AEROBLADE: Training-Free Detection of Latent Diffusion Images Using Autoencoder Reconstruction Error

Supplementary Material

7. Implementation Details

7.1. Data Collection

Real To obtain real images of high visual quality, we resort to LAION-Aesthetics[2], which defines several subsets from LAION-5B [39] based on image aesthetics. The authors train a linear model on top of CLIP [30] features to predict the aesthetics of an image, which is then used to create multiple collections. For our experiments we choose images with an aesthetics score of 6.5 or higher. Since the images come in various resolutions, we only use images whose smaller side has at least 512 pixels but whose total number of pixels is less or equal to 768^2. We then take the center crop of size 512×512, ensuring that crops contain meaningful content while avoiding resizing operations, which could potentially distort the results.

Stable Diffusion [3] and Kandinsky [32] We use the Diffusers[3] library to generate images using the prompts extracted from real images. All images are generated with the default settings and have size 512×512. We use the same library to compute the reconstructions based on the AEs.

Midjourney [18] We take images generated by Midjourney from a dataset available on Kaggle[4]. It contains the URLs to images from the official Midjourney Discord server, together with some metadata, including the used version. We filter the dataset by version (v4, v5, and v5.1) and only select images which have size 1024×1024.

7.2. Baselines

Gragnaniello et al. [13] and Corvi et al. [5] We use the code and model checkpoints from the official repository[5] provided by Corvi et al. [5], which also contains the detector from Gragnaniello et al. [13].

Ojha et al. [26] We use the code and model checkpoints from the official repository[6]. According to the code, a center crop of size 224×224 is used as input. Since the authors evaluate their method on smaller images (256×256) than we do, we also try resizing images before cropping. However, this does not significantly alter the results.

DIRE [53] We use the code and model checkpoints from the official repository[7]. In particular, we compute the DIRE representations using ADM [7] trained on LSUN Bedroom and classify with the corresponding detector. This setting corresponds to the setting in Table 3 in the original paper [53], where the authors report good generalization to Stable Diffusion 1.5.

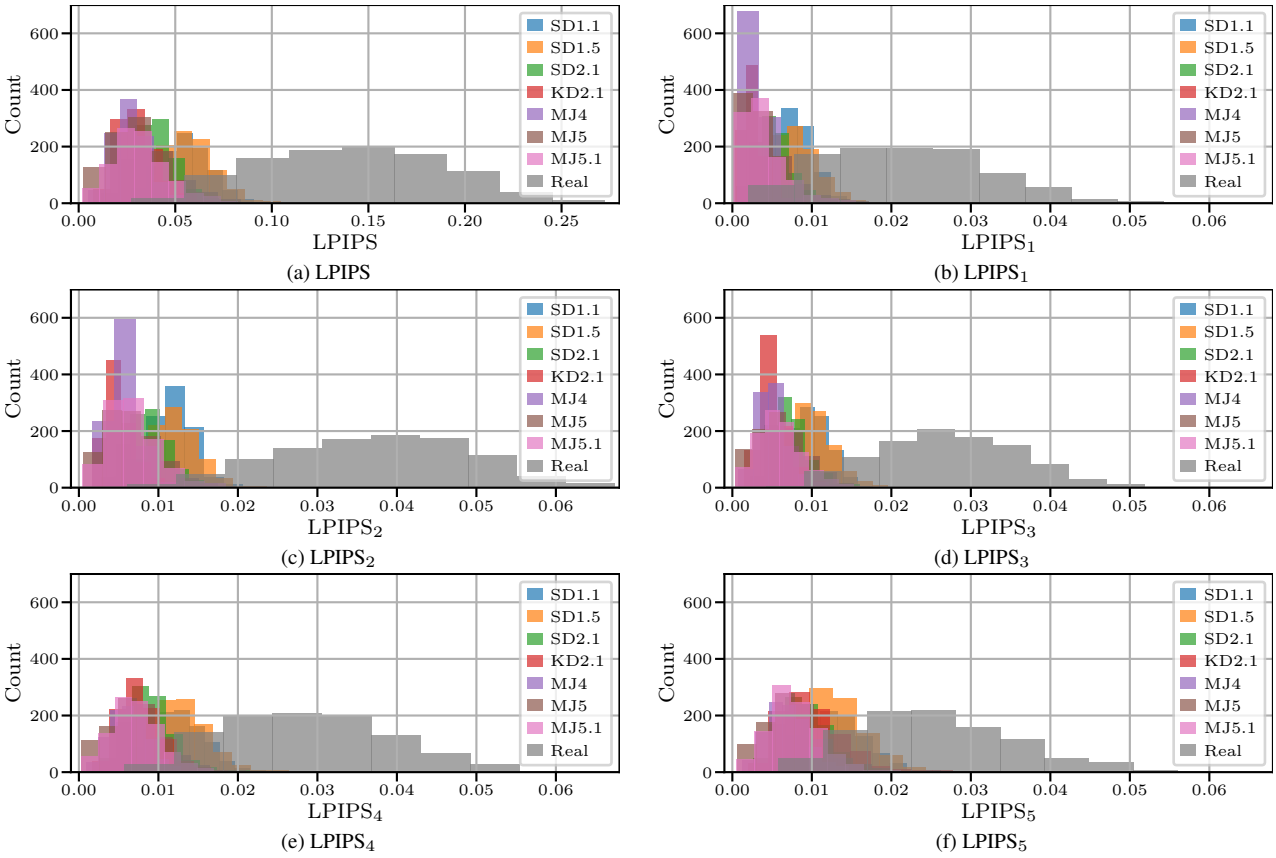
SeDID$_{\text{Stat}}$ [23] At the time of writing, no code is publicly available, which is why we reimplement the method based on the authors' definitions. To guide the denoising process we use the same technique as in our experiment with deeper reconstructions (see Sec. 5.5). We experiment with different values for the total number of steps and T_{SE} and observe that 50 steps in total and $T_{SE} = 25$ achieves the best results.

