

# On the “steerability” of generative adversarial networks

Ali Jahanian\*, Lucy Chai\*, Phillip Isola

MIT CSAIL

{jahanian, lrchai, phillipi}@mit.edu

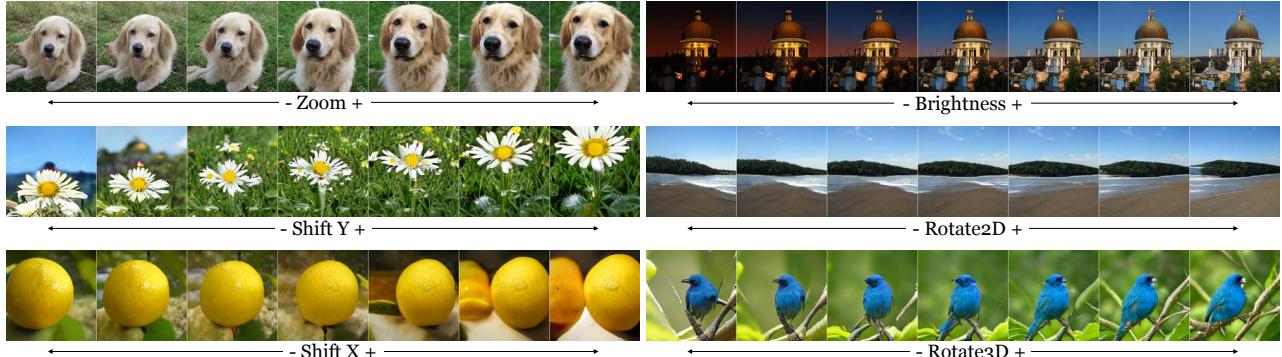


Figure 1: Learned latent space trajectories in generative adversarial networks correspond to visual transformations like camera shift and zoom. These transformations can change the distribution of generated data, but only so much – biases in the data, like centered objects, reveal themselves as objects get “stuck” at the image borders when we try to shift them out of frame. Take the “steering wheel”, drive in the latent space, and explore the natural image manifold via generative transformations!

## Abstract

*An open secret in contemporary machine learning is that many models work beautifully on standard benchmarks but fail to generalize outside the lab. This has been attributed to training on biased data, which provide poor coverage over real world events. Generative models are no exception, but recent advances in generative adversarial networks (GANs) suggest otherwise – these models can now synthesize strikingly realistic and diverse images. Is generative modeling of photos a solved problem? We show that although current GANs can fit standard datasets very well, they still fall short of being comprehensive models of the visual manifold. In particular, we study their ability to fit simple transformations such as camera movements and color changes. We find that the models reflect the biases of the datasets on which they are trained (e.g., centered objects), but that they also exhibit some capacity for generalization: by “steering” in latent space, we can shift the distribution while still creating realistic images. We hypothesize that the degree of distributional shift is related to the breadth of the training data distribution, and conduct experiments that demonstrate this. Code is released on our project page: [https://ali-design.github.io/gan\\_steerability/](https://ali-design.github.io/gan_steerability/)*

## 1. Introduction

The quality of deep generative models has increased dramatically over the past few years. When introduced in 2014, Generative Adversarial Networks (GANs) could only synthesize MNIST digits and low-resolution grayscale faces [10]. The most recent models, on the other hand, produce diverse and high-resolution images that are often indistinguishable from natural photos [4, 13].

Science fiction has long dreamed of virtual realities filled of synthetic content as rich, or richer, than the real world (e.g., *The Matrix, Ready Player One*). How close are we to this dream? Traditional computer graphics can render photorealistic 3D scenes, but has no way to automatically generate detailed content. Generative models like GANs, in contrast, can hallucinate content from scratch, but we do not currently have tools for navigating the generated scenes in the same kind of way as you can walk through and interact with a 3D game engine.

In this paper, we explore the degree to which you can navigate the visual world of a GAN. Figure 1 illustrates the kinds of transformations we explore. Consider the dog at the top-left. By moving in some direction of GAN latent space, can we hallucinate walking toward this dog? As the figure indicates, and as we will show in this paper, the answer is yes. However, as we continue to zoom in,

we quickly reach limits. The direction in latent space that zooms toward the dog can only make the dog so big. Once the dog face fills the full frame, continuing to walk in this direction fails to increase the zoom. A similar effect occurs in the daisy example (row 2 of Fig. 1): we can find a direction in latent space that shifts the daisy up or down, but the daisy gets “stuck” at the edge of the frame – continuing to walk in this direction does not shift the daisy out of the frame.

We hypothesize that these limits are due to biases in the distribution of images on which the GAN is trained. For example, if dogs and daisies tend to be centered and in frame in the dataset, the same may be the case in the GAN-generated images as well. Nonetheless, we find that *some* degree of transformation is possible: we can still shift the generated daisies up and down in the frame. When and why can we achieve certain transformations but not others?

This paper seeks to quantify the degree to which we can achieve basic visual transformations by navigating in GAN latent space. In other words, are GANs “steerable” in latent space?<sup>1</sup> We further analyze the relationship between the data distribution on which the model is trained and the success in achieving these transformations. We find that it is possible to shift the distribution of generated images to some degree, but cannot extrapolate entirely out of the support of the training data. In particular, we find that attributes can be shifted in proportion to the variability of that attribute in the training data.

One of the current criticisms of generative models is that they simply interpolate between datapoints, and fail to generate anything truly new. Our results add nuance to this story. It *is* possible to achieve distributional shift, where the model generates realistic images from a different distribution than that on which it was trained. However, the ability to steer the model in this way depends on the data distribution having sufficient diversity along the dimension we are steering.

Our main contributions are:

- We show a simple walk in the  $z$  space of GANs can achieve camera motion and color transformations in the output space, in self-supervised manner without labeled attributes or distinct source and target images.
- We show this linear walk is as effective as more complex non-linear walks, suggesting that GANs models learn to linearize these operations without being explicitly trained to do so.
- We design measures to quantify the extent and limits of the transformations generative models can achieve and show experimental evidence. We further show the

---

<sup>1</sup>We use the term “steerable” in analogy to the classic steerable filters of Freeman & Adelson [7].

relation between shiftability of model distribution and the variability in datasets.

- We demonstrate these transformations as a general-purpose framework on different model architectures, e.g. BigGAN and StyleGAN, illustrating different disentanglement properties in their respective latent spaces.

## 2. Related work

Walking in the latent space can be seen from different perspectives: how to achieve it, what limits it, and what does it enable us to do. Our work addresses these three aspects together, and we briefly refer to each one in prior work.

**Interpolations in GAN latent space** Traditional approaches to image editing in latent space involve finding linear directions in GAN latent space that correspond to changes in labeled attributes, such as smile-vectors and gender-vectors for faces [20, 13]. However smoothly varying latent space interpolations are not exclusive to GANs; in flow-based generative models, linear interpolations between encoded images allow one to edit a source image toward attributes contained in a separate target [16]. With Deep Feature Interpolation [24], one can remove the generative model entirely. Instead, the interpolation operation is performed on an intermediate feature space of a pretrained classifier, again using the feature mappings of source and target sets to determine an edit direction. Unlike these approaches, we learn our latent-space trajectories in a self-supervised manner without relying on labeled attributes or distinct source and target images. We measure the edit distance in pixel space rather than latent space, because our desired target image may not directly exist in the latent space of the generative model. Despite allowing for non-linear trajectories, we find that linear trajectories in the latent space model simple image manipulations – e.g., zoom-vectors and shift-vectors – surprisingly well, even when models were not explicitly trained to exhibit this property in latent space.

**Dataset bias** Biases from training data and network architecture both factor into the generalization capacity of learned models [23, 8, 1]. Dataset biases partly comes from human preferences in taking photos: we typically capture images in specific “canonical” views that are not fully representative of the entire visual world [19, 12]. When models are trained to fit these datasets, they inherit the biases in the data. Such biases may result in models that misrepresent the given task – such as tendencies towards texture bias rather than shape bias on ImageNet classifiers [8] – which in turn limits their generalization performance on similar

objectives [2]. Our latent space trajectories transform the output corresponding to various camera motion and image editing operations, but ultimately we are constrained by biases in the data and cannot extrapolate arbitrarily far beyond the dataset.

**Deep learning for content creation** The recent progress in generative models has enabled interesting applications for content creation [4, 13], including variants that enable end users to control and fine-tune the generated output [22, 26, 3]. A by-product the current work is to further enable users to modify various image properties by turning a single knob – the magnitude of the learned transformation. Similar to [26], we show that GANs allow users to achieve basic image editing operations by manipulating the latent space. However, we further demonstrate that these image manipulations are not just a simple creativity tool; they also provide us with a window into biases and generalization capacity of these models.

**Concurrent work** We note a few concurrent papers that also explore trajectories in GAN latent space. [6] learns linear walks in the latent space that correspond to various facial characteristics; they use these walks to measure biases in facial attribute detectors, whereas we study biases in the generative model that originate from training data. [21] also assumes linear latent space trajectories and learns paths for face attribute editing according to semantic concepts such as age and expression, thus demonstrating disentanglement properties of the latent space. [9] applies a linear walk to achieve transformations in learning and editing features that pertain cognitive properties of an image such as memorability, aesthetics, and emotional valence. Unlike these works, we do not require an attribute detector or assessor function to learn the latent space trajectory, and therefore our loss function is based on image similarity between source and target images. In addition to linear walks, we explore using non-linear walks to achieve camera motion and color transformations.

### 3. Method

Generative models such as GANs [10] learn a mapping function  $G$  such that  $G : z \rightarrow x$ . Here,  $z$  is the latent code drawn from a Gaussian density and  $x$  is an output, e.g., an image. Our goal is to achieve transformations in the output space by walking in the latent space, as shown in Fig. 2. In general, this goal also captures the idea in equivariance where transformations in the input space result in equivalent transformations in the output space (e.g., refer to [11, 5, 17]).

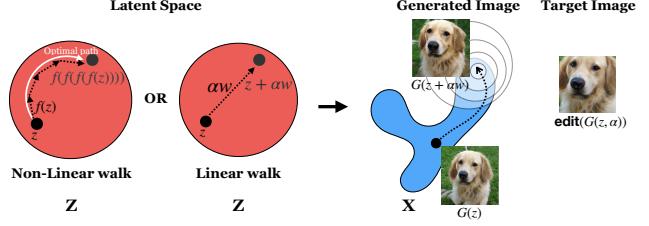


Figure 2: We aim to find a path in  $z$  space to transform the generated image  $G(z)$  to its edited version  $edit(G(z, \alpha))$ , e.g., an  $\alpha \times$  zoom. This walk results in the generated image  $G(z + \alpha w)$  when we choose a linear walk, or  $G(f(f(\dots(z)))$  when we choose a non-linear walk.

**Objective** How can we maneuver in the latent space to discover different transformations on a given image? We desire to learn a vector representing the optimal path for this latent space transformation, which we parametrize as an  $N$ -dimensional learned vector. We weight this vector by a continuous parameter  $\alpha$  which signifies the step size of the transformation: large  $\alpha$  values correspond to a greater degree of transformation, while small  $\alpha$  values correspond to a lesser degree. Formally, we learn the walk  $w$  by minimizing the objective function:

$$w^* = \underset{w}{\operatorname{argmin}} \mathbb{E}_{z, \alpha} [\mathcal{L}(G(z + \alpha w), edit(G(z, \alpha)))] \quad (1)$$

Here,  $\mathcal{L}$  measures the distance between the generated image after taking an  $\alpha$ -step in the latent direction  $G(z + \alpha w)$  and the target  $edit(G(z), \alpha)$  derived from the source image  $G(z)$ . We use  $L2$  loss as our objective  $\mathcal{L}$ , however we also obtain similar results when using the LPIPS perceptual image similarity metric [25] (see B.4.1). Note that we can learn this walk in a fully self-supervised manner – we perform our desired transformation on an arbitrary generated image, and subsequently we optimize our walk to satisfy the objective. We learn the optimal walk  $w^*$  to model the transformation applied in  $edit(\cdot)$ . Let  $\text{model}(\alpha)$  denote such transformation in the direction of  $w^*$ , controlled with the variable step size  $\alpha$ , defined as  $\text{model}(\alpha) = G(z + \alpha w^*)$ .

