Supplementary Material: Leveraging Frequency Analysis for Deep Fake Image Recognition

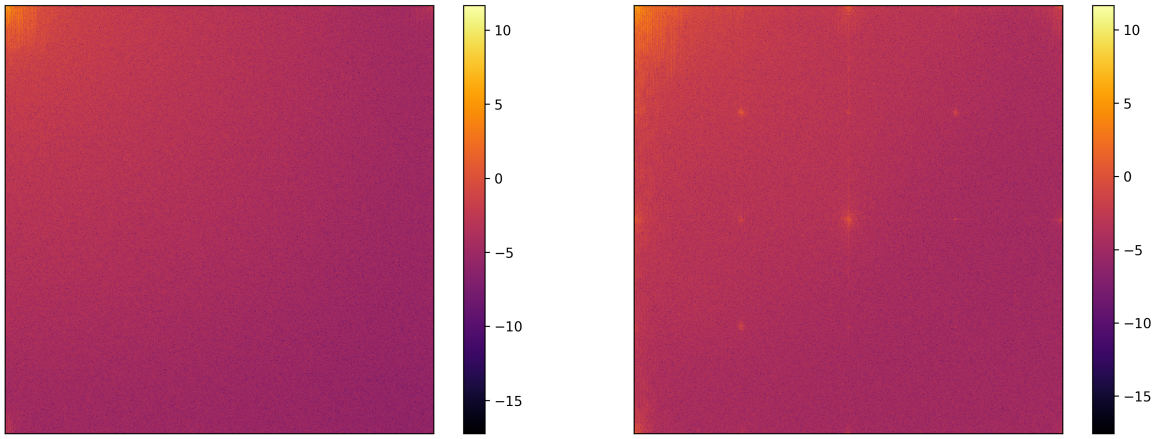
Joel Frank [1] **Thorsten Eisenhofer** [1] **Lea Schönherr** [1] **Asja Fischer** [1] **Dorothea Kolossa** [1] **Thorsten Holz** [1]

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.

1. 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.



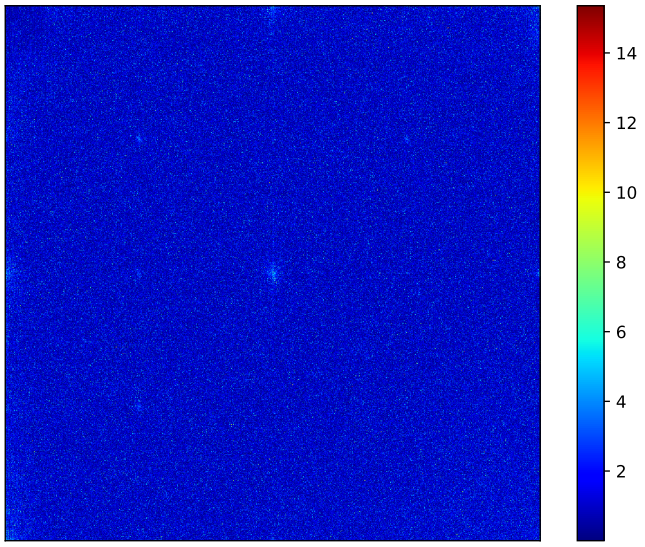
FFHQ StyleGAN

Figure 1: **The frequency spectrum for real and generated faces (grayscale)**

Here we also present a plot of a LASSO-regression trained on the FFHQ data set.

[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, Online, PMLR 119, 2020.



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

Figure 2: **The absolute difference between the spectra (grayscale)**

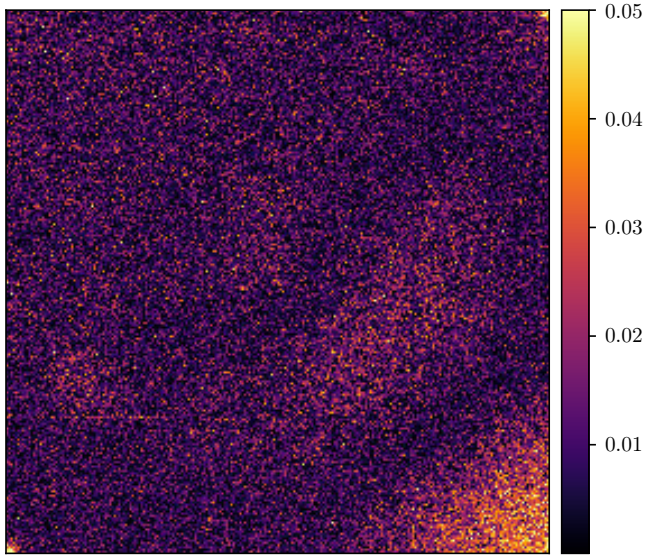
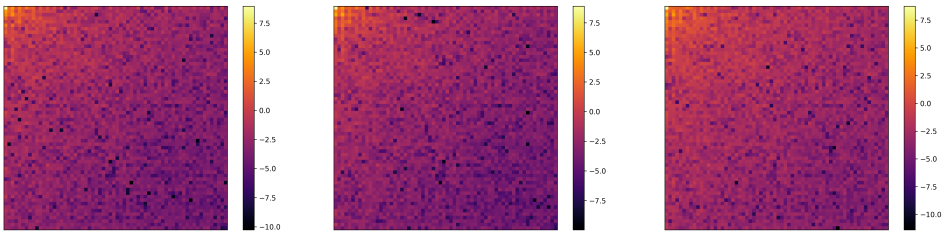


