Comparing the interpretability of GAN architectures

Henry Tu
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
henry.tu@mail.utoronto.ca

Seel Patel
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
seel.patel@mail.utoronto.ca

Abstract

A Generative Adversarial Network (GAN) uses an adversarial process between two models which are simultaneously trained to estimate a generative model.(1) There are many architectures, such as COCO-GAN(2) and StyleGAN(3), which can generate samples that mimic real images. We are exploring methods of comparing the quantitative performance of these architectures with their interpretability. Our quantitative measures include C2ST(4) and image quality measures(4). To analyze these networks qualitatively, we will use methods such as Latent Space Exploration(5) and nearest neighbour tests(4) to decipher what the networks have learned. By combining these two forms of analysis, we will gain insight into the relationship between performance metrics and the generated output of the models.

1 Introduction

StyleGAN and COCO-GAN generate images using different techniques to accomplish two different goals: StyleGAN is designed such that high level features of the output image can be finely tuned and adjusted (e.g. lighting, hair colour, etc.)(3). On the other hand, COCO-GAN generates each part of the image separately before stitching it together in order to simulate the human perception of vision(2). As a result, we expected each model to generate images with different qualities that relate to their designed tasks. Both GAN models were trained on the LSUN Bedroom dataset(6) at a resolution of 256×256. Due to time constraints, we used pre-trained models to generate images for our analysis.

NVIDIA Research Projects' Official TensorFlow Implementation of StyleGAN pretrained to the LSUN Bedroom dataset(7) was used with 5000 latent vectors $\mathbf{z} \in \mathbb{R}^{512} \sim \mathcal{N}(0, 1\mathbf{I})$, which each correspond to an output image. The TensorFlow implementation of COCO-GAN by its authors pretrained to the LSUN Bedroom dataset(2) using 5000 latent vectors $\mathbf{z} \in [-1, 1]^{128} \sim \mathcal{U}[-1, 1]^{128}$, which each correspond to an output image.

2 Quantitative analysis

2.1 Classifier Two-Sample tests (C2ST)

C2ST is used to predict if two samples came from the same distribution(8). We trained a CNN to perform classification using a variation of the DCGAN discriminator(9). The goal is to compare the difficulty in which the classifier has with distinguishing between real and generated samples. The classifier was trained using Cross Entropy loss and the Adam optimizer. The training, validation, and test datasets were drawn from $\beta RealImages + (1 - \beta)GANImages, \beta \sim Bern(0.5)$. Out of the pool of 10000 images, 5000 were allocated for training, 2500 validation, and 2500 testing. Let $StyleGANImages, COCOGANImages$ be the distribution of StyleGAN and COCO-GAN generated images respectively. The model was trained with a batch size of 200 for 1000 epoches.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

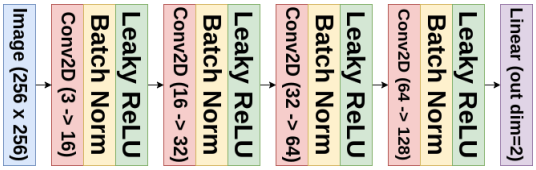

Figure 1: **C2ST architecture.** For all Conv2D layers, Kernel Size: 3, Stride: 2, Padding: 1, Leaky ReLU slope: 0.2. Let $\mathbf{f} : [0,1]^{3\times256\times256} \rightarrow \mathbb{R}^2$ be the neural network. Let $GANImages, RealImages \in [0,1]^{3\times256\times256}$ be the distribution of generated images and samples from the LSUN bedroom dataset respectively. For $\mathbf{x}_{real} \sim RealImages$ (Class 1) and $\mathbf{x}_{gan} \sim GANImages$ (Class 0), the predicted class $\hat{c} = argmax_c(\mathbf{f}(\mathbf{x})[c])$ is correct iff $\hat{c} = class(\mathbf{x})$.