The previous setup assumes linear latent space walks, but we can also optimize for non-linear trajectories in which the walk direction depends on the current latent space position. For the non-linear walk, we learn a function,  $f^*(z)$ , which corresponds to a small  $\epsilon$ -step transformation  $edit(G(z), \epsilon)$ . To achieve bigger transformations, we learn to apply  $f$  recursively, mimicking discrete Euler ODE approximations. Formally, for a fixed  $\epsilon$ , we minimize

$$\mathcal{L} = \mathbb{E}_{z, n} [| | f^n(z) - edit(G(z, n\epsilon)) | |], \quad (2)$$

where  $f^n(\cdot)$  is an  $n$ th-order function composition  $f(f(f(\dots)))$ , and  $f(z)$  is parametrized with a neural network. We discuss further implementation details in A.4.

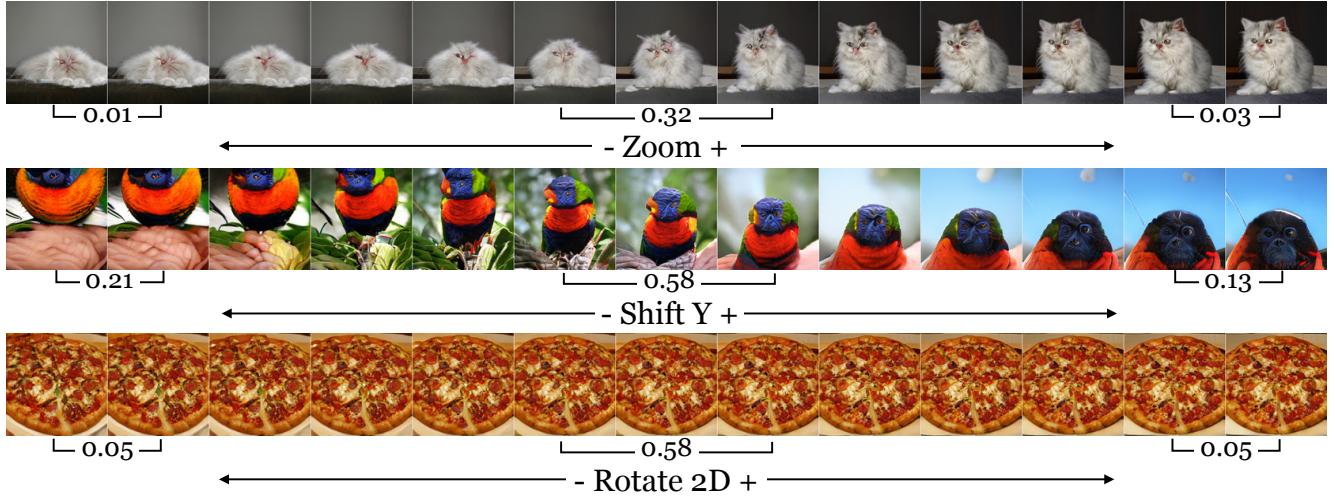


Figure 3: Transformation limits. As we increase the magnitude of the learned walk vector, we reach a limit to how much we can achieve a given transformation. The learned operation either ceases to transform the image any further, or the image starts to deviate from the natural image manifold. Below each figure we also indicate the average LPIPS perceptual distance between 200 sampled image pairs of that category. Perceptual distance decreases as we move farther from the source (center image), which indicates that the images are converging.

We use this function composition approach rather than the simpler setup of  $G(z + \alpha \text{NN}(z))$  because the latter learns to ignore the input  $z$  when  $\alpha$  takes on continuous values, and is thus equivalent to the previous linear trajectory (see A.3 for further details).

**Quantifying Steerability** We further seek to quantify the extent to which we are able to achieve the desired image manipulations with respect to each transformation. To this end, we compare the distribution of a given attribute, e.g., ‘‘luminance’’, in the dataset versus in images generated after walking in latent space.

For color transformations, we consider the effect of increasing or decreasing the  $\alpha$  coefficient corresponding to each color channel. To estimate the color distribution of model-generated images, we randomly sample  $N = 100$  pixels per image both before and after taking a step in latent space. Then, we compute the pixel value for each channel, or the mean RGB value for luminance, and normalize the range between 0 and 1.

For zoom and shift transformations, we rely on an object detector which captures the central object in the image class. We use a *MobileNet-SSD v1* [18] detector to estimate object bounding boxes, and restrict ourselves to image classes recognizable by the detector. For each successful detection, we take the highest probability bounding box corresponding to the desired class and use that to quantify the degree to which we are able to achieve each transformation. For the zoom operation, we use the area of the bounding box normalized by the area of the total image. For shift in

the X and Y directions, we take the center X and Y coordinates of the bounding box, and normalize by image width or height.

Truncation parameters in GANs (as used in [4, 13]) trade off between variety of the generated images and sample quality. When comparing generated images to the dataset distribution, we use the largest possible truncation we use the largest possible truncation for the model and perform similar cropping and resizing of the dataset as done during model training (see [4]). When comparing the attributes of generated distributions under different  $\alpha$  magnitudes to each other but not to the dataset, we reduce truncation to 0.5 to ensure better performance of the object detector.

## 4. Experiments

We demonstrate our approach using BigGAN [4], a class-conditional GAN trained on 1000 ImageNet categories. We learn a shared latent space walk by averaging across the image categories, and further quantify how this walk affects each class differently. We focus on linear walks in latent space for the main text, and show additional results on nonlinear walks in Sec. 4.3 and B.4.2. We also conduct experiments on StyleGAN [13], which uses an unconditional style-based generator architecture in Sec. 4.3 and B.5.

### 4.1. What image transformations can we achieve by steering in the latent space?

We show qualitative results of the learned transformations in Fig. 1. By walking in the generator latent space, we are able to learn a variety of transformations on a given

source image (shown in the center image of each transformation). Interestingly, several priors come into play when learning these image transformations. When we shift a daisy downwards in the Y direction, the model hallucinates that the sky exists on the top of the image. However, when we shift the daisy up, the model inpaints the remainder of the image with grass. When we alter the brightness of an image, the model transitions between nighttime and daytime. This suggests that the model is able to extrapolate from the original source image, and yet remain consistent with the image context.

We further ask the question, how much are we able to achieve each transformation? When we increase the step size of  $\alpha$ , we observe that the extent to which we can achieve each transformation is limited. In Fig. 3 we observe two potential failure cases: one in which the image becomes unrealistic, and the other in which the image fails to transform any further. When we try to zoom in on a Persian cat, we observe that the cat no longer increases in size beyond some point. In fact, the size of the cat under the latent space transformation consistently undershoots the target zoom. On the other hand, when we try to zoom out on the cat, we observe that it begins to fall off the image manifold, and at some point is unable to become any smaller. Indeed, the perceptual distance (using LPIPS) between images decreases as we push  $\alpha$  towards the transformation limits. Similar trends hold with other transformations: we are able to shift a lorikeet up and down to some degree until the transformation fails to have any further effect and yields unrealistic output, and despite adjusting  $\alpha$  on the rotation vector, we are unable to rotate a pizza. Are the limitations to these transformations governed by the training dataset? In other words, are our latent space walks limited because in ImageNet photos the cats are mostly centered and taken within a certain size? We seek to investigate and quantify these biases in the next sections.

An intriguing property of the learned trajectory is that the degree to which it affects the output depends on the image class. In Fig. 4, we investigate the impact of the walk for different image categories under color transformations. By stepping  $w$  in the direction of increasing redness in the image, we are able to successfully recolor a jellyfish, but we are unable to change the color of a goldfinch – the goldfinch remains yellow and the only effect of  $w$  is to slightly redder the background. Likewise, increasing the brightness of a scene transforms an erupting volcano to a dormant volcano, but does not have such effect on Alps, which only transitions between night and day. In the third example, we use our latent walk to turn red sports cars to blue. When we apply this vector to firetrucks, the only effect is to slightly change the color of the sky. Again, perceptual distance over multiple image samples per class confirms these qualitative observations: a 2-sample  $t$ -test yields  $t = 20.77, p < 0.001$

for jellyfish/goldfinch,  $t = 8.14, p < 0.001$  for volcano/alp, and  $t = 6.84, p < 0.001$  for sports car/fire engine. We hypothesize that the differential impact of the same transformation on various image classes relates to the variability in the underlying dataset. Firetrucks only appear in red, but sports cars appear in a variety of colors. Therefore, our color transformation is constrained by the biases exhibited in individual classes in the dataset.

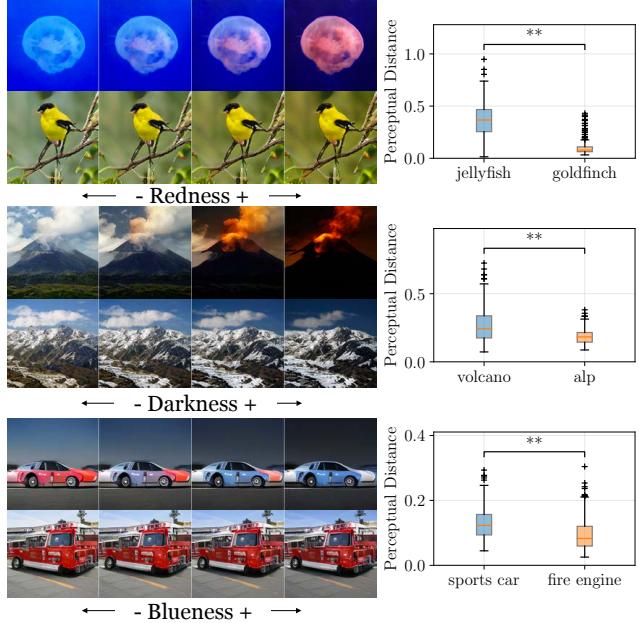


Figure 4: Each pair of rows shows how a single latent direction  $w$  affects two different ImageNet classes. We observe that the same direction affects different classes differently, and in a way consistent with semantic priors (e.g., “Volcanoes” explode, “Alps” do not). Boxplots show the LPIPS perceptual distance before and after transformation for 200 samples per class.

Motivated by our qualitative observations in Fig. 1, Fig. 3, and Fig. 4, we next quantitatively investigate the degree to which we are able to achieve certain transformations.

**Limits of Transformations:** In quantifying the extent of our transformations, we are bounded by two opposing factors: how much we can transform the distribution while also remaining within the manifold of realistic looking images.

For the color (luminance) transformation, we observe that by changing  $\alpha$ , we shift the proportion of light and dark colored pixels, and can do so without large increases in FID (Fig. 5 first column). However, the extent to which we can move the distribution is limited, and each transformed distribution still retains some probability mass over the full range of pixel intensities.

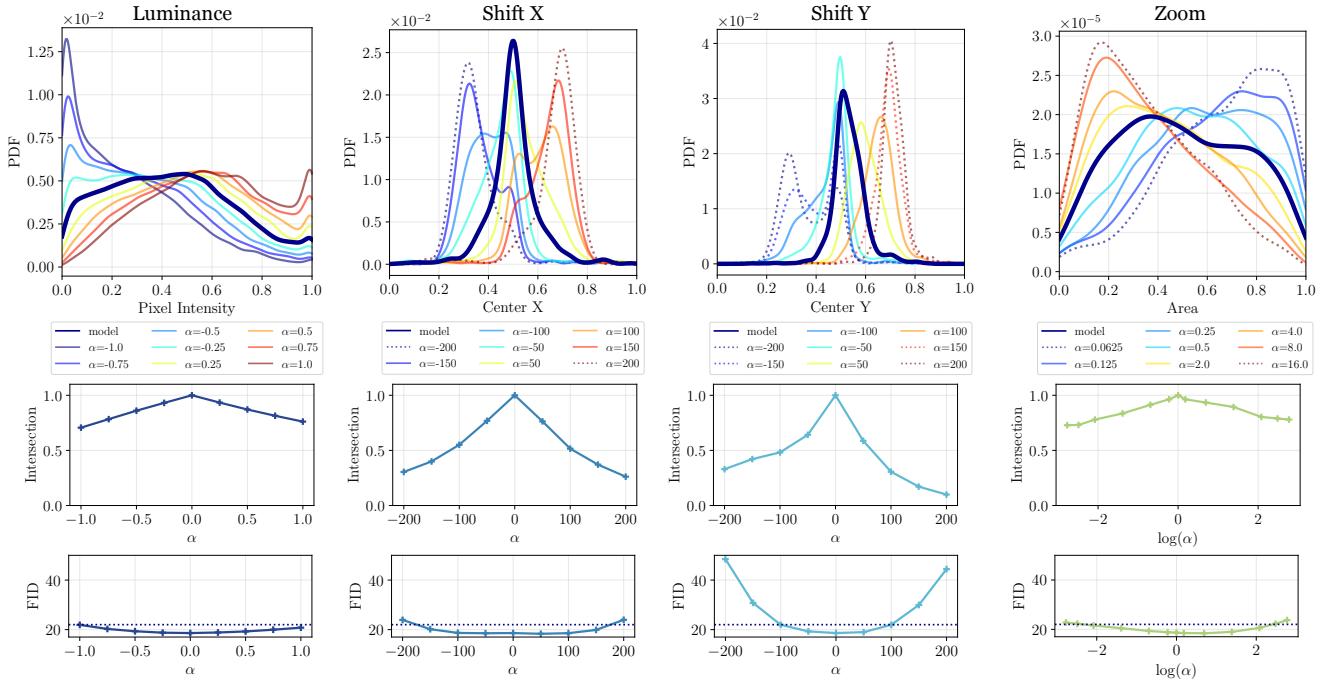


Figure 5: Quantifying the extent of transformations. We compare the attributes of generated images under the raw model output  $G(z)$ , compared to the distribution under a learned transformation  $\text{model}(\alpha)$ . We measure the intersection between  $G(z)$  and  $\text{model}(\alpha)$ , and also compute the FID on the transformed image to limit our transformations to the natural image manifold.