[2] https://laion.ai/blog/laion-aesthetics
[3] https://github.com/huggingface/diffusers
[4] https://kaggle.com/datasets/iraklip/modjourney-v51-cleaned-data
[5] https://github.com/grip-unina/DMimageDetection
[6] https://github.com/Yuheng-Li/UniversalFakeDetect
[7] https://github.com/ZhendongWang6/DIRE

8. Additional Results

8.1. Reconstruction Error Histograms

Fig. 9 is the extended version of Fig. 3 that includes the other variants of LPIPS. For LPIPS_2 the distributions of reconstruction errors for real and generated samples are most separated. This confirms our results in Tab. 1, according to which LPIPS_2 achieves the highest APs. We observe that for LPIPS_1, the distributions appear to be shifted to the left, indicating that the reconstruction errors are lower for both real and generated images. For higher layers, the results suggest that the error becomes lower for real images but higher for generated images, making the distributions less separable.



(a) LPIPS (b) LPIPS_1 (c) LPIPS_2 (d) LPIPS_3 (e) LPIPS_4 (f) LPIPS_5

Figure 9. Distributions of reconstruction errors Δ_{Min} using different LPIPS variants. (c) is identical to Fig. 3 and is included here for completeness. The x-axis in (a) differs because LPIPS is defined as the sum of all layers.

8.2. Attribution Based on Minimal Reconstruction Error

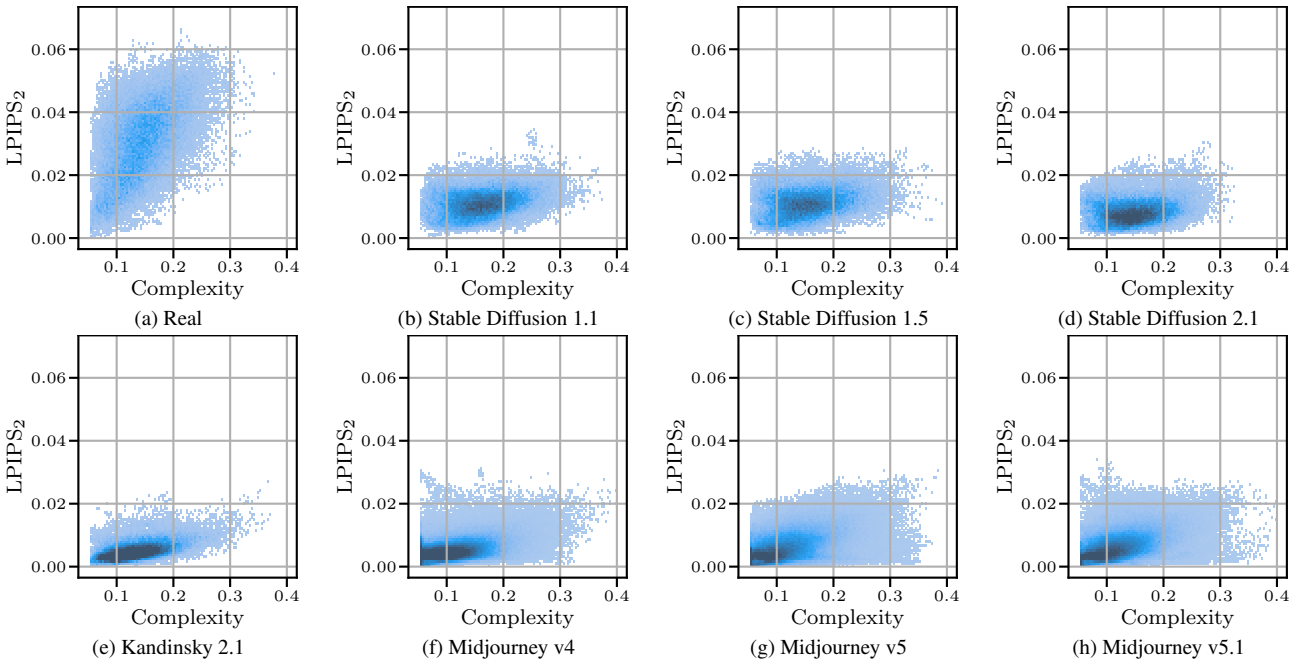
Tab. 5 is the extended version of Tab. 2 that includes all variants of LPIPS. We observe that for LPIPS, LPIPS_1, and LPIPS_2, almost all images are correctly attributed (based on the minimal reconstruction error) to the AE that actually generated them. While this is technically not the case for Midjourney, since the AE is not publicly available, the results strongly indicate that the AE is similar to that of Stable Diffusion 2. Towards higher layers, especially images from Stable Diffusion 1.1 and 1.5 tend to be attributed to the AE of Stable Diffusion 2.