Let $GANImages = \beta StyleGANImages + (1-\beta)COCOGANImages, \beta \sim Bern(0.5)$. After tuning hyper parameters and performing early stopping, the following results were attained:

Table 1: Real vs GAN classifier	
Data Set	Accuracy
Training	75.9% of 5000
Validation	67.6% of 2500
Test (Real)	58.2% of 1250
Test (StyleGAN)	60.5% of 625
Test (COCO-GAN)	89.6% of 625

This data suggests that StyleGAN is better at tricking the classifier than COCO-GAN. To confirm this, we trained additional classifiers using images from only one of the GAN models. Let $GANImages = StyleGANImages, COCOGANImages$ respectively:

Table 2: Real vs StyleGAN classifier	
Data Set	Accuracy
Training	77.6% of 5000
Validation	68.1% of 2500
Test (Real)	74.6% of 1250
Test (StyleGAN)	61.4% of 1250

Table 3: Real vs COCO-GAN classifier	
Data Set	Accuracy
Training	87.9% of 5000
Validation	88.2% of 2500
Test (Real)	96.4% of 1250
Test (COCO-GAN)	76.1% of 1250

As seen here, the Real vs COCO-GAN classifier (Table 3) performed significantly better than the Real vs StyleGAN classifier (Table 2).

2.2 Image sharpness

A common indicator of image quality is sharpness. When an image is sharp, the details within the image are seen more clearly and there is a better distinction between different objects in the picture. Overall, this allows humans to form a better understanding of what they are seeing when they view the image. Using directional image gradients along the x and y axes, we measured the mean image sharpness of the generated images and real images as seen below. (Higher is better)

Table 4: Mean image sharpness			
	Real	StyleGAN	COCO-GAN
Sharpness	8.534	8.214	7.970

Therefore the real images are the sharpest and StyleGAN does the better job of producing sharper images than COCO-GAN. However, does higher sharpness always mean better image quality? We explore this question in the combined analysis section below.

2.3 Color distribution comparison

In order to properly mimic the real images, the generated images have to closely follow the color distribution of the real images. Each image is represented by RGB values between 0 and 1. We extracted the frequencies of RGB color values using 25 bins for each of the datasets. The distributions of these frequencies are plotted in the appendix for the real and generated images. From the frequency plots, we see that both models do a good job of matching the real distribution of colors. However, both generated image color frequency plots show many spikes that are not present in the real image frequency plot. We will explore these spikes in our combined analysis section. Additionally, to quantify this color reproduction quality, we also took the KL Divergence between the generated image color distributions to the real images.

Table 5: KL divergence of color channels w/ real imgs				
	Red	Green	Blue	Gray
StyleGAN	0.286	0.263	0.232	0.251
COCO-GAN	0.044	0.047	0.059	0.016

Therefore COCO-GAN does a statistically better job of modelling the color distribution of the real images. Note that this does not guarantee that COCO-GAN is superior because this method of comparison does not take into account the spatial position of the colors. This means that the images would post the same KL Divergence even if the pixels were scrambled. Therefore further analysis is needed to determine which model performs better overall.

3 Qualitative analysis

3.1 Nearest neighbours

A common problem within GANs is mode collapse. This occurs when the generator consistently outputs the same image for distinct latent vectors. In addition to this, we want to identify if the generator creates multiple images with identical structure or color scheme. To identify these issues, we randomly sampled generated images and visually compared them with the k nearest neighbours from the images produced by the same model. In our search, we found no evidence of mode collapse in either StyleGAN or COCO-GAN. View the appendix for samples used in the analysis.

4 Combined analysis

4.1 Colour frequency peaks

In our analysis of color frequency distributions above, we noticed that both StyleGAN and COCO-GAN exhibited spikes in their frequency graphs. For both models, one prominent spike occurs simultaneously in the green and blue channel at intensity level 0.6. We decided to sample some of these colors and found generated images with color schemes closely matching them.