Using the shift vector, we show that we are able to move the distribution of the center object by changing  $\alpha$ . In the underlying model, the center coordinate of the object is most concentrated at half of the image width and height, but after applying the shift in X and shift in Y transformation, the mode of the transformed distribution varies between 0.3 and 0.7 of the image width/height. To quantify the extent to which the distribution changes, we compute the area of intersection between the original model distribution and the distribution after applying the transformation, and we observe that the intersection decreases as we increase or decrease the magnitude of  $\alpha$ . However, our transformations are limited to a certain extent – if we increase  $\alpha$  beyond 150 pixels for vertical shifts, we start to generate unrealistic images, as evidenced by a sharp rise in FID and converging modes in the transformed distributions (Fig. 5 columns 2 & 3).

We perform a similar procedure for the zoom operation, by measuring the area of the bounding box for the detected object under different magnitudes of  $\alpha$ . Like shift, we observe that subsequent increases in  $\alpha$  magnitude start to have smaller and smaller effects on the mode of the resulting distribution (Fig. 5 last column). Past an 8x zoom in or out, we observe an increase in the FID of the transformed output, signifying decreasing image quality. Interestingly for

zoom, the FID under zooming in and zooming out is anti-symmetric, indicating that the success to which we are able to achieve the zoom-in operation under the constraint of realistic looking images differs from that of zooming out.

These trends are consistent with the plateau in transformation behavior that we qualitatively observe in Fig. 3. Although we can arbitrarily increase the  $\alpha$  step size, after some point we are unable to achieve further transformation and risk deviating from the natural image manifold.

## 4.2. How does the data affect the transformations?

How do the statistics of the data distribution affect the ability to achieve these transformations? In other words, is the extent to which we can transform each class, as we observed in Fig. 4, due to limited variability in the underlying dataset for each class?

One intuitive way of quantifying the amount of transformation achieved, dependent on the variability in the dataset, is to measure the difference in transformed model means,  $\text{model}(+\alpha)$  and  $\text{model}(-\alpha)$ , and compare it to the spread of the dataset distribution. For each class, we compute standard deviation of the dataset with respect to our statistic of interest (pixel RGB value for color, and bounding box area and center value for zoom and shift transformations respectively). We hypothesize that if the amount of

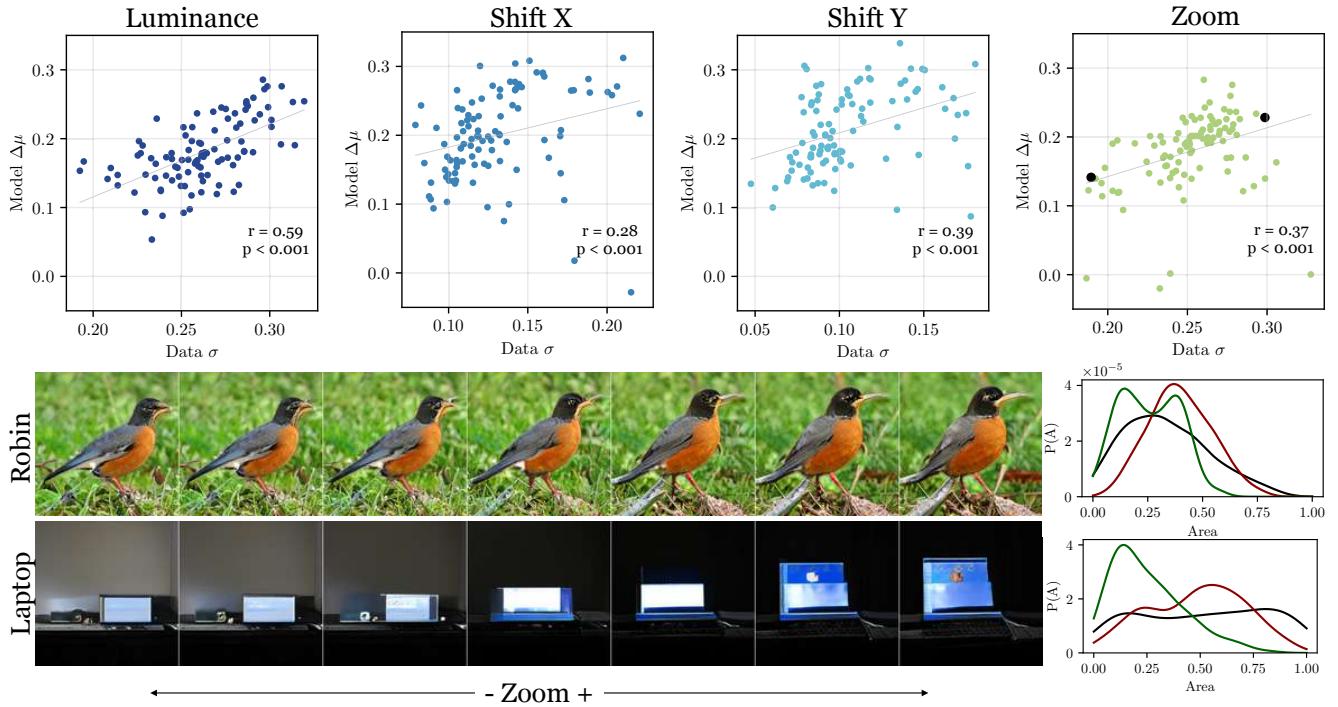


Figure 6: Understanding per-class biases. We observe a correlation between the variability in the training data for ImageNet classes, and our ability to shift the distribution under latent space transformations. Classes with low variability (e.g., robin) limit our ability to achieve desired transformations, in comparison to classes with a broad dataset distribution (e.g., laptop).

transformation is biased depending on the image class, we will observe a correlation between the distance of the mean shifts and the standard deviation of the data distribution.

More concretely, we define the change in model means under a given transformation as:

$$\Delta\mu_k = \|\mu_{k,\text{model}(+\alpha^*)} - \mu_{k,\text{model}(-\alpha^*)}\|_2, \quad (3)$$

for a given class  $k$  under the optimal walk for each transformation  $w^*$  under Eq. 5, and we set  $\alpha^*$  to be largest and smallest  $\alpha$  values used in training. We note that the degree to which we achieve each transformation is a function of  $\alpha$ , so we use the same value of  $\alpha$  across all classes – one that is large enough to separate the means of  $\mu_{k,\text{model}(\alpha^*)}$  and  $\mu_{k,\text{model}(-\alpha^*)}$  under transformation, but also for which the FID of the generated distribution remains below a threshold  $T$  of generating reasonably realistic images (for our experiments we use  $T = 22$ ).

We apply this measure in Fig. 6, where on the x-axis we plot the standard deviation  $\sigma$  of the dataset, and on the y-axis we show the model  $\Delta\mu$  under a  $+\alpha^*$  and  $-\alpha^*$  transformation as defined in Eq. 3. We sample randomly from 100 classes for the color, zoom and shift transformations, and generate 200 samples of each class under the positive and negative transformations. We use the same setup of drawing samples from the model and dataset and computing the

statistics for each transformation as described in Sec. 4.1.

Indeed, we find that the width of the dataset distribution, captured by the standard deviation of random samples drawn from the dataset for each class, is related to the extent to which we achieve each transformation. For these transformations, we observe a positive correlation between the spread of the dataset, and the magnitude of  $\Delta\mu$  observed in the transformed model distributions. Furthermore, the slope of all observed trends differs significantly from zero ( $p < 0.001$  for all transformations).

For the zoom transformation, we show examples of two extremes along the trend. We show an example of the “robin” class, in which the spread  $\sigma$  in the dataset is low, and subsequently, the separation  $\Delta\mu$  that we are able to achieve by applying  $+\alpha^*$  and  $-\alpha^*$  transformations is limited. On the other hand, in the “laptop” class, the underlying spread in the dataset is broad; ImageNet contains images of laptops of various sizes, and we are able to attain a wider shift in the model distribution when taking a  $\alpha^*$ -sized step in the latent space.

From these results, we conclude that the extent to which we achieve each transformation is related to the variability that inherently exists in the dataset. Consistent with our qualitative observations in Fig. 4, we find that if the images for a particular class have adequate coverage over the

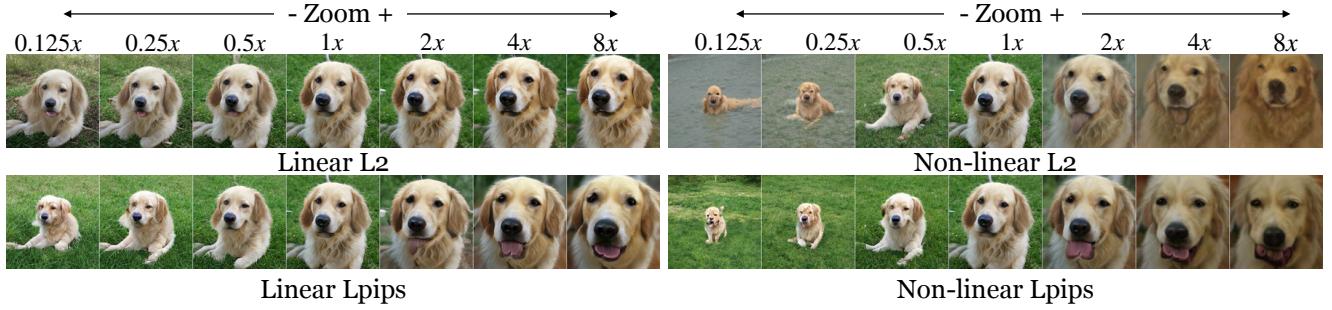


Figure 7: Comparison of linear and nonlinear walks for the zoom operation. The linear walk undershoots the targeted level of transformation, but maintains more realistic output.

entire range of a given transformation, then we are better able to move the model distribution to both extremes. On the other hand, if the images for the given class exhibit less variability, the extent of our transformation is limited by this property of the dataset.

### 4.3. Alternative architectures and walks

We ran an identical set of experiments using the nonlinear walk in the BigGAN latent space, and obtain similar quantitative results. To summarize, the Pearson’s correlation coefficient between dataset  $\sigma$  and model  $\Delta\mu$  for linear walks and nonlinear walks is shown in Table 1, and full results in B.4.2. Qualitatively, we observe that while the linear trajectory undershoots the targeted level of transformation, it is able to preserve more realistic-looking results (Fig. 7). The transformations involve a trade-off between minimizing the loss and maintaining realistic output, and we hypothesize that the linear walk functions as an implicit regularizer that corresponds well with the inherent organization of the latent space.

	Luminance	Shift X	Shift Y	Zoom
Linear	0.59	0.28	0.39	0.37
Non-linear	0.49	0.49	0.55	0.60

Table 1: Pearson’s correlation coefficient between dataset  $\sigma$  and model  $\Delta\mu$  for measured attributes. p-value for slope  $< 0.001$  for all transformations.