Distance	AE	SD1.1	SD1.5	SD2.1	KD2.1	MJ4	MJ5	MJ5.1
LPIPS	SD1	**1.000**	0.999	0.000	0.000	0.000	0.000	0.000
	SD2	0.000	0.001	**1.000**	0.000	**0.999**	0.993	0.997
	KD2.1	0.000	0.000	0.000	**1.000**	0.001	0.007	0.003
LPIPS_1	SD1	**1.000**	0.999	0.000	0.000	0.000	0.000	0.000
	SD2	0.000	0.001	**1.000**	0.000	**0.999**	**0.999**	**0.999**
	KD2.1	0.000	0.000	0.000	**1.000**	0.001	0.001	0.001
LPIPS_2	SD1	**1.000**	**1.000**	0.000	0.000	0.000	0.000	0.000
	SD2	0.000	0.000	**1.000**	0.000	**0.999**	0.997	0.995
	KD2.1	0.000	0.000	0.000	**1.000**	0.001	0.003	0.005
LPIPS_3	SD1	0.998	0.995	0.000	0.000	0.000	0.001	0.000
	SD2	0.002	0.005	**1.000**	0.000	0.997	0.991	0.995
	KD2.1	0.000	0.000	0.000	**1.000**	0.003	0.008	0.005
LPIPS_4	SD1	0.975	0.954	0.000	0.000	0.000	0.000	0.000
	SD2	0.025	0.046	0.998	0.003	0.998	0.982	0.993
	KD2.1	0.000	0.000	0.002	0.997	0.002	0.018	0.007
LPIPS_5	SD1	0.835	0.772	0.001	0.001	0.000	0.000	0.000
	SD2	0.162	0.228	0.995	0.013	0.997	0.977	0.990
	KD2.1	0.003	0.000	0.004	0.986	0.003	0.023	0.010

Table 5. Fraction of samples for which an AE has the smallest reconstruction error using different LPIPS variants. The highest fraction for each dataset is highlighted in **bold**. The results for LPIPS_2 are identical to Tab. 2 and are included here for completeness.

8.3. Reconstruction Error Against Complexity

Fig. 10 is the extended version of Fig. 5 which contains plots for individual datasets. We find that the relation between complexity and reconstruction error is relatively similar for different generative models, with the variants of Stable Diffusion and Kandinsky yielding more compact distributions than Midjourney.



(a) Real (b) Stable Diffusion 1.1 (c) Stable Diffusion 1.5 (d) Stable Diffusion 2.1

(e) Kandinsky 2.1 (f) Midjourney v4 (g) Midjourney v5 (h) Midjourney v5.1

Figure 10. Reconstruction error against complexity for different datasets. (a) is identical to Fig. 5a and is included her for completeness. The color map is clipped at 1 000 samples for better visibility.

8.4. Inpainting Localization

In Fig. 11 we provide additional examples for reconstruction maps of real images with inpainted regions (corresponding to Fig. 6). Across different scenes, the reconstruction error provides a good indication of where the inpainted region is located.

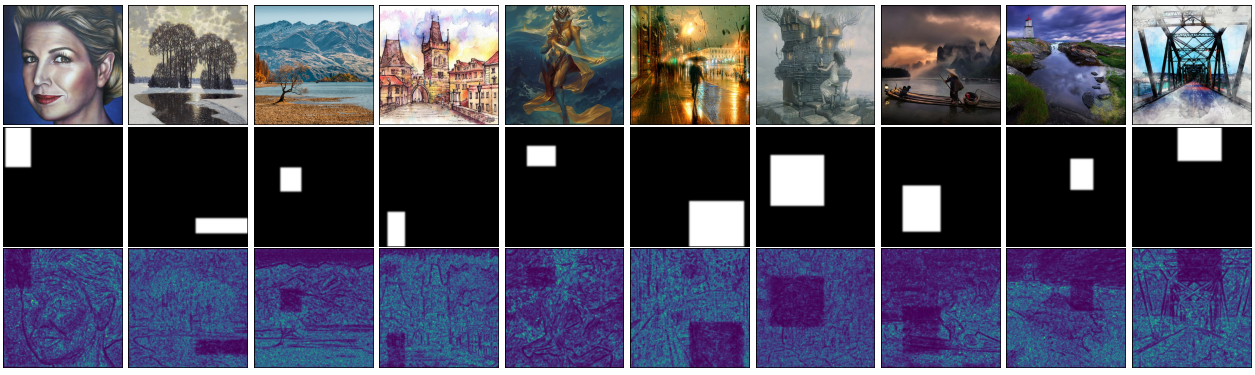
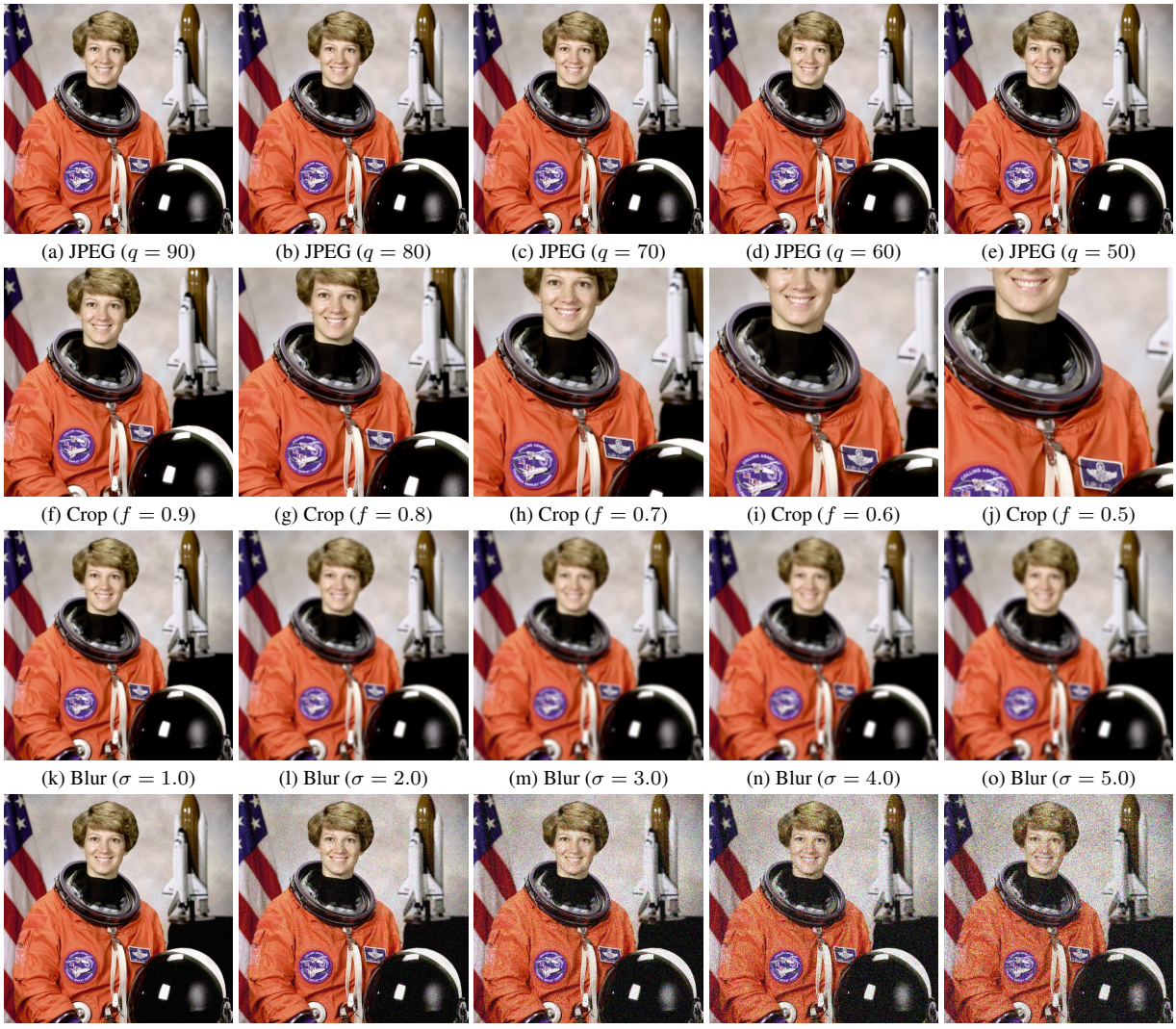