Figure 2: **RGB (0.3,0.6,0.6) StyleGAN images vs CocoGAN images.** For both models, we are seeing an increase in noise and distortion in the images. Especially COCO-GAN, which displays noisy textures or bad structure in every image. What does this trend tell us about COCO-GAN? From the samples, we can see that many of these images have windows in them that display the sky. COCO-GAN has done a relatively poor job of displaying the sky and natural imagery in the background. Additionally, in the COCO-GAN images the sky color often bleeds into the rest of the image and takes on colors from our gradient of interest. Therefore we theorize that this spike in frequency occurs from COCO-GAN's inability to generate good images that include the sky.

4.2 High sharpness images

In our image sharpness analysis, we found that StyleGAN produced sharper images than COCO-GAN. To qualitatively explore this higher sharpness, we plotted the sharpest images from both models.

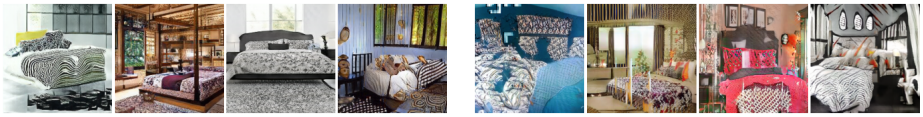

Figure 3: **Sharpest StyleGAN images vs sharpest COCO-GAN images.** As we can see, the StyleGAN images are of high quality, but the COCO-GAN images are consistently filled with noisy patterns. Both sets consist of very sharp images, and therefore we have visualized the sharpness of these images below in order to understand why there is such a large difference in quality.

Figure 4: **StyleGAN vs COCO-GAN sharpness visualizations.** We can see that the sharpness of StyleGAN comes from effectively generating complex patterns while the sharpness of CocoGAN comes from unwanted noise and messy patterns. Therefore, the qualitative analysis indicates that the quality of CocoGAN images is lower than sharpness numbers indicate.

4.3 C2ST smoothgrad analysis

We can use SmoothGrad to generate a sensitivity map to visually discover which parts of the image most strongly affects the classification (10). Smoothgrad was performed on all 3 classifier models outlined in the C2ST section from earlier. Let $\mathbf{x} \in [0,1]^{3\times256\times256}$ be the input image, $\mathcal{L}$ be CE loss, and $\nabla^*_{\mathbf{x}}\mathcal{L}(\mathbf{x})$ be the sensitivity map. $\nabla^*_{\mathbf{x}}\mathcal{L}(\mathbf{x}) = \frac{1}{100}\sum_{i=1}^{100}\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}+\mathcal{N}(0,0.1^2\mathbf{I}))$. Additional sensitivity map samples are available in the appendix.

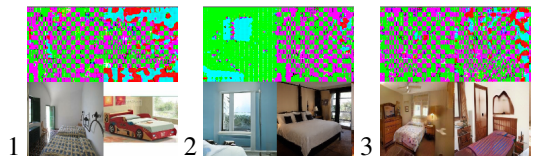

Figure 5: **Real vs GAN classifier sensitivity map. (1: real, 2: StyleGAN, 3: COCO-GAN)** Evidently, there is a clear checkerboard pattern, which could be caused by COCO-GAN's tile generation(2). We can use the classifier trained on only one GAN to validate this theory.

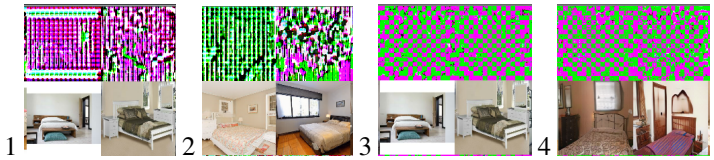

Figure 6: **Real vs StyleGAN $\{1,2\}$ & Real vs COCO-GAN $\{3,4\}$ classifier sensitivity map.** $\{1,3\}$ are real images, and $\{2,4\}$ are from StyleGAN and COCO-GAN respectively. As seen in images $\{3,4\}$, the Real vs COCO-GAN classifier uses a checkerboard filter, which is quite effective with a test accuracy of $\frac{76.1\%+96.4\%}{2} = 86.3\%$ (Table 3). On the other hand, the StyleGAN classifier (images $\{1,2\}$) is very uniform and shows very few artifacts. This seems to suggest there are irregularities in between tiles generated by COCO-GAN which are easily detected by the classifier. Therefore according to C2ST, StyleGAN produces more realistic images than COCO-GAN.

