Diamond in the rough: Improving image realism by traversing the GAN latent space

Jeffrey Wen[1] **Fabian Benitez-Quiroz**[1,2] **Qianli Feng**[1,2] **Aleix Martinez**[1,2]
[1] The Ohio State University [2] Amazon
Columbus, OH 43202
{wen.254, benitez-quiroz.1, feng.559, martinez.158}@osu.edu

Abstract

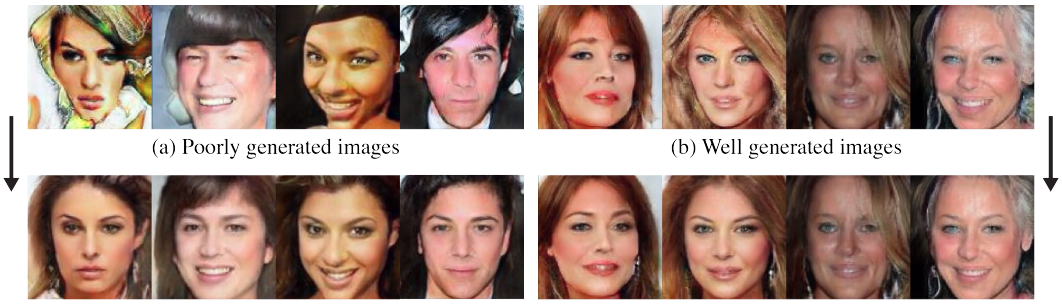
In just a few years, the photo-realism of images synthesized by Generative Adversarial Networks (GANs) has gone from somewhat reasonable to almost perfect largely by increasing the complexity of the networks, e.g., adding layers, intermediate latent spaces, style-transfer parameters, etc. This trajectory has led many of the state-of-the-art GANs to be inaccessibly large, disengaging many without large computational resources. Recognizing this, we explore a method for squeezing additional performance from existing, low-complexity GANs. Formally, we present an unsupervised method to find a direction in the latent space that aligns with improved photo-realism. Our approach leaves the network unchanged while enhancing the fidelity of the generated image. We use a simple generator inversion to find the direction in the latent space that results in the smallest change in the image space. Leveraging the learned structure of the latent space, we find moving in this direction corrects many image artifacts and presents a more realistic image. We verify our findings qualitatively and quantitatively, showing an improvement in Frechet Inception Distance (FID) exists along our trajectory which surpasses the original GAN and other approaches including a supervised method. We expand further and provide an optimization method to automatically select latent vectors along the path that balance the variation and realism of samples. We apply our method to several diverse datasets and three architectures of varying complexity to illustrate the generalizability of our approach. By expanding the utility of low-complexity and existing networks, we hope to encourage the democratization of GANs. Our code is publicly available at `https://github.com/jwen307/diamondintherough`

1 Introduction

Generative Adversarial Networks (GANs) are powerful algorithms which map vectors in a latent space into novel images [20, 19, 4, 18, 27]. While some latent vectors yield photo-realistic images, others can be easily identified as synthetic. In fact, early architectures often produce images that hardly resemble the intended object. The differentiation between latent vectors associated with photo-realistic images and those that do not has largely remained a mystery, which has made the task of filtering poor results difficult. The lack of "quality" control has lead to the abandonment of many low-complexity networks. In this paper, we propose one mechanism to constrain the latent space to vectors that produce more photo-realistic images while still maintaining diversity in class identity and attributes.

Since the original algorithm derived by [10], many variants have been proposed to improve the photo-realism of the generated images [20, 19, 4, 17, 18]. Typically, improvements in photo-realism have come at the cost of increasing the network complexity [19, 7, 29, 20]. However, the large computational requirements of these state-of-the-art networks has instilled a disconnect within the GAN community, alienating those without industrial resources and limiting the applicability of GANs

1st CtrlGen: Controllable Generative Modeling in Language and Vision Workshop at NeurIPS 2021.