Figure 11. Additional examples illustrating the localization of inpainted regions using the reconstruction error. We show the images inpainted with Stable Diffusion 1.5 (top), the masks used for inpainting (center) and the reconstruction error maps (bottom), computed with LPIPS$_2$ and the AE from Stable Diffusion 1.5.

8.5. Robustness to Perturbations

In Fig. 12 we first provide examples that illustrate how the perturbed images evaluated in Sec. 5.5 look like. Note that for JPEG and cropping, a smaller value (q or f) leads to a stronger perturbation, while for the addition of noise and blurring a larger value (σ) has a stronger effect. For blurring we use a kernel size of 9.



(a) JPEG ($q = 90$) (b) JPEG ($q = 80$) (c) JPEG ($q = 70$) (d) JPEG ($q = 60$) (e) JPEG ($q = 50$)

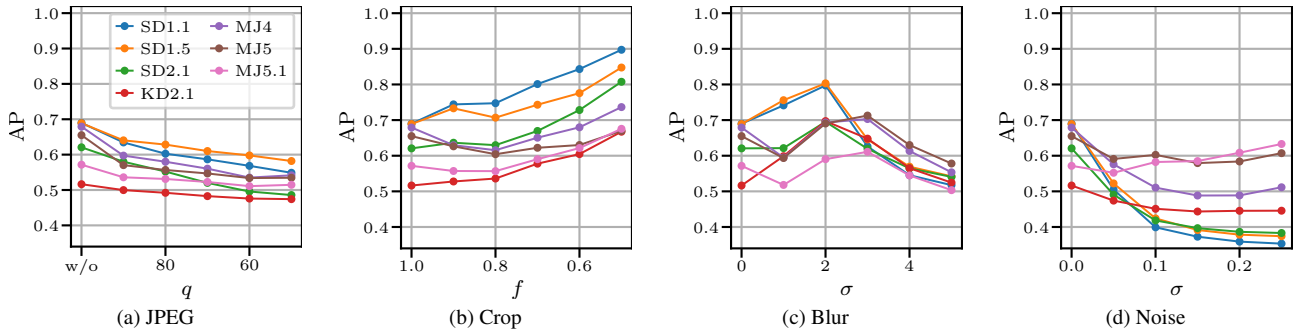
(f) Crop ($f = 0.9$) (g) Crop ($f = 0.8$) (h) Crop ($f = 0.7$) (i) Crop ($f = 0.6$) (j) Crop ($f = 0.5$)

(k) Blur ($\sigma = 1.0$) (l) Blur ($\sigma = 2.0$) (m) Blur ($\sigma = 3.0$) (n) Blur ($\sigma = 4.0$) (o) Blur ($\sigma = 5.0$)

(p) Noise ($\sigma = 0.05$) (q) Noise ($\sigma = 0.1$) (r) Noise ($\sigma = 0.15$) (s) Noise ($\sigma = 0.2$) (t) Noise ($\sigma = 0.25$)