5 Conclusion

In conclusion, ...

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. (2014) *Generative Adversarial Networks*

[2] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, Hwann-Tzong Chen. (2019) *COCO-GAN: Generation by Parts via Conditional Coordinating*

[3] Tero Karras, Samuli Laine, Timo Aila. (2018) *A Style-Based Generator Architecture for Generative Adversarial Networks*

[4] Brownlee, Jason. (2019) *How to Evaluate Generative Adversarial Networks*

[5] Tom White. (2016) *Sampling Generative Networks*

[6] Yu, Fisher and Zhang, Yinda and Song, Shuran and Seff, Ari and Xiao, Jianxiong. (2015) *LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop*

[7] NVLabs. (2019) *StyleGAN — Official TensorFlow Implementation*

[8] David Lopez-Paz, Maxime Oquab. (2016) *Revisiting Classifier Two-Sample Tests for GAN Evaluation and Causal Discovery*

[9] Alec Radford, Luke Metz, Soumith Chintala. (2015) *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*

[10] Daniel Smilko, Nikhil Thorat, Been Kim, Fernanda Viegas, Martin Wattenberg (2017) SmoothGrad: removing noise by adding noise

6 Appendix

6.1 Latent space interpolation

An important part of the generator is it's ability to combine different features from the real image in order to generate new images. For example, in our dataset, an image could contain a bed, a window or a painting. What about a combination of these? What if the locations of these features is changed? We use latent space interpolation to determine if the model produces robust images in all of these cases.

Figure 7: CocoGAN interpolated images

Figure 8: StyleGAN interpolated images

As we can see, both models do a good job of interpolating between different parts of the image. Specifically, both models are able to robustly interpolate between different sizes, colors, orientations, and shapes of beds. The bed is the most important part of a bedroom picture, and this is evidence that both models have a good understanding of beds. Additionally, we can also see that the models are able to interpolate between features like doors and paintings. Overall, the robust latent space interpolation of both models shows that the models have a good understanding of different image features.