(a) Poorly generated images (b) Well generated images

(c) Ours

Figure 1: Generated images of a PGAN [18] trained on CelebA [23] alongside the outputs of our algorithm. (a) Obviously-synthetic PGAN images. (b) Well-generated PGAN images. (c) The output of our method. Images in (a) are greatly improved while images in (b) maintain diversity with slight improvement to photo-realism

for large-scale use. In order to promote the integration of GANs into real-world applications, we must look for accessible, alternative methods.

Here, we take a different approach. Rather than increasing the network's complexity, we propose a method to traverse the latent space in search of vectors corresponding to more photo-realistic images. Our search contains two main constraints. Firstly, we constrain our search to the surface of a hypersphere where the majority of the probability mass lies for a n-dimensional normal distribution. Secondly, for any given latent vector $\mathbf{z}_0$ yielding image X, we find an approximation $\mathbf{z}_1$ by inverting the generator and direct our search in the direction defined by these two points in the latent space. Combining these constraints equates to searching along a great circle. We find these great circles include average sample (proto) images of high realism, Fig. 2. By moving towards the vector of the protoimage along the great circle, we observe a continuous improvement in photo-realism. We derive one possible algorithm for selecting a latent vector $\hat{\mathbf{z}}$, congruent with a higher fidelity image $\hat{\mathbf{X}}$, within the proposed search space. Samples of the resulting optimized images are shown in Fig. 1.

The proposed algorithm allows us to achieve consistent photo-realistic images even when using lower-cost networks like Progressive Growing of GANs (PGANs) [18]. Through utilizing the full capability of lower-complexity and existing networks, we hope to stimulate the accessibility of GANs.

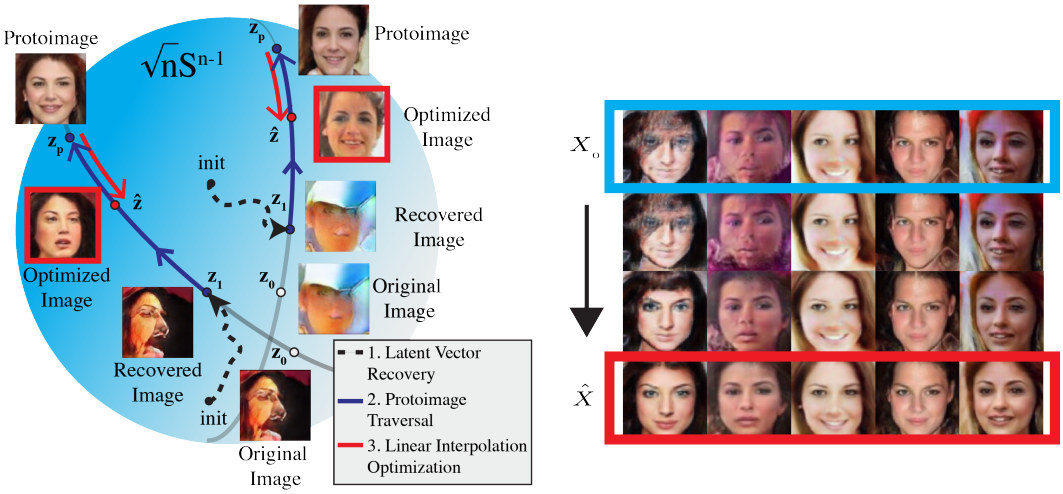
2 Related Work

2.1 Sampling from areas of high probability

Brock [4] proposed to draw latent vectors from a truncated distribution of the original prior $p(\mathbf{z})$. Given a sample drawn from $p(\mathbf{z})$, the values of vector $\mathbf{z}$ above a given threshold are resampled to be within the threshold. As the threshold becomes smaller, the samples are drawn from a smaller area of higher likelihood resulting in an increase in photo-realism.

Alternatively, Menon [25] approximates the natural image manifold by projecting sampled vectors onto the hypersphere of radius $\sqrt{n}$, n the number of dimensions of the latent space. High-dimensional Gaussian distributions take the form of a "soap bubble" with probability mass lying at or near $\sqrt{n}S^{n-1}$, where S^{n-1} is the n-dimensional unit hypersphere. This allows the authors to constrain the latent space in a way that results in increased consistency in the realism of the generated images.