To test the generality of our findings across model architecture, we ran similar experiments on StyleGAN, in which the latent space is divided into two spaces,  $z$  and  $W$ . As [13] notes that the  $W$  space is less entangled than  $z$ , we apply the linear walk to  $W$  and show results in Fig. 8 and B.5. One interesting aspect of StyleGAN is that we can change color while leaving other structure in the image unchanged. In other words, while green faces do not naturally exist in the dataset, the StyleGAN model is still able to generate them. This differs from the behavior of BigGAN, where changing

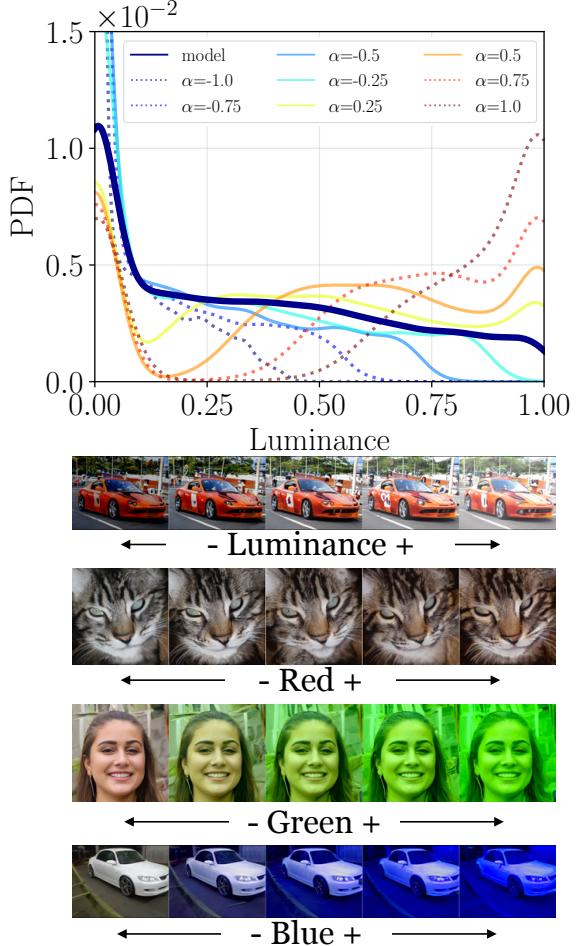


Figure 8: Distribution for luminance transformation learned from the StyleGAN cars generator, and qualitative examples of color transformations on various datasets using StyleGAN.

color results in different semantics in the image, e.g., turning a dormant volcano to an active one or a daytime scene to a nighttime one. StyleGAN, however, does not preserve the exact geometry of objects under other transformations, e.g., zoom and shift (see Sec. B.5).

## 5. Conclusion

GANs are powerful generative models, but are they simply replicating the existing training datapoints, or are they able to generalize outside of the training distribution? We investigate this question by exploring walks in the latent space of GANs. We optimize trajectories in latent space to reflect simple image transformations in the generated output, learned in a self-supervised manner. We find that the model is able to exhibit characteristics of extrapolation - we are able to “steer” the generated output to simulate camera zoom, horizontal and vertical movement, camera rotations, and recolorization. However, our ability to shift the distribution is finite: we can transform images to some degree but cannot extrapolate entirely outside the support of the training data. Because training datasets are biased in capturing the most basic properties of the natural visual world, transformations in the latent space of generative models can only do so much.

## Acknowledgements

We would like to thank Quang H Le, Lore Goetschalckx, Alex Andonian, David Bau, and Jonas Wulff for helpful discussions. This work was supported by a Google Faculty Research Award to P.I., and a U.S. National Science Foundation Graduate Research Fellowship to L.C.

## References

- [1] A. Amini, A. Soleimany, W. Schwarting, S. Bhatia, and D. Rus. Uncovering and mitigating algorithmic bias through learned latent structure. [2](#)
- [2] A. Azulay and Y. Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, 2018. [2](#)
- [3] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597*, 2018. [3](#)
- [4] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. [1, 3, 4](#)
- [5] T. S. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling. Gauge equivariant convolutional networks and the icosahedral cnn. *arXiv preprint arXiv:1902.04615*, 2019. [3](#)
- [6] E. Denton, B. Hutchinson, M. Mitchell, and T. Gebru. Detecting bias with generative counterfactual face attribute augmentation. *arXiv preprint arXiv:1906.06439*, 2019. [3](#)
- [7] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):891–906, 1991. [2](#)
- [8] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018. [2](#)
- [9] L. Goetschalckx, A. Andonian, A. Oliva, and P. Isola. Ganalyze: Toward visual definitions of cognitive image properties. *arXiv preprint arXiv:1906.10112*, 2019. [3](#)
- [10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. [1, 3](#)
- [11] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011. [3](#)
- [12] A. Jahanian, S. Vishwanathan, and J. P. Allebach. Learning visual balance from large-scale datasets of aesthetically highly rated images. In *Human Vision and Electronic Imaging XX*, volume 9394, page 93940Y. International Society for Optics and Photonics, 2015. [2](#)
- [13] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018. [1, 2, 3, 4, 8, 12](#)
- [14] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009. [24](#)
- [15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. [10](#)
- [16] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018. [2](#)
- [17] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015. [3](#)
- [18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. [4](#)
- [19] E. Mezuman and Y. Weiss. Learning about canonical views from internet image collections. In *Advances in neural information processing systems*, pages 719–727, 2012. [2](#)
- [20] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. [2](#)
- [21] Y. Shen, J. Gu, J.-t. Huang, X. Tang, and B. Zhou. Interpreting the latent space of gans for semantic face editing. *arXiv preprint*, 2019. [3, 12](#)
- [22] J. Simon. Ganbreeder. <http://https://ganbreeder.app/>, accessed 2019-03-22. [3](#)
- [23] A. Torralba and A. A. Efros. Unbiased look at dataset bias. 2011. [2](#)
- [24] P. Upchurch, J. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Weinberger. Deep feature interpolation for image content changes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7064–7073, 2017. [2](#)
- [25] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. [3, 11](#)
- [26] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016. [3](#)

## Appendix

### A. Method details

#### A.1. Optimization for the linear walk

We learn the walk vector using mini-batch stochastic gradient descent with the Adam optimizer [15] in tensorflow, trained on 20000 unique samples from the latent space  $z$ . We share the vector  $w$  across all ImageNet categories for the BigGAN model.

#### A.2. Implementation Details for Linear Walk

We experiment with a number of different transformations learned in the latent space, each corresponding to a different walk vector. Each of these transformations can be learned without any direct supervision, simply by applying our desired edit to the source image. Furthermore, the parameter  $\alpha$  allows us to vary the extent of the transformation. We found that a slight modification to each transformation improved the degree to which we were able to steer the output space: we scale  $\alpha$  differently for the learned transformation  $G(z + \alpha_g w)$ , and the target edit  $\text{edit}(G(z), \alpha_t)$ . We detail each transformation below:

**Shift.** We learn transformations corresponding to shifting an image in the horizontal X direction and the vertical Y direction. We train on source images that are shifted  $-\alpha_t$  pixels to the left and  $\alpha_t$  pixels to the right, where we set  $\alpha_t$  to be between zero and one-half of the source image width or height  $D$ . When training the walk, we enforce that the  $\alpha_g$  parameter ranges between -1 and 1; thus for a random shift by  $t$  pixels, we use the value  $\alpha_g = \alpha_t/D$ . We apply a mask to the shifted image, so that we only apply the loss function on the visible portion of the source image. This forces the generator to extrapolate on the obscured region of the target image.

**Zoom.** We learn a walk which is optimized to zoom in and out up to four times the original image. For zooming in, we crop the central portion of the source image by some  $\alpha_t$  amount, where  $0.25 < \alpha_t < 1$  and resize it back to its original size. To zoom out, we downsample the image by  $\alpha_t$  where  $1 < \alpha_t < 4$ . To allow for both a positive and negative walk direction, we set  $\alpha_g = \log(\alpha_t)$ . Similar to shift, a mask applied during training allows the generator to inpaint the background scene.

**Color.** We implement color as a continuous RGB slider, e.g., a 3-tuple  $\alpha_t = (\alpha_R, \alpha_G, \alpha_B)$ , where each  $\alpha_R, \alpha_G, \alpha_B$  can take values between  $[-0.5, 0.5]$  in training. To edit the source image, we simply add the corresponding  $\alpha_t$  values to each of the image channels. Our latent space walk is parameterized as  $z + \alpha_g w = z + \alpha_R w_R + \alpha_G w_G + \alpha_B w_B$

where we jointly learn the three walk directions  $w_R, w_G$ , and  $w_B$ .

**Rotate in 2D.** Rotation in 2D is trained in a similar manner as the shift operations, where we train with  $-45 \leq \alpha_t \leq 45$  degree rotation. Using  $R = 45$ , scale  $\alpha_g = \alpha_t/R$ . We use a mask to enforce the loss only on visible regions of the target.

**Rotate in 3D.** We simulate a 3D rotation using a perspective transformation along the Z-axis, essentially treating the image as a rotating billboard. Similar to the 2D rotation, we train with  $-45 \leq \alpha_t \leq 45$  degree rotation, we scale  $\alpha_g = \alpha_t/R$  where  $R = 45$ , and apply a mask during training.

#### A.3. Linear $\text{NN}(z)$ walk

Rather than defining  $w$  as a vector in  $z$  space (Eq. 1), one could define it as a function that takes a  $z$  as input and maps it to the desired  $z'$  after taking a variable-sized step  $\alpha$  in latent space. In this case, we may parametrize the walk with a neural network  $w = \text{NN}(z)$ , and transform the image using  $G(z + \alpha \text{NN}(z))$ . However, as we show in the following proof, this idea will not learn to let  $w$  be a function of  $z$ .

*Proof.* For simplicity, let  $w = F(z)$ . We optimize for  $J(w, \alpha) = \mathbb{E}_z [\mathcal{L}(G(z + \alpha w), \text{edit}(G(z), \alpha))]$  where  $\alpha$  is an arbitrary scalar value. Note that for the target image, two equal edit operations is equivalent to performing a single edit of twice the size (e.g., shifting by 10px the same as shifting by 5px twice; zooming by 4x is the same as zooming by 2x twice). That is,

$$\text{edit}(G(z), 2\alpha) = \text{edit}(\text{edit}(G(z), \alpha), \alpha).$$

To achieve this target, starting from an initial  $z$ , we can take two steps of size  $\alpha$  in latent space as follows:

$$\begin{aligned} z_1 &= z + \alpha F(z) \\ z_2 &= z_1 + \alpha F(z_1) \end{aligned}$$

However, because we let  $\alpha$  take on any scalar value during optimization, our objective function enforces that starting from  $z$  and taking a step of size  $2\alpha$  equals taking two steps of size  $\alpha$ :

$$z + 2\alpha F(z) = z_1 + \alpha F(z_1) \quad (4)$$

Therefore:

$$\begin{aligned} z + 2\alpha F(z) &= z + \alpha F(z) + \alpha F(z_1) \Rightarrow \\ \alpha F(z) &= \alpha F(z_1) \Rightarrow \\ F(z) &= F(z_1). \end{aligned}$$

Thus  $F(\cdot)$  simply becomes a linear trajectory that is independent of the input  $z$ .  $\square$

#### A.4. Optimization for the non-linear walk

Given the limitations of the previous walk, we define our nonlinear walk  $F(z)$  using discrete step sizes  $\epsilon$ . We define  $F(z)$  as  $z + \text{NN}(z)$ , where the neural network  $\text{NN}$  learns a fixed  $\epsilon$  step transformation, rather than a variable  $\alpha$  step. We then renormalize the magnitude  $z$ . This approach mimics the Euler method for solving ODEs with a discrete step size, where we assume that the gradient of the transformation in latent space is of the form  $\epsilon \frac{dz}{dt} = \text{NN}(z)$  and we approximate  $z_{i+1} = z_i + \epsilon \frac{dz}{dt}|_{z_i}$ . The key difference from A.3 is the fixed step size, which avoids optimizing for the equality in (4).

We use a two-layer neural network to parametrize the walk, and optimize over 20000 samples using the Adam optimizer as before. Positive and negative transformation directions are handled with two neural networks having identical architecture but independent weights. We set  $\epsilon$  to achieve the same transformation ranges as the linear trajectory within 4-5 steps.