6.2 Color distribution frequency plots

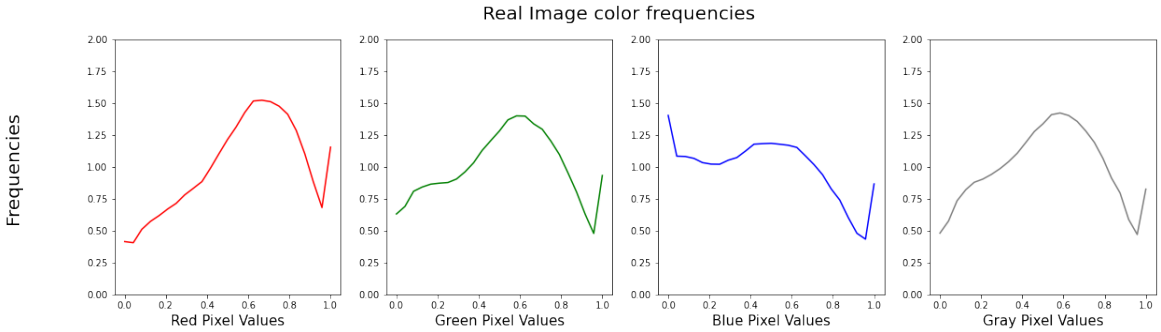

Figure 9: Real image color frequency distribution

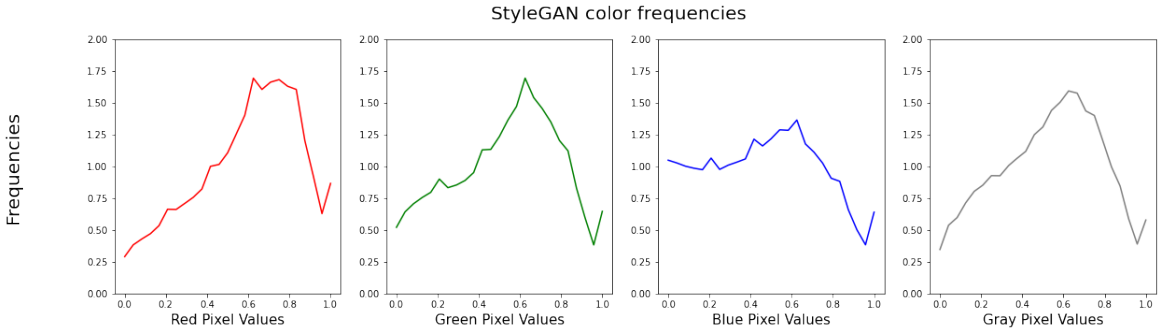

Figure 10: StyleGAN image color frequency distribution

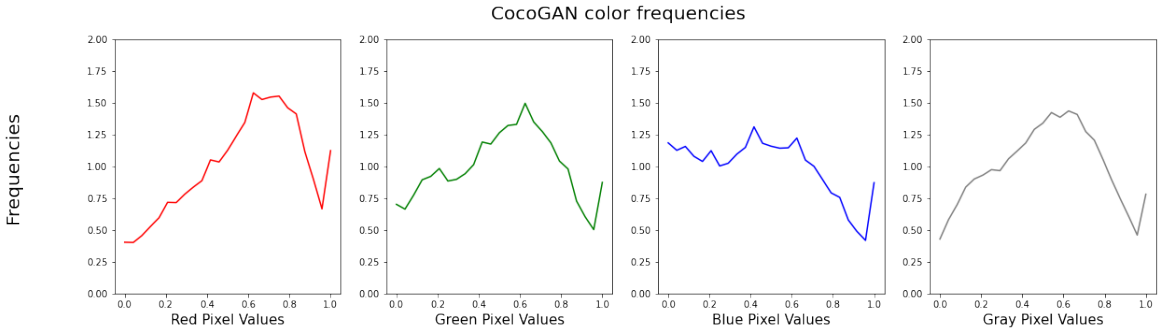

Figure 11: CocoGAN image color frequency distribution

6.3 Nearest neighbour images

Figure 12: CocoGAN test image vs nearest neighbours to test image

Figure 13: StyleGAN test image vs nearest neighbours to test image

6.4 More color frequency peak samples

Figure 14: More CocoGAN images matching color (0.3, 0.6, 0.6)

Figure 15: More StyleGAN images matching color (0.3, 0.6, 0.6)