While these two approaches have shown improvements for specific GAN architectures, they do not fully describe the natural image manifold. As most of the mass of the distribution is on the surface of $\sqrt{n}S^{n-1}$, the truncation trick is not guaranteed to find an area of high probability. Importantly, for architectures like PGAN where the input vector is scaled to be on the hypersphere before passing it to the generator, the approach from [25] cannot identify the regions associated to photo-realistic images. Contrary to these approaches, our framework constrains the latent space based on observations of the learned latent structure rather than simply sampling from areas of higher likelihood.



(a) Hypersphere Traversal

(b) Qualitative Progression

Figure 2: (a) A visualization of the method described in Section 3. We illustrate the traversal across the latent space for two examples, detailing the results at each step. (b) The progression as we traverse the latent space towards the protoimage.

2.2 GAN Manipulation

Several other recent works have investigated the capability of controlling the behavior of GANs through exploring the latent space [35, 30, 31, 9, 14, 26, 13, 34, 39, 15]. The goal of these methods is to find directions in the latent space that allow for interpretable semantic changes in the resulting images. [28] discovered the vector arithmetic effect allowing attributes to be added to images by adding an appropriate vector in the latent space. [35, 30] expand this idea by finding a hyperplane where latent vectors on one side exhibit a binary attribute, e.g. presence of glasses, and vectors on the other side do not. By moving latent vectors in the normal direction of the hyperplanes, they are able to semantically manipulate the resulting image for desired attributes. [9] uses a pretrained assessor network to study the direction of varying image memorability. These methods rely on pretrained classifier networks or manual annotations which may be unavailable or expensive to obtain.

In [14, 26], they utilize a self-supervision framework that transforms a generated image in the pixel space and finds the resulting direction of change in the latent space. However, both these approaches are limited to simple transformations such as rotation, zoom, and translation. [33] takes an unsupervised approach with a trainable reconstructor network and direction matrix which find directions that are easily distinguishable, and [13, 31] learn interpretable directions by performing principle component analysis (PCA) on samples in the latent space or on the learned weights that map the latent vector to the first convolutional layers.

While these methods are able to find directions that allow users to manipulate semantic features like rotation, hair color, lighting, etc., only [30] discovers a direction that allows for the control of photo-realism. The approach in [30], however, requires sampling images and manual annotations of "good" and "bad" synthesized images. As a user must subjectively label a large quantity of generated images (4k images are labeled by the authors), this method can be expensive and noisy. Contrary to this approach, our method finds the mentioned direction without supervision by exploiting the learned semantic structure of the latent space to find the direction of the protoimage vectors that represent photo-realistic sample means of well-structured data.

2.3 Generator Inversion

To begin our analysis of the generator latent space, we need a mapping from the image space to the generator's latent space. The inversion of the generator has two common approaches. The first approach is to use an optimization method to minimize the reconstruction loss [22, 5, 24]. Given an initial latent vector $\mathbf{z} \in \mathbb{R}^n$, the objective function is

$$\min_z \|X - G(\mathbf{z})\|_F, \tag{1}$$

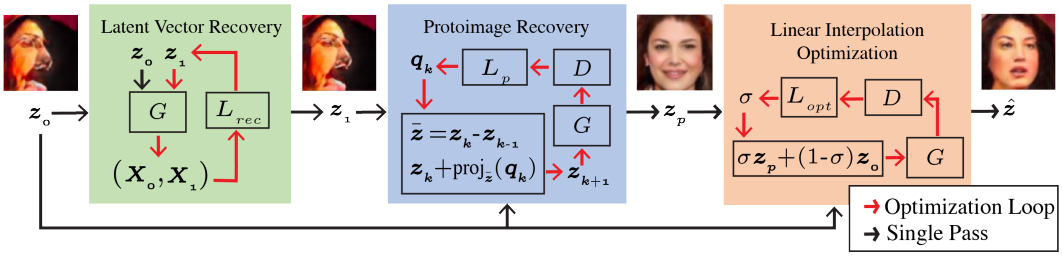


Figure 3: Overview of our framework. We find a latent vector $\mathbf{z}_1$ that visually resembles the latent vector $\mathbf{z}_0$ by using the optimization proposed in (5) (green box). We traverse the latent space along the great circle that passes through $\mathbf{z}_0$ and $\mathbf{z}_1$ to find $\mathbf{z}_p$ by iteratively projecting the direction of travel onto the hypersphere as explained in (6) (blue box). We then use the interpolation technique in (8) to get an optimized image (orange box)