### B. Additional Experiments

#### B.1. Model and Data Distributions

How well does the model distribution of each property match the dataset distribution? If the generated images do not form a good approximation of the dataset variability, we expect that this would also impact our ability to transform generated images. In Fig. 9 we show the attribute distributions of the BigGAN model  $G(z)$  compared to samples from the ImageNet dataset. We show corresponding results for StyleGAN and its respective datasets in Sec. B.5. While there is some bias in how well model-generated images approximated the dataset distribution, we hypothesize that additional biases in our transformations come from variability in the training data.

#### B.2. Quantifying Transformation Limits

We observe that when we increase the transformation magnitude  $\alpha$  in latent space, the generated images become unrealistic and the transformation ceases to have further effect. We show this qualitatively in Fig. 3. To quantitatively verify this trends, we can compute the LPIPS perceptual distance of images generated using consecutive pairs of  $\alpha_i$  and  $\alpha_{i+1}$ . For shift and zoom transformations, perceptual distance is larger when  $\alpha$  (or  $\log(\alpha)$  for zoom) is near zero, and decreases as the the magnitude of  $\alpha$  increases, which indicates that large  $\alpha$  magnitudes have a smaller transformation effect, and the transformed images appear more similar. On the other hand, color and rotate in 2D/3D exhibit a steady transformation rate as the magnitude of  $\alpha$  increases.

Note that this analysis does not tell us how well we achieve the specific transformation, nor whether the latent

trajectory deviates from natural-looking images. Rather, it tells us how much we manage to change the image, regardless of the transformation target. To quantify how well each transformation is achieved, we rely on attribute detectors such as object bounding boxes (see B.3).

#### B.3. Detected Bounding Boxes

To quantify the degree to which we are able to achieve the zoom and shift transformations, we rely on a pre-trained *MobileNet-SSD v1*<sup>2</sup> object detection model. In Fig. 11 and 12 we show the results of applying the object detection model to images from the dataset, and images generated by the model under the zoom, horizontal shift, and vertical shift transformations for randomly selected values of  $\alpha$ , to qualitatively verify that the object detection boundaries are reasonable. Not all ImageNet images contain recognizable objects, so we only use ImageNet classes containing objects recognizable by the detector for this analysis.

#### B.4. Alternative Walks in BigGAN

##### B.4.1 LPIPS objective

In the main text, we learn the latent space walk  $w$  by minimizing the objective function:

$$J(w, \alpha) = \mathbb{E}_z [\mathcal{L}(G(z + \alpha w), \text{edit}(G(z), \alpha))]. \quad (5)$$

using a Euclidean loss for  $\mathcal{L}$ . In Fig. 13 we show qualitative results using the LPIPS perceptual similarity metric [25] instead of Euclidean loss. Walks were trained using the same parameters as those in the linear-L2 walk shown in the main text: we use 20k samples for training, with Adam optimizer and learning rate 0.001 for zoom and color, 0.0001 for the remaining edit operations (due to scaling of  $\alpha$ ).

##### B.4.2 Non-linear Walks

Following B.4.2, we modify our objective to use discrete step sizes  $\epsilon$  rather than continuous steps. We learn a function  $F(z)$  to perform this  $\epsilon$ -step transformation on given latent code  $z$ , where  $F(z)$  is parametrized with a neural network. We show qualitative results in Fig. 14. We perform the same set of experiments shown in the main text using this nonlinear walk in Fig. 15. These experiments exhibit similar trends as we observed in the main text – we are able to modify the generated distribution of images using latent space walks, and the amount to which we can transform is related to the variability in the dataset. However, there are greater increases in FID when we apply the non-linear transformation, suggesting that these generated images deviate more from natural images and look less realistic.

<sup>2</sup><https://github.com/opencv/opencv/wiki/TensorFlow-Object-Detection-API>

### B.4.3 Additional Qualitative Examples

We show qualitative examples for randomly generated categories for BigGAN linear-L2, linear LPIPS, and nonlinear trajectories in Figs. 16, 17, 18 respectively.

### B.5. Walks in StyleGAN

We perform similar experiments for linear latent space walks using StyleGAN models trained on the LSUN cat, LSUN car, and FFHQ face datasets. As suggested by [13], we learn the walk vector in the intermediate  $W$  latent space due to improved attribute disentanglement in  $W$ . We show qualitative results for color, shift, and zoom transformations in Figs. 19, 21, 23 and corresponding quantitative analyses

in Figs. 20, 22, 24.

### B.6. Qualitative examples for additional transformations

Since the color transformation operates on individual pixels, we can optimize the walk using a segmented target – for example when learning a walk for cars, we only modify pixels in segmented car region when generating  $\text{edit}(G(z), \alpha)$ . StyleGAN is able to roughly localize the color transformation to this region, suggesting disentanglement of different objects within the  $W$  latent space (Fig. 25 left) as also noted in [13, 21]. We also show qualitative results for adjust image contrast (Fig. 25 right), and for combining zoom, shift X, and shift Y transformations (Fig. 26).

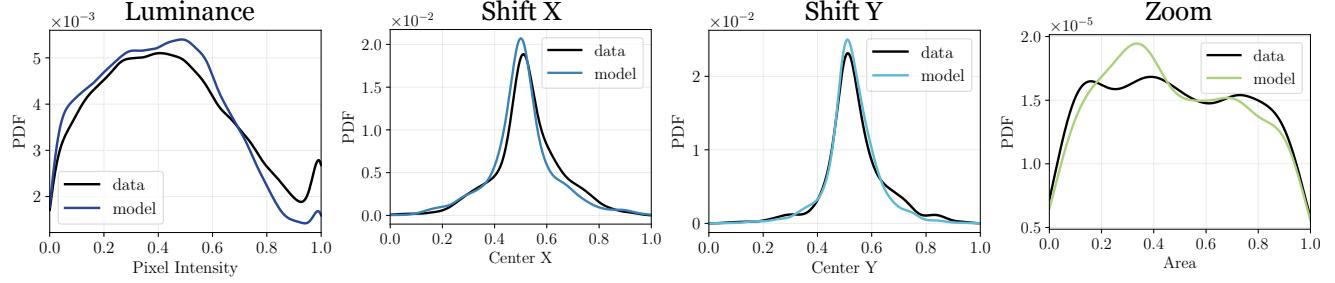


Figure 9: Comparing model versus dataset distribution. We plot statistics of the generated under the color (luminance), zoom (object bounding box size), and shift operations (bounding box center), and compare them to the statistics of images in the training dataset.

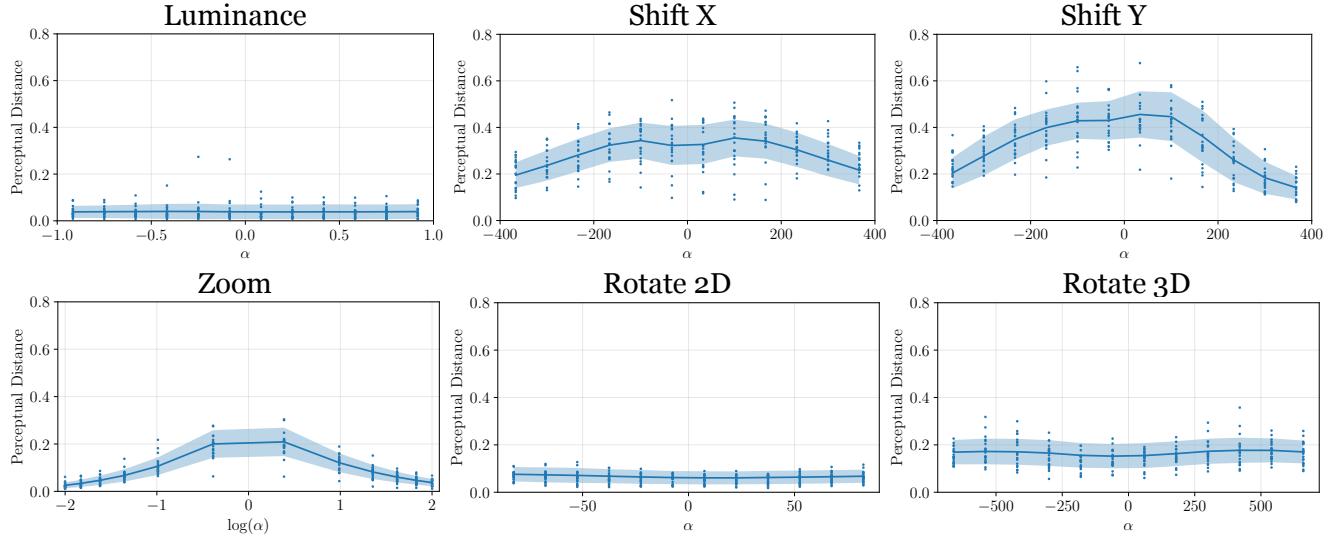


Figure 10: LPIPS Perceptual distances between images generated from pairs of consecutive  $\alpha_i$  and  $\alpha_{i+1}$ . We sample 1000 images from randomly selected categories using BigGAN, transform them according to the learned linear trajectory for each transformation. We plot the mean perceptual distance and one standard deviation across the 1000 samples (shaded area), as well as 20 individual samples (scatterplot). Because the Rotate 3D operation undershoots the targeted transformation, we observe more visible effects when we increase the  $\alpha$  magnitude.

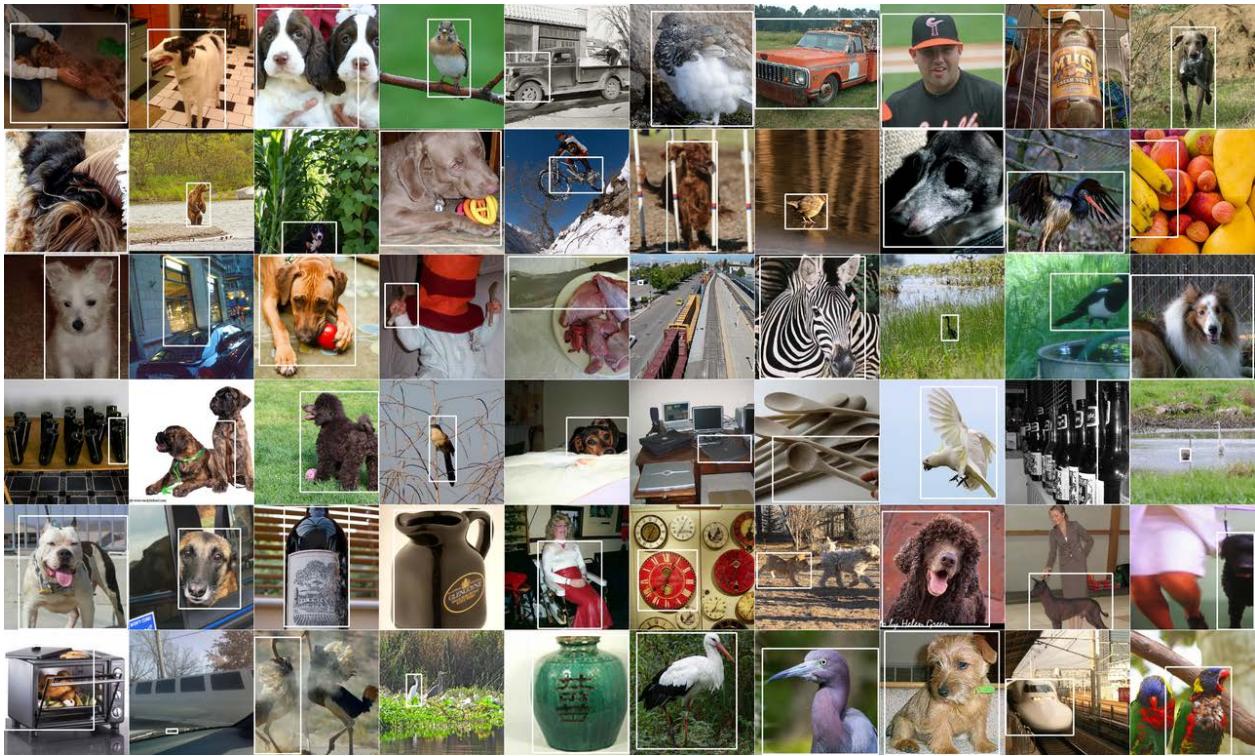


Figure 11: Bounding boxes for random selected classes using ImageNet training images.