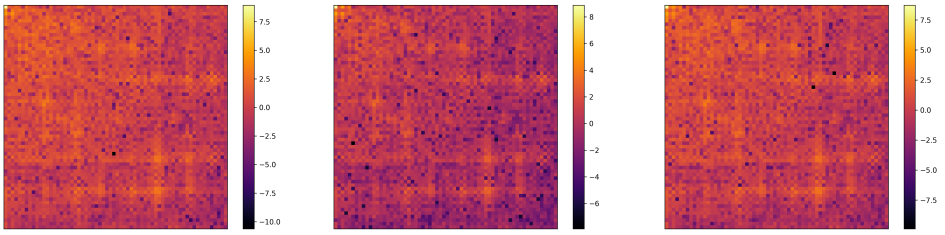
Figure 3: **A heatmap of which frequencies the LASSO-regression uses.** We extracted the weight vector of the regression classifier and mapped it back to the corresponding frequencies. We plot the absolute value of the individual weights and clip their maximum value to 0.05 for better visibility. Note the general focus towards higher frequencies, as well as the top right and lower left corner.

2. Kaggle

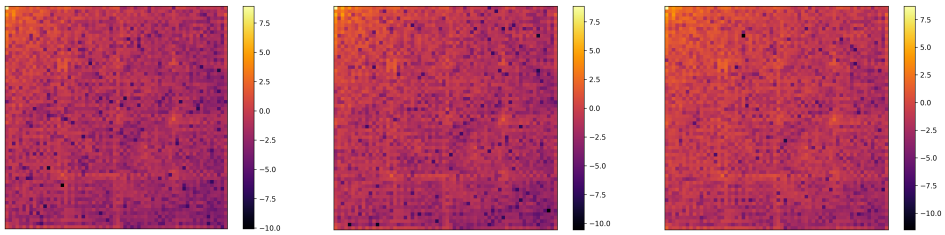
We plot the mean of the DCT spectrum of the Standford dog data set and images generated by different instances of GANs (BigGAN, ProGAN, StyleGAN, SN-DCGAN) trained upon it. We estimate $\mathbb{E}[\mathcal{D}(I)]$ by averaging over 10,000 images.



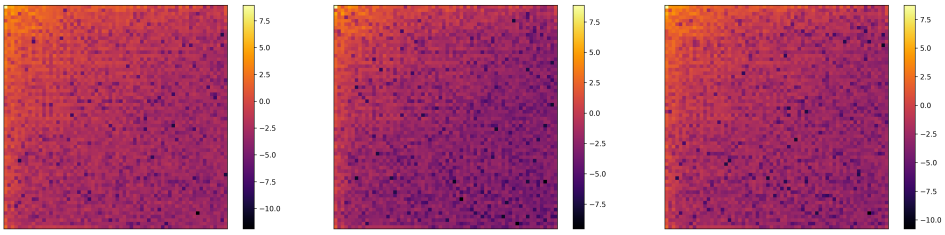
Stanford dogs (red) Stanford dogs (green) Stanford dogs (blue)



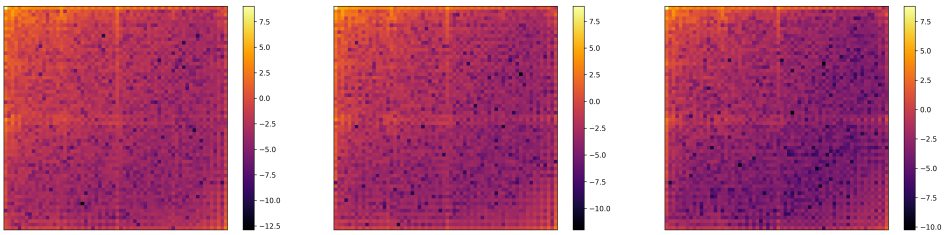
BigGAN (red) BigGAN (green) BigGAN (blue)