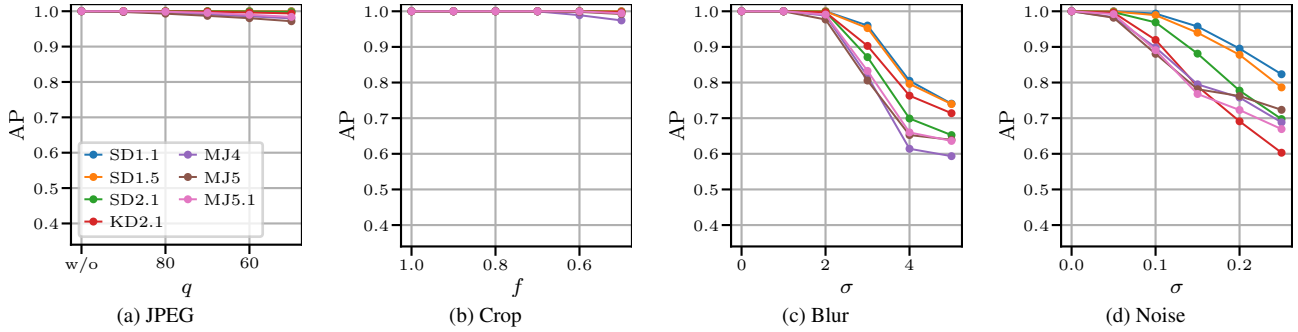
Figure 12. Visualization of different perturbations. In (a)-(e), q denotes the JPEG quality factor. In (f)-(j) f denotes the crop factor. In (k)-(t), σ denotes the standard deviation of the Gaussian blur and noise, respectively.

In addition to Fig. 7, we provide the results on individual datasets for the baselines (see Figs. 13 to 15) as well as all variants of LPIPS (see Figs. 16 to 21). We observe that in some settings, higher layers perform better than LPIPS_2. We suppose that higher layers are less affected by the loss of details caused by the perturbations. This is illustrated by an alternative version of Fig. 7 where we select the optimal LPIPS layer for each perturbation (see Fig. 22). While this of course gives an unfair advantage, it shows the potential robustness of AEROBLADE.



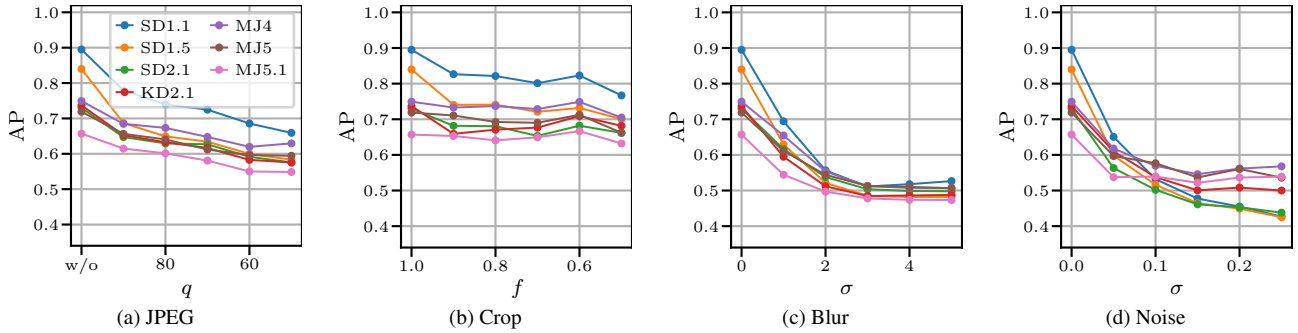
(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 13. Detection performance of the detector from Gragnaniello et al. [13] on perturbed images, measured in AP.



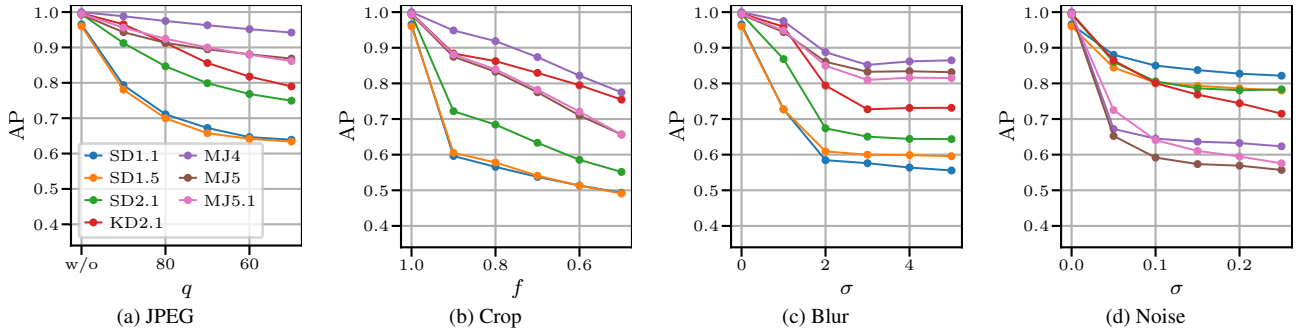
(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 14. Detection performance of the detector from Corvi et al. [5] on perturbed images, measured in AP.



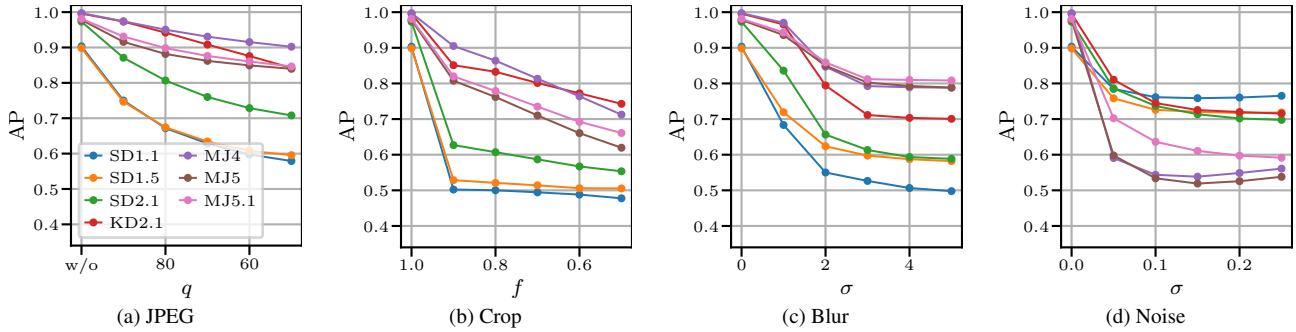
(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 15. Detection performance of the detector from Ojha et al. [26] on perturbed images, measured in AP.