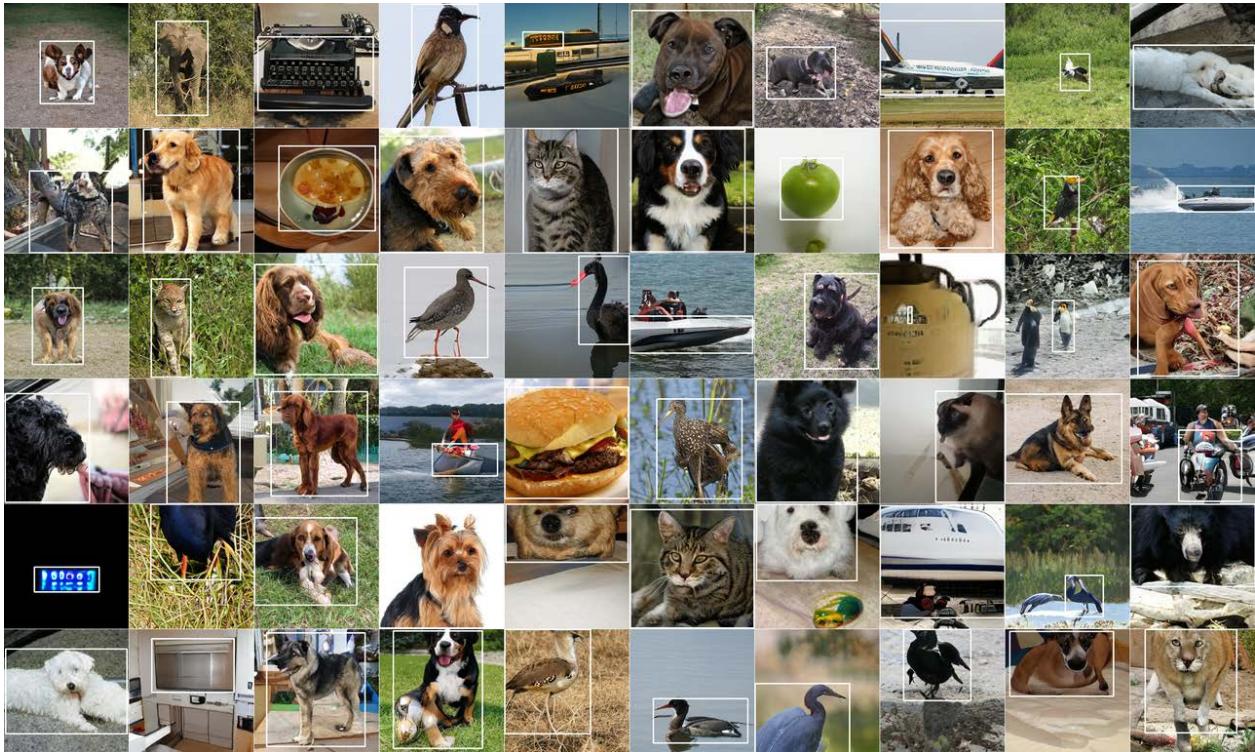


Figure 12: Bounding boxes for random selected classes using model-generated images for zoom and horizontal and vertical shift transformations under random values of  $\alpha$ .



Figure 13: Linear walks in BigGAN, trained to minimize LPIPS loss. For comparison, we show the same samples as in Fig. 1 (which used a linear walk with L2 loss).



Figure 14: Nonlinear walks in BigGAN, trained to minimize L2 loss for color and LPIPS loss for the remaining transformations. For comparison, we show the same samples in Fig. 1 (which used a linear walk with L2 loss), replacing the linear walk vector  $w$  with a nonlinear walk.

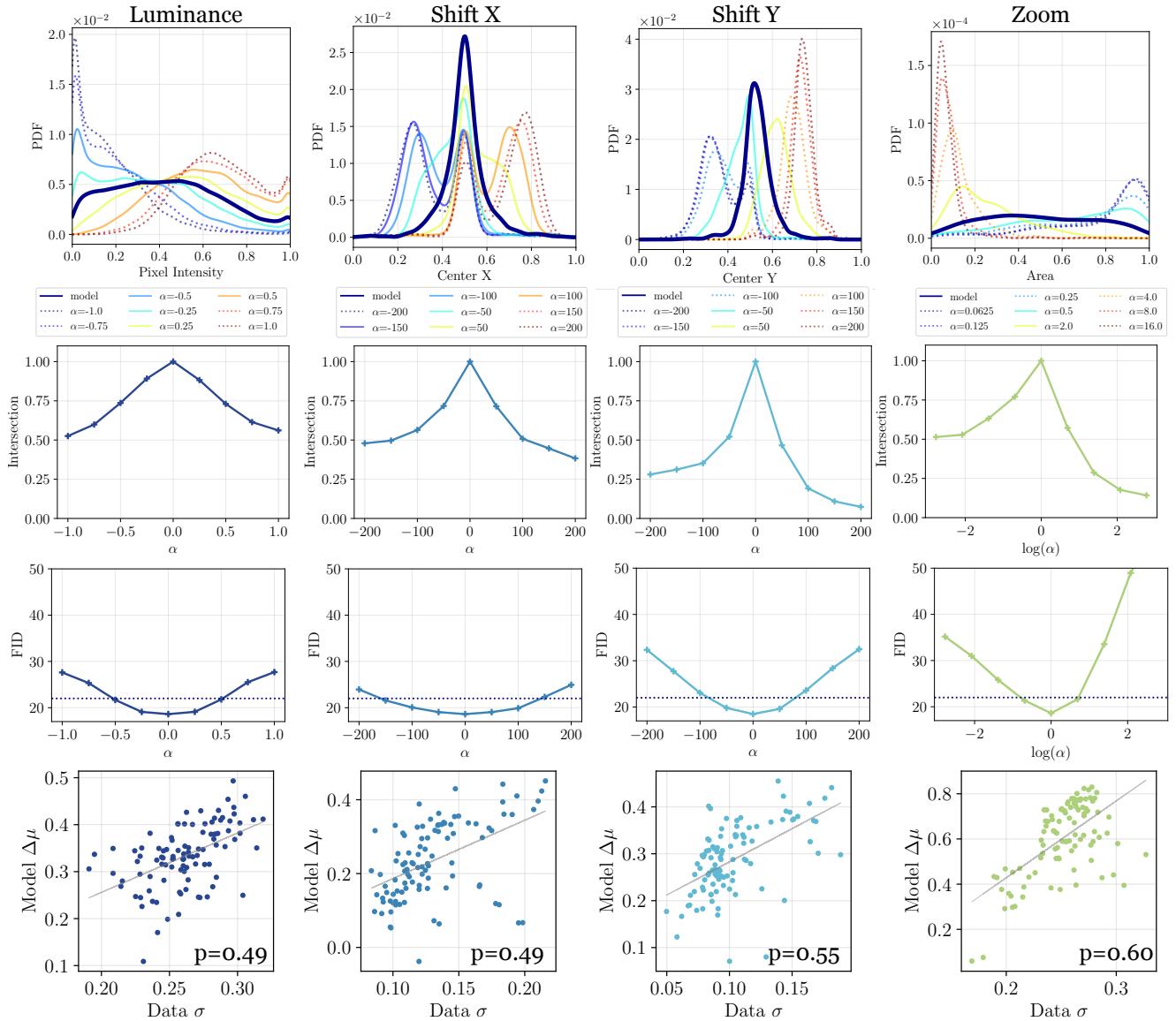
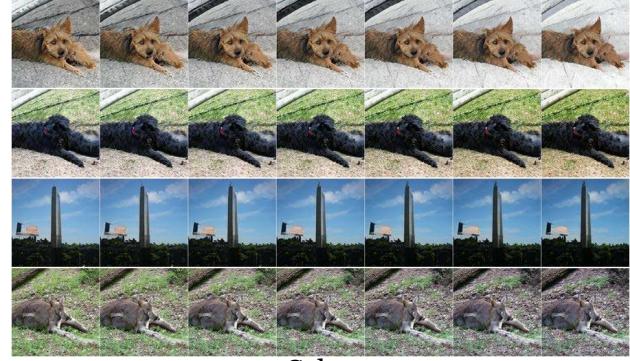


Figure 15: Quantitative experiments for nonlinear walks in BigGAN. We show the attributes of generated images under the raw model output  $G(z)$ , compared to the distribution under a learned transformation  $\text{model}(\alpha)$ , the intersection area between  $G(z)$  and  $\text{model}(\alpha)$ , FID score on transformed images, and scatterplots relating dataset variability to the extent of model transformation.



← - Zoom + →



← - Color + →



← - Shift X + →



← - Shift Y + →

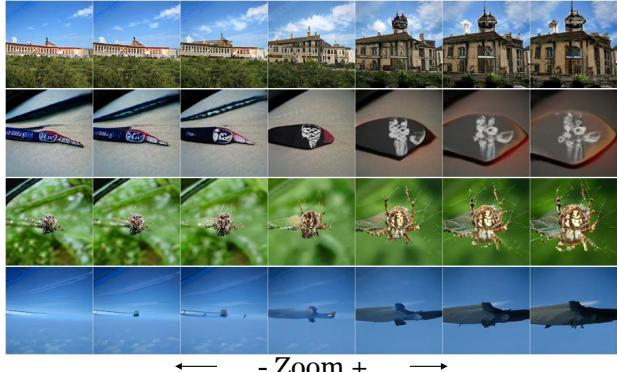


← - Rotate 2D + →

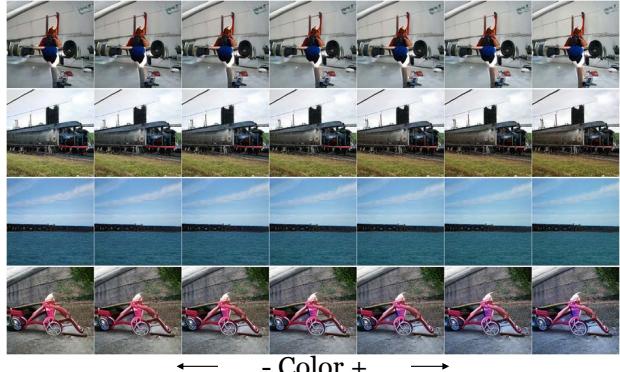


← - Rotate 3D + →

Figure 16: Qualitative examples for randomly selected categories in BigGAN, using the linear trajectory and L2 objective.



← - Zoom + →



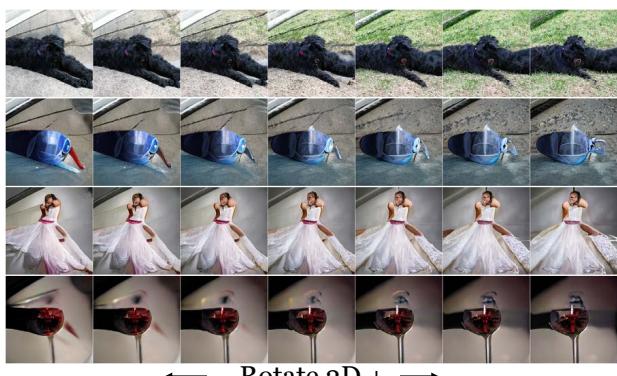
← - Color + →



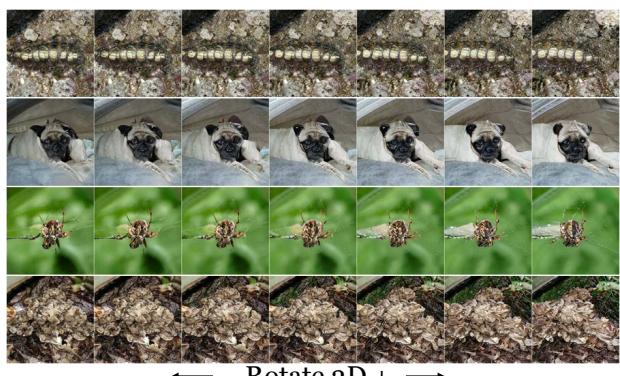
← - Shift X + →



← - Shift Y + →



← - Rotate 2D + →

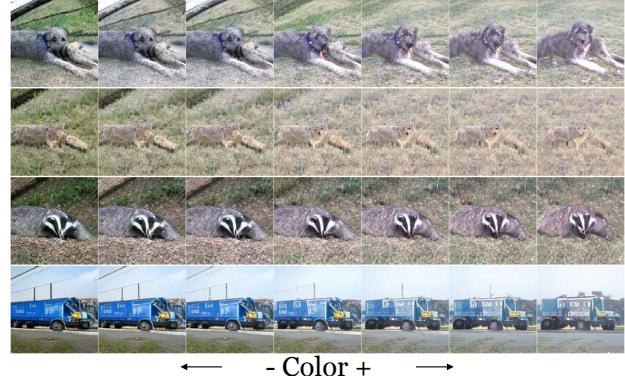


← - Rotate 3D + →

Figure 17: Qualitative examples for randomly selected categories in BigGAN, using the linear trajectory and LPIPS objective.