6.5 More sharp generated image samples

Figure 16: CocoGAN sharpest images and sharpness visualization

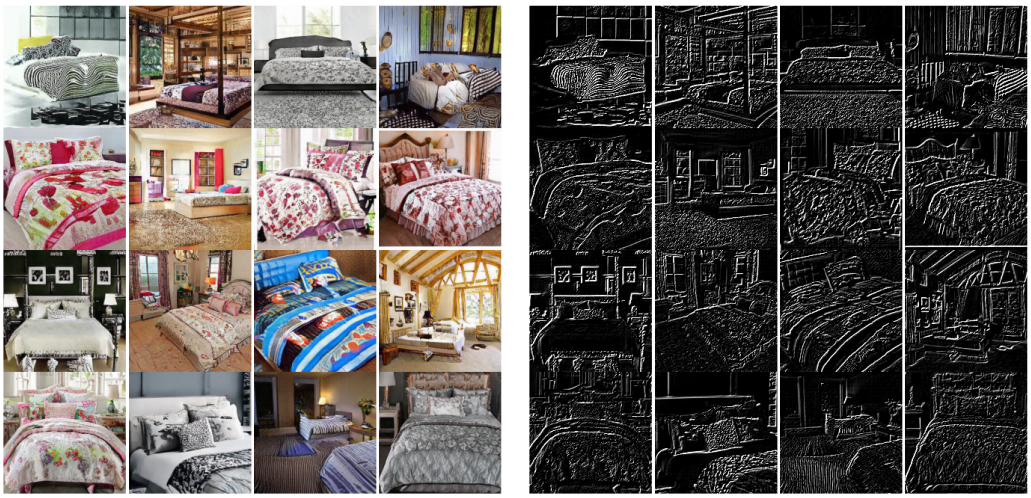

Figure 17: StyleGAN sharpest images and sharpness visualization

6.6 C2ST SmoothGrad

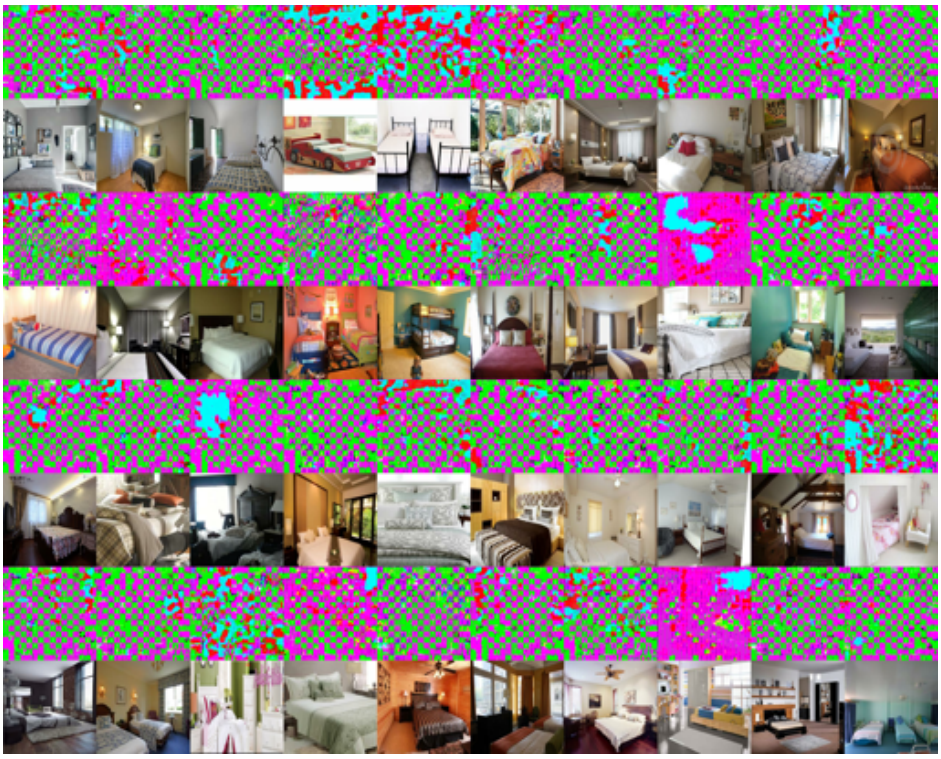

Figure 18: Real vs GAN classifier sensitivity - real image

Figure 19: Real vs GAN classifier sensitivity - StyleGAN