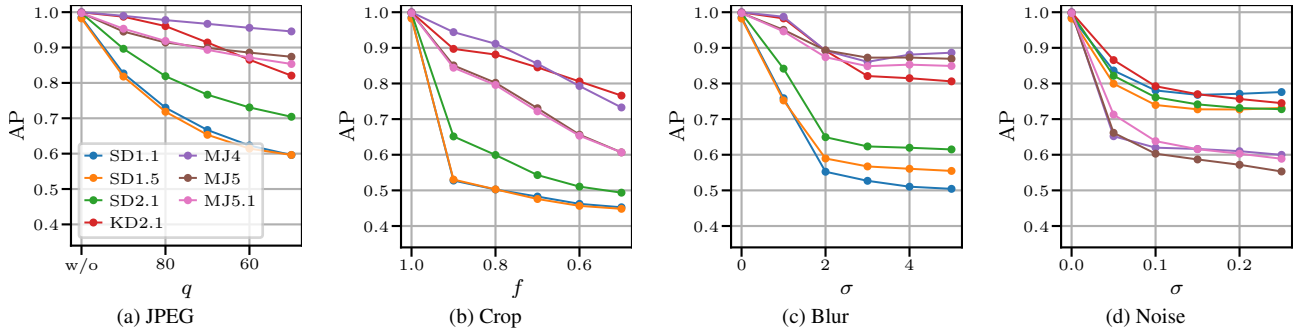
(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 16. Detection performance of AEROBLADE (with LPIPS and Δ_{Min}) on perturbed images, measured in AP.



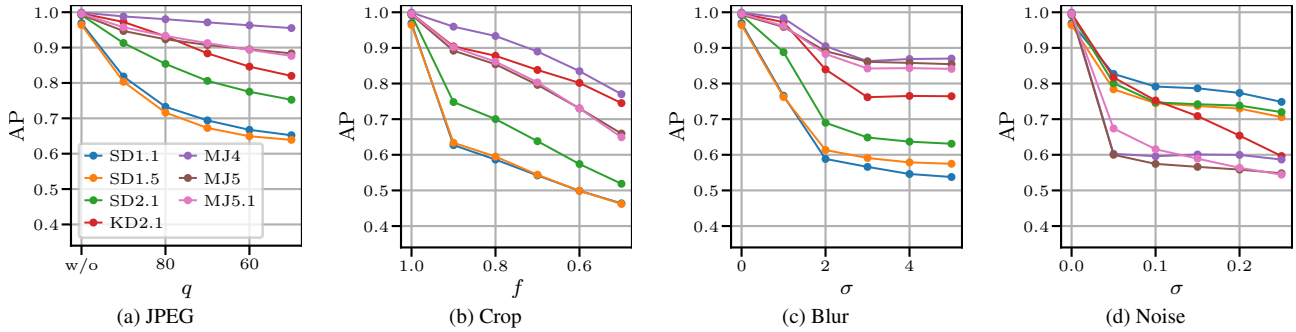
(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 17. Detection performance of AEROBLADE (with LPIPS_1 and Δ_{Min}) on perturbed images, measured in AP.



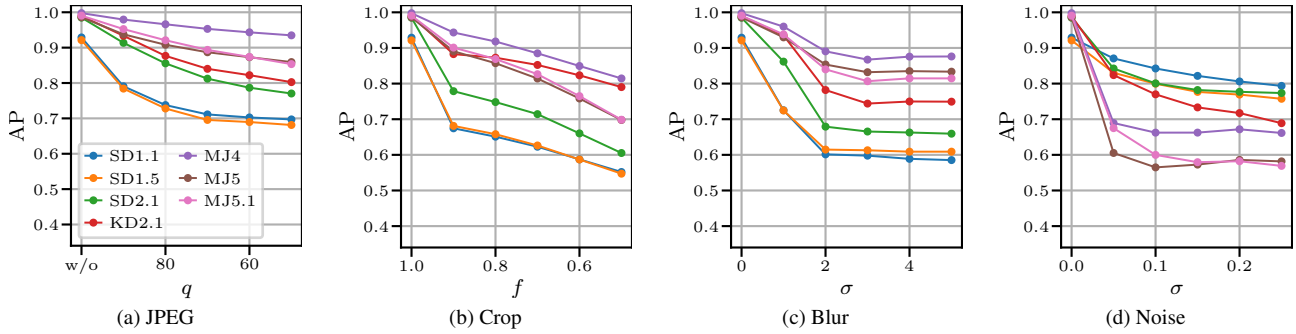
(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 18. Detection performance of AEROBLADE (with LPIPS_2 and Δ_{Min}) on perturbed images, measured in AP.