← - Zoom + →



← - Color + →



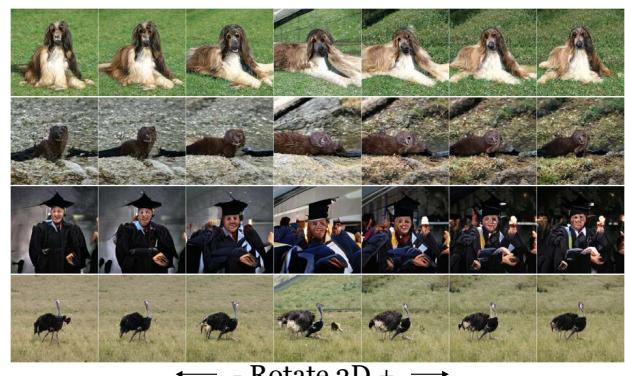
← - Shift X + →



← - Shift Y + →

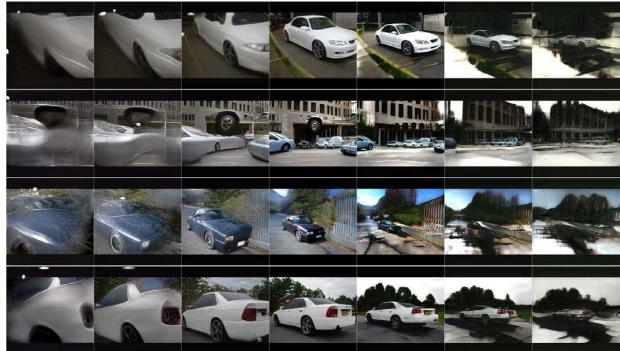


← - Rotate 2D + →



← - Rotate 3D + →

Figure 18: Qualitative examples for randomly selected categories in BigGAN, using a nonlinear trajectory.



← - Zoom + →



← - Color + →



← - Shift X + →



← - Shift Y + →



← - Rotate 2D + →



← - Rotate 3D + →

Figure 19: Qualitative examples for learned transformations using the StyleGAN car generator.

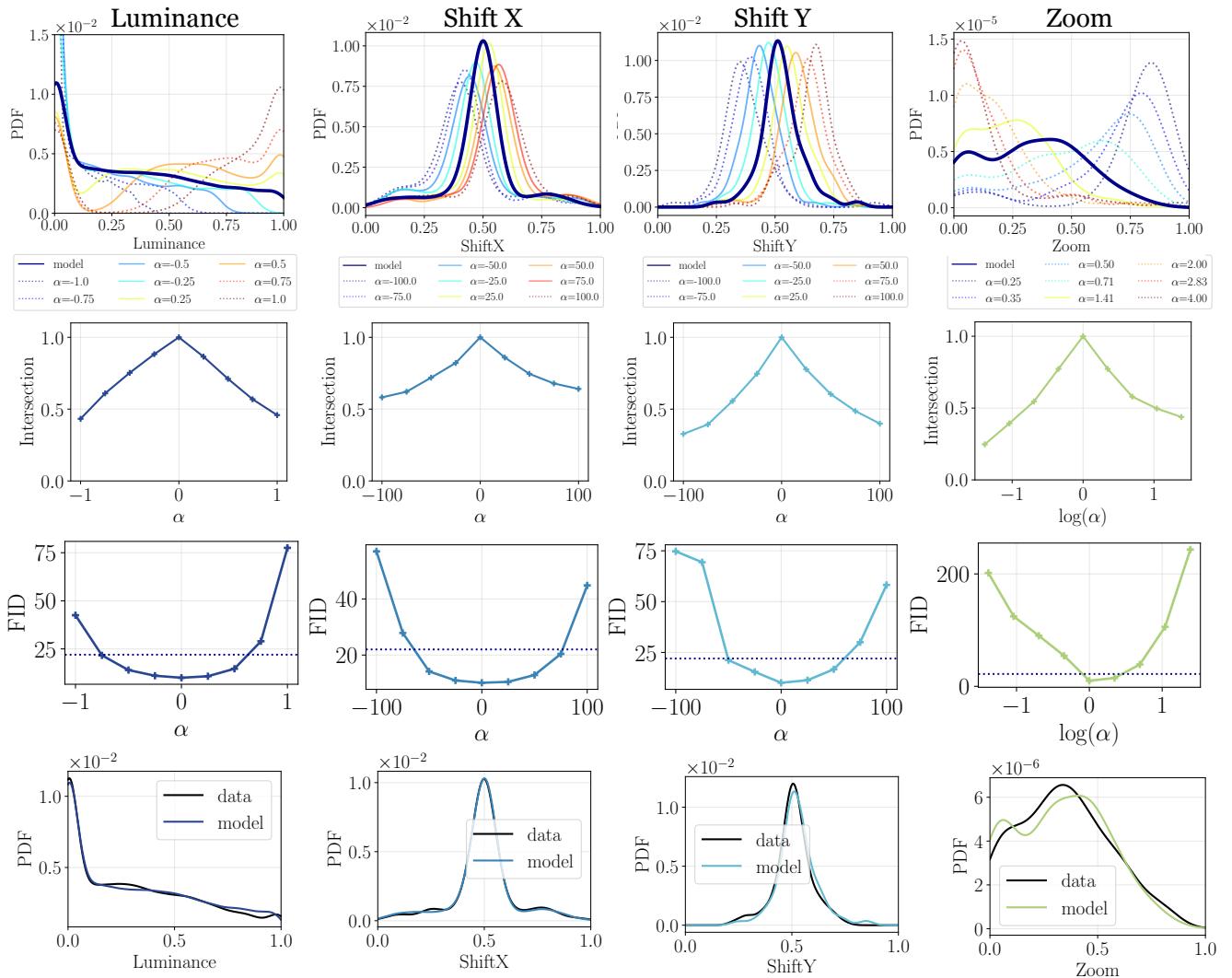


Figure 20: Quantitative experiments for learned transformations using the StyleGAN car generator.

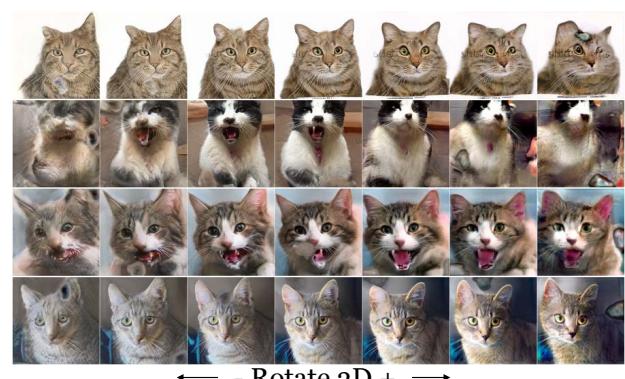
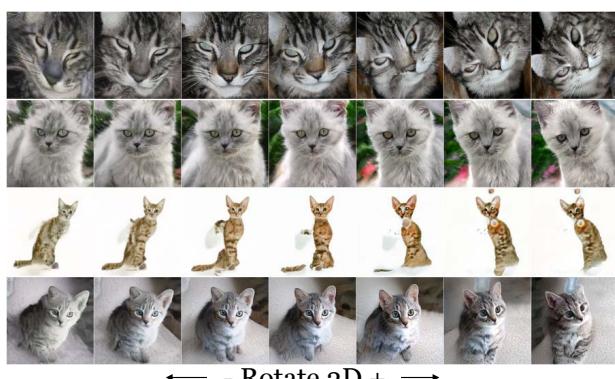
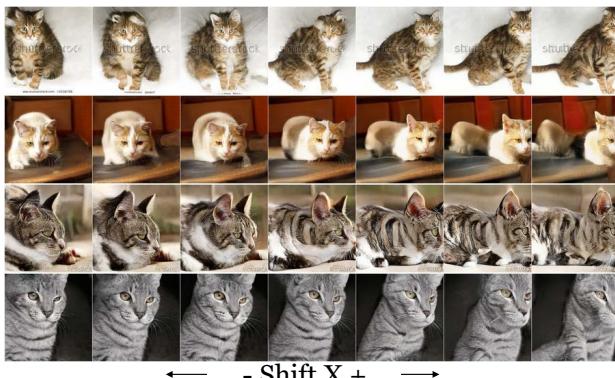
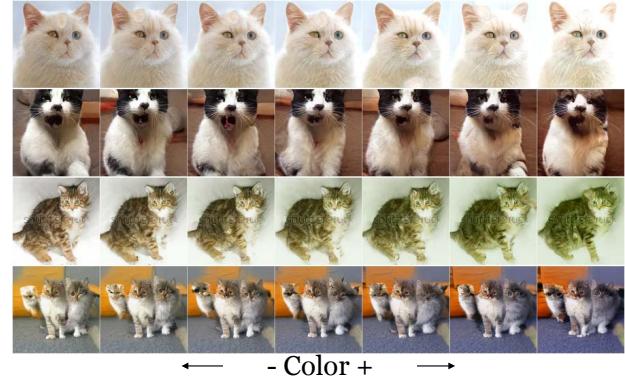
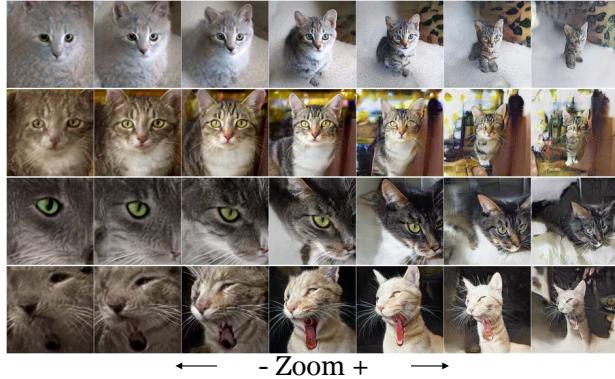


Figure 21: Qualitative examples for learned transformations using the StyleGAN cat generator.

