------------|---------------------------|
| Convolution (3x8x3x3), ReLU, | Convolution (192x24x3x3), ReLU, |
| Convolution (8x8x3), ReLU,  | Convolution (24x24x6x6), ReLU, |
| Pool (2, 2),              | Convolution (24x24x9), ReLU, |
| Convolution (8x16x3x3), ReLU, | Linear (24, classes)       |
| Pool (2, 2),              |                           |
| Convolution (16x32x3x3), ReLU, |                           |
| Linear (32*28*28, classes) |                           |

Table 5: The CNN architectures we used in our experiments. We show the convolution and pooling kernel, as well as matrix sizes in brackets. As the packet representation generates additional channels we are forced to adapt the CNN architecture. The parameter counts vary depending on the number of classes, but has been approximately 100k for all experiments.

Unless explicitly stated otherwise, we train with a batch size of 512 and a learning rate of 0.001 using Adam [21]. For the 5 repetitions we report the seed values are set to 0,1,2,3,4.