Figure 20: Real vs GAN classifier sensitivity - COCO-GAN

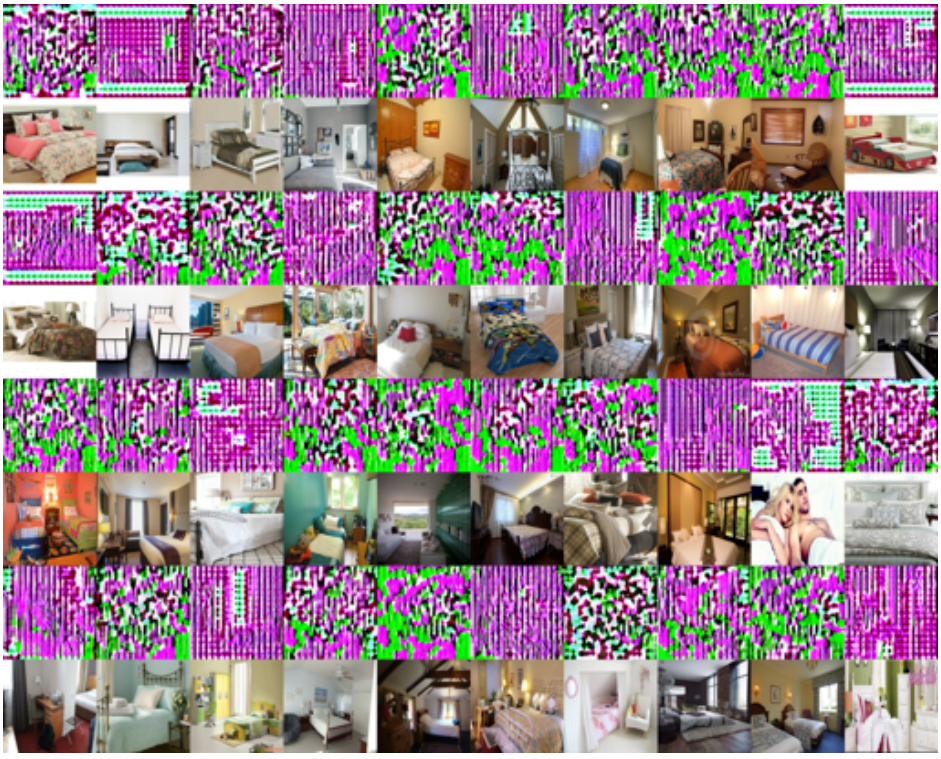

Figure 21: Real vs StyleGAN classifier sensitivity - real image

Figure 22: Real vs StyleGAN classifier sensitivity - StyleGAN

Figure 23: Real vs COCO-GAN classifier sensitivity - real image

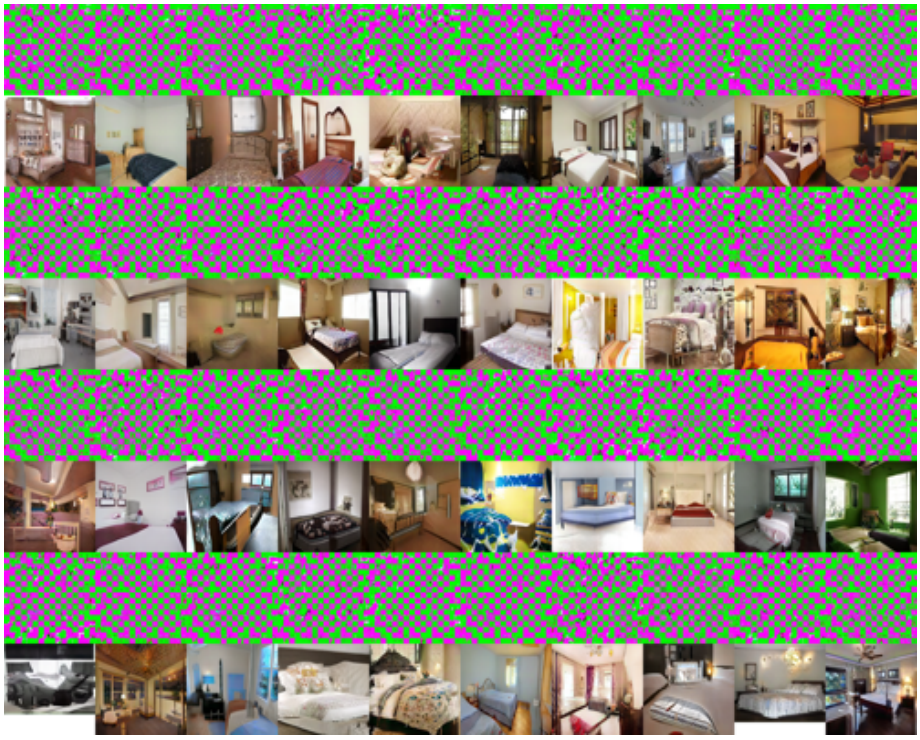

Figure 24: Real vs COCO-GAN classifier sensitivity - COCO-GAN