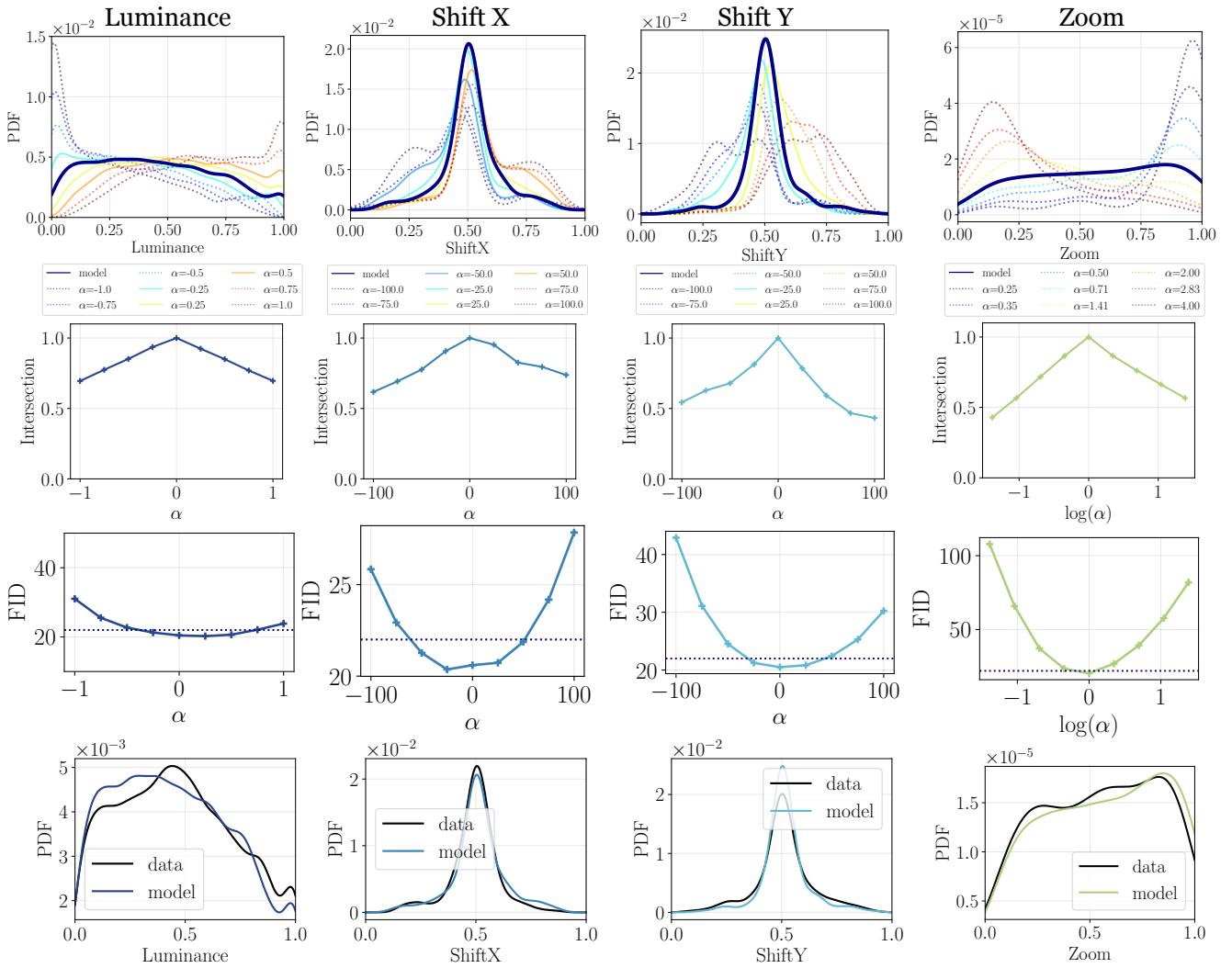


Figure 22: Quantitative experiments for learned transformations using the StyleGAN cat generator.

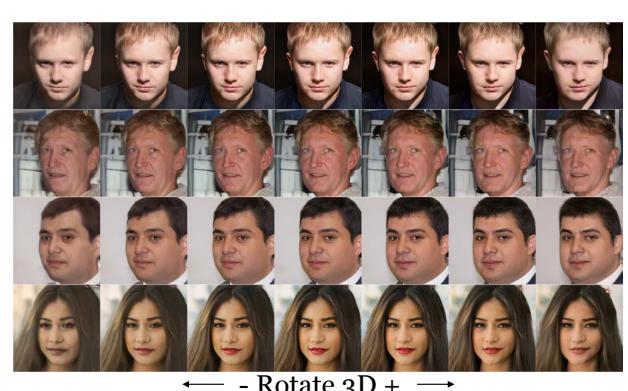
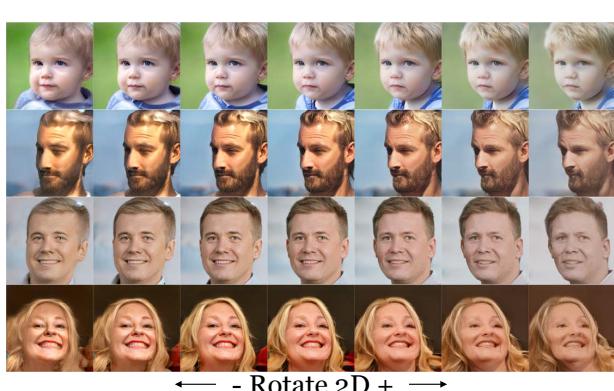
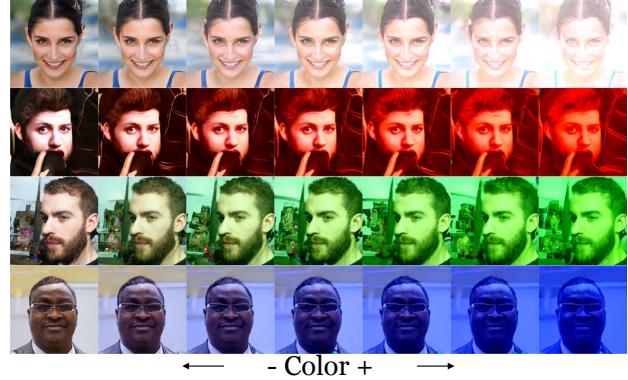


Figure 23: Qualitative examples for learned transformations using the StyleGAN FFHQ face generator.

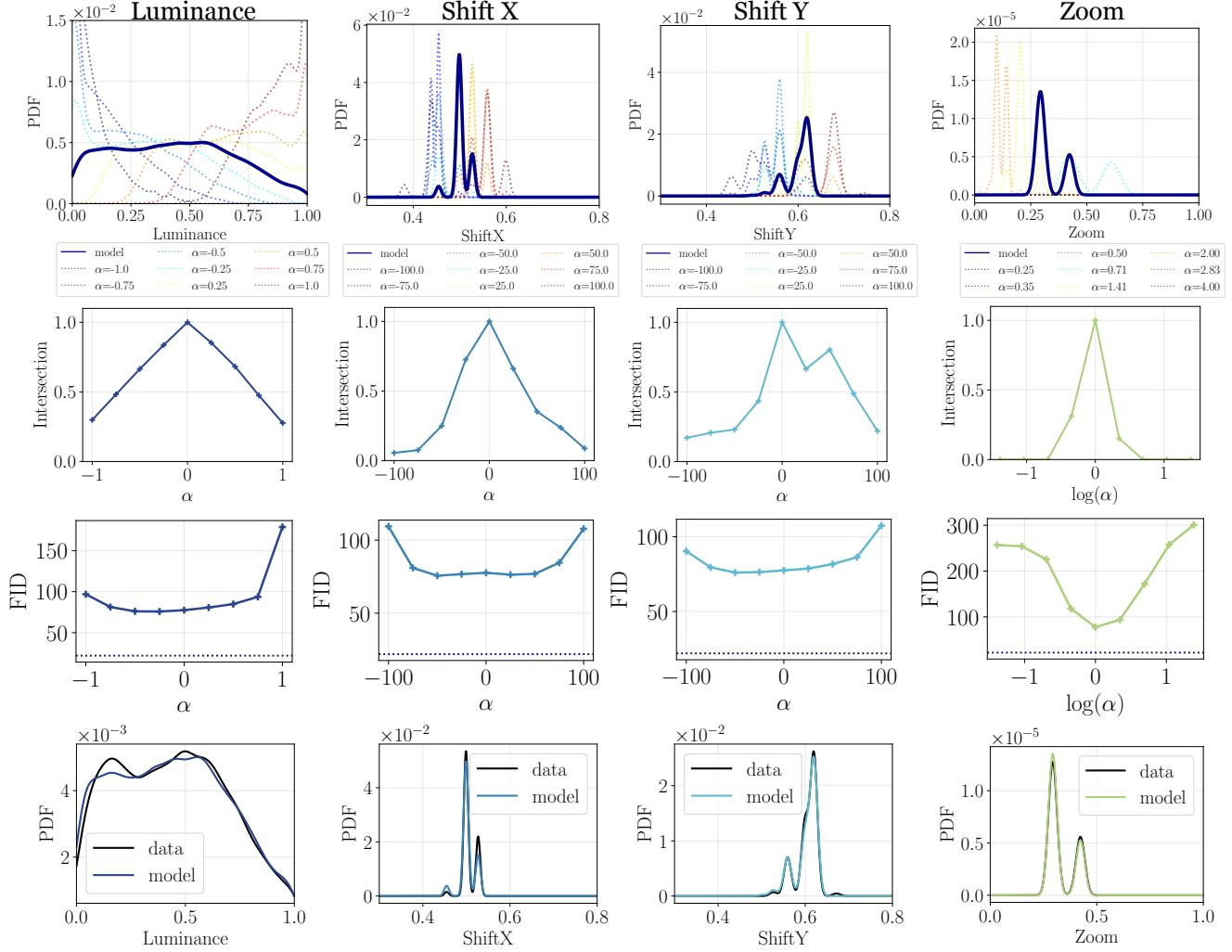


Figure 24: Quantitative experiments for learned transformations using the StyleGAN FFHQ face generator. For the zoom operation not all faces are detectable; we plot the distribution as zeros for  $\alpha$  values in which no face is detected. We use the dlib face detector [14] for bounding box coordinates.

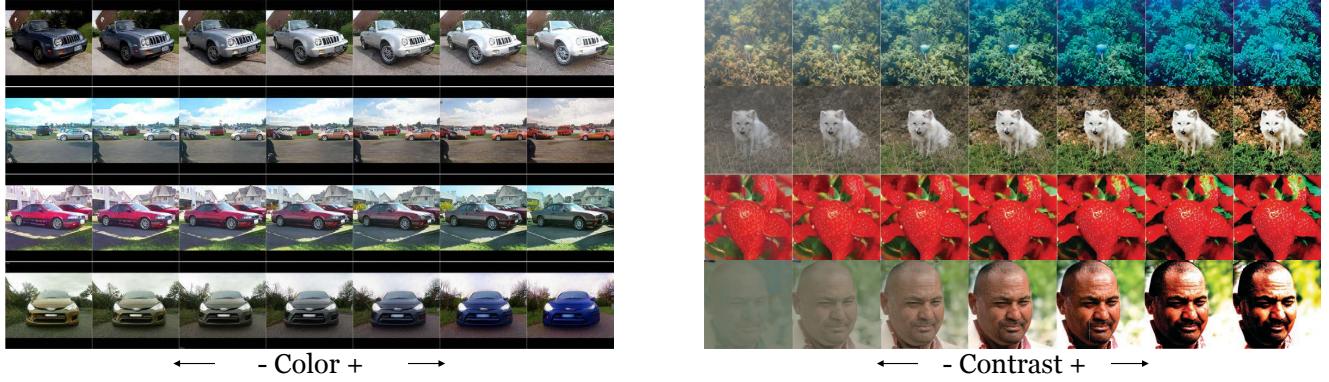


Figure 25: Qualitative examples of optimizing for a color walk with a segmented target using StyleGAN in left column and a contrast walk for both BigGAN and StyleGAN in the right column.

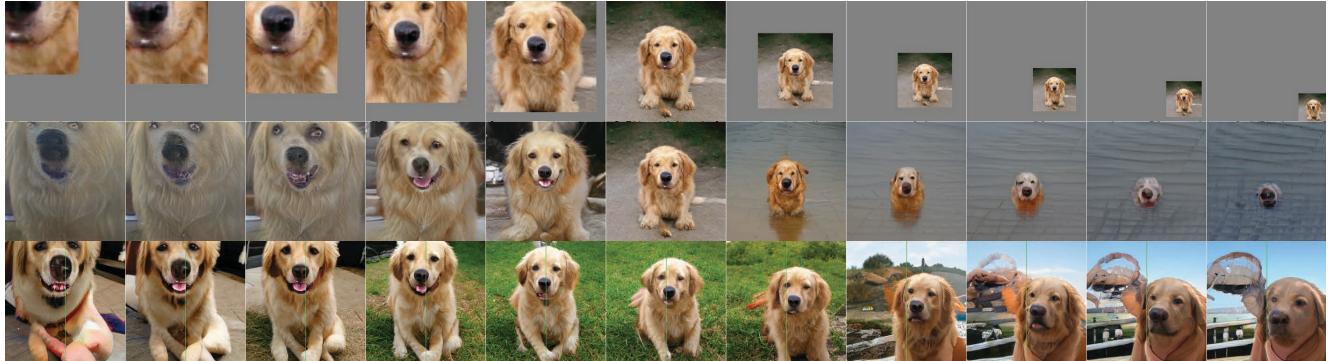


Figure 26: Qualitative examples of a linear walk combining the zoom, shift X, and shift Y transformations. First row shows the target image, second row shows the result of learning a walk for the three transformations jointly, and the third row shows results for combining the separately trained walks. Green vertical line denotes image center.