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



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

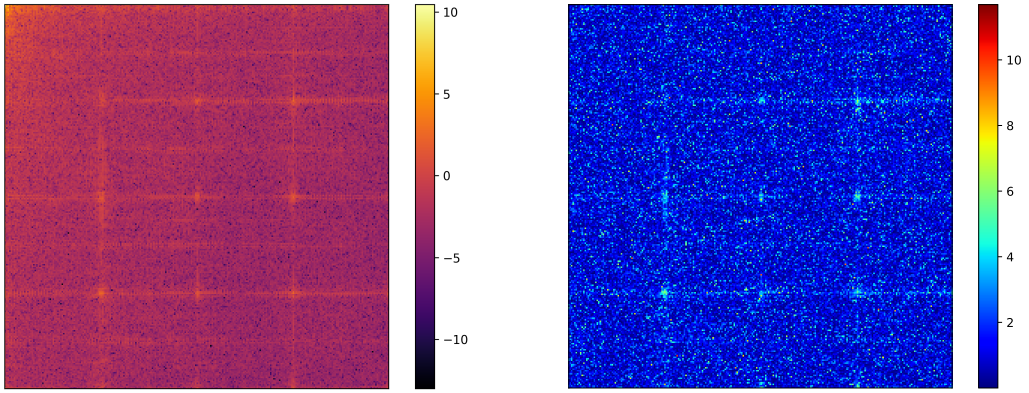


SN-DCGAN (red) SN-DCGAN (green) SN-DCGAN (blue)

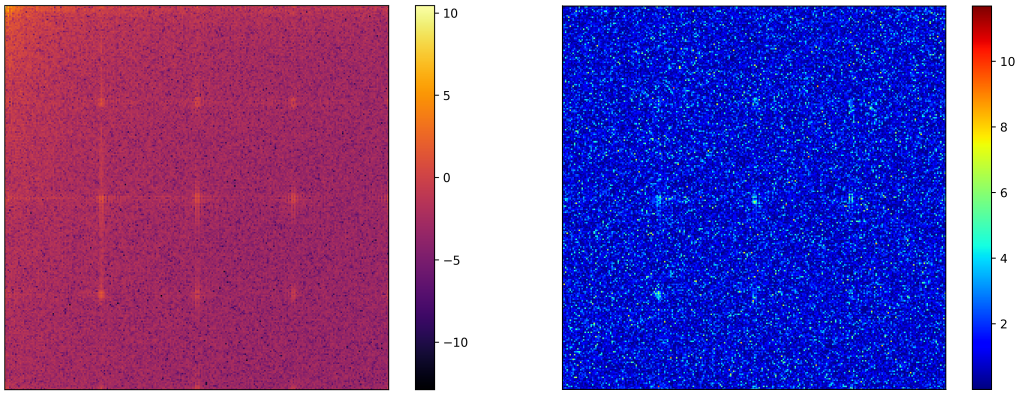
Figure 4: **The frequency spectrum of sample sets generated by different types of GANs trained on the Stanford dog data set (split into color channel)**

3. Upsampling

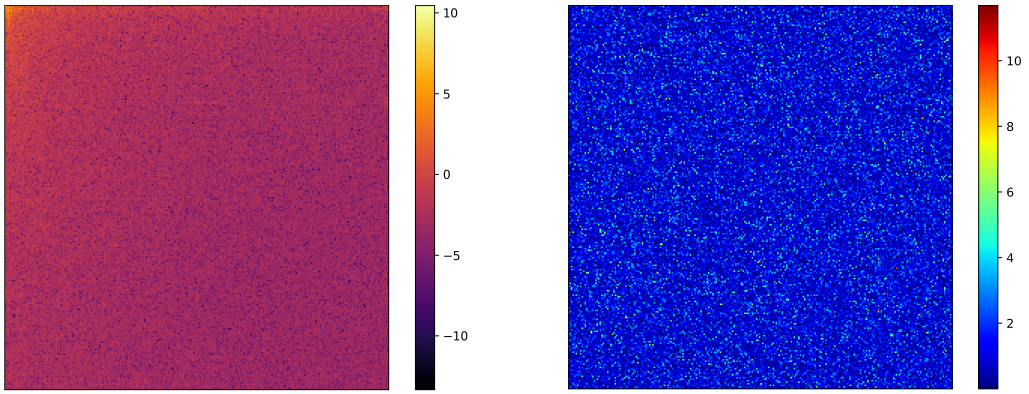
The frequency spectrum resulting from different upsampling techniques. We plot the mean of the DCT spectrum. We estimate $\mathbb{E}[\mathcal{D}(I)]$ by averaging over 10,000 images sampled from the corresponding network or the training data. We additionally plot the absolute difference to the mean spectrum 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 5: **The frequency spectrum resulting from different upsampling techniques, as well as the absolute difference (grayscale)**

4. 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 1: **The network architecture for our simply CNN.** We report the size of each layer in (brackets).