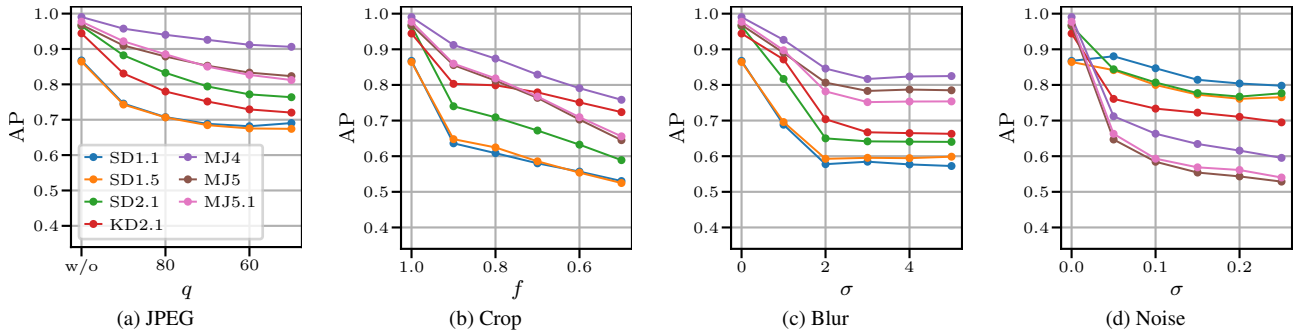
(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 19. Detection performance of AEROBLADE (with LPIPS_3 and Δ_{Min}) on perturbed images, measured in AP.



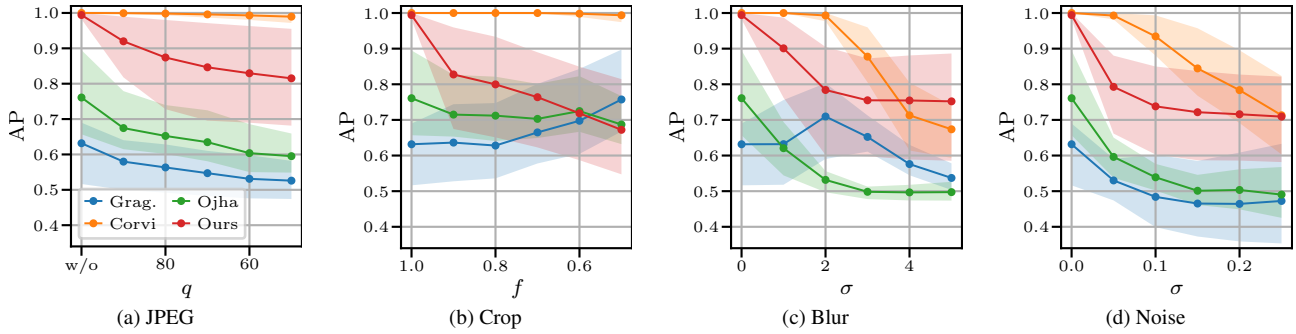
(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 20. Detection performance of AEROBLADE (with LPIPS_4 and Δ_{Min}) on perturbed images, measured in AP.



(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 21. Detection performance of AEROBLADE (with LPIPS_5 and Δ_{Min}) on perturbed images, measured in AP.



(a) JPEG (b) Crop (c) Blur (d) Noise

Figure 22. Detection performance of AEROBLADE (with optimal LPIPS variant for each setting and Δ_{Min}) and baselines on perturbed images, measured in AP. Results are averaged over all datasets, with shaded areas indicating the minimum and maximum.

9. Analysis of DIRE

While trying to reproduce the results from Wang et al. [53], we found that the experiments conducted by the authors might have been affected by an unwanted bias in the data, as we briefly mention in Sec. 5.3. In this section, we provide details on the experiments and results that led us to our findings.

9.1. DIRE

Before proceeding, we briefly summarize the approach proposed by Wang et al. [53]. The idea behind DIRE is that a diffusion model should be able to reconstruct an image more accurately (compared to the original image), if that image is generated by a diffusion model, as opposed to being a real image. In their analysis, the authors use a diffusion model (ADM [7] in most of their experiments) to invert a given image x_0 to its initial time step x_T (corresponding to Gaussian noise) using the inverted DDIM sampler [44], and then denoise back to x'_0 using DDIM as usual. The authors hypothesize that the pixel-wise difference between x_0 and x'_0, the DIRE representation, has different characteristics for real and generated images. To exploit this for classification, they train a binary classifier on the pixel-wise reconstruction error images $|x_0 - x'_0|$ (from real images and images generated using some diffusion model). The authors provide code, data and pre-trained models[8], which we use according to the official instructions in the following experiments.

In the following tables, we test different pre-trained classifiers, which all use the ResNet-50 architecture but are trained on different datasets. These classifiers are ADM-B (trained on LSUN Bedroom vs images generated by ADM trained on LSUN Bedroom), ADM-IN (trained on ImageNet vs images generated by ADM trained on ImageNet), PNDM-B, and IDDPM-B. Further, when referring to datasets (such as in the second and third column in the tables), LSUN-B stands for LSUN Bedroom and ADM-B and ADM-IN stand for images generated using ADM trained on LSUN Bedroom and ImageNet, respectively.

9.2. Experiment 1

We first experiment with the LSUN Bedroom test set provided by the authors. We obtain the images by downloading and extracting `adm.tar.gz` (images generated by ADM trained on LSUN Bedroom) and `real.tar.gz` (real LSUN Bedroom images) from `dire/test/lsun_bedroom/lsun_bedroom`. It should be noted that the authors provide original images, reconstructions, and the DIRE representations (i.e., the differences between originals and reconstructions). The script `test.py`, which is used to evaluate the classification performance, directly uses the DIRE representations.

The key observation we make is that the DIREs of fake images are stored as PNG files (which uses lossless compression), while the DIREs of real images are stored as JPEG files (compressed with quality factor 95). We suspect the reason for this is in `guided_diffusion/compute_dire.py`, where the DIRE files inherit the extension of the corresponding original input image file and the image library automatically applies the corresponding compression. Since the original real images were provided in JPEG, their DIRE images are also automatically stored with JPEG compression. We stress that the authors could not have avoided the compression of the original real images, since the dataset simply comes in this format. However, saving the DIRE in the lossy JPEG format introduces an unwanted bias, as our experiments show. A possible solution to this would have been to store the generated images using the same JPEG compression as the original real images (since uncompressed real images are not available) before computing their DIREs.

Variant 1 We first run `test.py` according to the authors' instructions given in the repository. We compare the performance of different pre-trained classifiers and report the results in Tab. 6a. The "Classifier" column denotes on which kind of generated images (together with the corresponding real images) the classifier was trained on. In this setting, we are able to reproduce the authors' results: all classifiers achieve very good detection and even generalize to other datasets.

Variant 2 We repeat the previous experiment, but this time we ensure that the DIREs of real and generated images are saved consistently. Since we do not have access to real DIREs in their uncompressed form, we instead convert the fake DIREs to JPEG with the same quality level (95) as the real DIREs provided by the authors.

As the results in Tab. 6b show, the performance of all classifiers drops from almost perfect detection (see Tab. 6a) to random guessing. In particular, all images are classified as being real. We emphasize that the only change we made (compared to the previous evaluation) was to convert the uncompressed DIREs of generated images to the same format as the DIREs of real images.

[8] https://github.com/ZhendongWang6/DIRE

Classifier	Real (Test)	Fake (Test)	Acc	AP	Acc_R	Acc_F
ADM-B	LSUN-B	ADM-B	1.000	1.000	1.000	1.000
PNDM-B	LSUN-B	ADM-B	1.000	1.000	1.000	1.000
IDDPM-B	LSUN-B	ADM-B	0.996	1.000	1.000	0.991
ADM-IN	ImageNet	ADM-IN	0.999	1.000	1.000	0.998
ADM-B	ImageNet	ADM-IN	0.998	1.000	0.999	0.999

(a) real DIREs: JPEG, fake DIREs: PNG

Classifier	Real (Test)	Fake (Test)	Acc	AP	Acc_R	Acc_F
ADM-B	LSUN-B	ADM-B	0.501	0.660	1.000	0.200
PNDM-B	LSUN-B	ADM-B	0.500	0.660	1.000	0.000
IDDPM-B	LSUN-B	ADM-B	0.502	0.670	1.000	0.400
ADM-IN	ImageNet	ADM-IN	0.500	0.670	1.000	0.100
ADM-B	ImageNet	ADM-IN	0.500	0.520	0.999	0.000

(b) real DIREs: JPEG, fake DIREs: JPEG

Table 6. Performance of different pre-trained classifiers on test data provided by the authors. The only difference between (a) and (b) is that in (b) the reconstructions of fake images are stored in the JPEG format (quality level 95), like the real images.

9.3. Experiment 2

In a second experiment we further demonstrate that the format in which the DIREs are saved in has a significant influence on the detection. We take 1 000 DIREs of images generated by ADM (trained on LSUN Bedroom) (`dire/test/lsun_bedroom/lsun_bedroom/adm.tar.gz`) and split them in two sets A and B, each containing 500 images.

Variant 1 All DIREs are stored in the lossless PNG format. We pretend that set A is actually a collection of real images, while set B is considered to be fake. Again, we use `test.py` provided by the authors to evaluate the detection performance. The results in Tab. 7a are as expected, since all images are actually generated and half of them have the wrong label, the classifiers achieve an accuracy of approximately 0.5.

Variant 2 We repeat the previous experiment, but convert all DIREs in set A (which we label as being real) to JPEG with quality 95. All other parameters remain unchanged. The results in Tab. 7b show that set A (containing DIREs from generated images saved as JPEGs) is now classified as being real. Thus, the almost perfect classification accuracy (compared to Tab. 7a) is caused exclusively by the fact that we converted the DIREs in set A to the JPEG format.

Classifier	Real* (Test)	Fake (Test)	Acc	AP	Acc_R	Acc_F
ADM-B	ADM-B	ADM-B	0.500	0.500	0.000	1.000
PNDM-B	ADM-B	ADM-B	0.500	0.490	0.000	1.000
IDDPM-B	ADM-B	ADM-B	0.503	0.490	0.012	0.994

(a) real* DIREs: PNG, fake DIREs: PNG

Classifier	Real* (Test)	Fake (Test)	Acc	AP	Acc_R	Acc_F
ADM-B	ADM-B	ADM-B	0.999	1.000	0.998	1.000
PNDM-B	ADM-B	ADM-B	0.997	1.000	0.994	1.000
IDDPM-B	ADM-B	ADM-B	0.997	1.000	1.000	0.994

(b) real* DIREs: JPEG, fake DIREs: PNG

Table 7. Performance of different pre-trained classifiers on test data provided by the authors. "Real*" indicates that the images are labeled as being real but are actually generated by ADM. The only difference between (a) and (b) is that in (b) the reconstructions of images that are labeled as being real are stored in the JPEG format (quality level 95).

9.4. Discussion

In both experiments we show that the classifiers provided by Wang et al. [53] suffer from an unwanted bias in the training data. The DIREs of real images were saved as JPEGs, while those of generated images were saved as lossless PNGs. As a consequence, the detector is highly sensitive to the presence of compression artifacts. In other words, the format in which the DIREs are saved controls whether it is classified as being real or fake, not whether the image is actually real or fake. Based on our findings, we believe that the authors' experiments should be re-evaluated without the bias caused by the inconsistent file formats.