where X is the image, $G(\mathbf{z})$ is the generated image given latent vector $\mathbf{z}$, and $\|\cdot\|_F$ denotes the Frobenius norm.

It is also common for a perceptual loss term to be added to this optimization [16, 8]. The perceptual loss term forces the distance between the original image and recovered image to be small in the feature space of a trained feature extraction network such as a VGG-16 pretrained on ImageNet [32]. We will use both of these criteria in our approach.

Another common approach is to train a separate encoder network [37, 2, 38]. Using randomly generated latent vectors and their corresponding generated images, an encoder network can be trained to map the images to the latent vector space. While this method is not sensitive to the initialization of $\mathbf{z}$ as in the optimization approach, it does typically result in overfitting to the training data. For these reasons, we elected to use the optimization approach together with a perceptual loss in our framework, but we show that one can also use the encoder approach efficiently in our framework.

3 Method

Let the latent vector $\mathbf{z_0}$ generate image X through generator $G(\cdot)$, $X = G(\mathbf{z_0})$. Our framework seeks to find a $\hat{\mathbf{z}}$ and corresponding $\hat{X} = G(\hat{\mathbf{z}})$ which contains the same semantic attributes as X while appearing more photo-realistic than X.

We achieve this by leveraging the discovery of a subset of similar, high-fidelity images that we call protoimages. The latent vector, $\mathbf{z}_p$, corresponding to a protoimage lies along a great circle passing through $\mathbf{z}_0$ and $\mathbf{z}_1$. $\mathbf{z}_1$ is found by inverting the generator to approximate the latent vector which produced X. By optimizing the interpolation between $\mathbf{z}_0$ and $\mathbf{z}_p$, we are able to find a $\hat{\mathbf{z}}$, which satisfies our conditions. The overall structure of the framework can be found in Fig. 3. A visualization of the movement along the hypersphere was shown in Fig. 2. We detail each of the steps in the next few sections.

3.1 Latent Vector Recovery

For a given latent vector, $\mathbf{z}_0 \sim N(\mathbf{0}, \mathbf{I}_n)$, we pass the vector through the generator to obtain the original generated image, $X \in \mathbb{R}^d$, d the number of pixels in the image. Given X, the goal is to now find a vector $\mathbf{z}_1$ which will produce a similar image $G(\mathbf{z}_1)$ to X. For this optimization problem, we utilize three loss terms. The first is a reconstruction loss, given by

$$L_{reconst} = \|X - G(\mathbf{z}_1)\|_1, \tag{2}$$

where $\|\cdot\|_1$ is the L1 norm. We use the L1 norm to prevent outlier-valued pixels from overtaking the optimization. The second term is a perceptual loss with the features extracted from the feature space of the discriminator. As [28] demonstrated the utility of the discriminator as a feature extractor for classification, we use the discriminator for our perceptual loss to keep the framework isolated from other networks. With this in mind, the perceptual loss term was formatted as

$$L_{percep} = \|D_f(X) - D_f(G(\mathbf{z}_1))\|_1, \tag{3}$$

where $D_f(\cdot)$ are the features from the discriminator. As previously mentioned, the majority of the probability mass of a high-dimensional Gaussian lies on the surface of a $\sqrt{d}S^{n-1}$ hypersphere. A proof of this can be found in the Appendix A. Thus, to keep our search in the space of high probability, we add a simple L2 regularization term to keep the latent vector search near the surface of the hypersphere. Formally,

$$L_{reg} = \big|\|\mathbf{z}_1\|_2 - \sqrt{n}\big|, \tag{4}$$

where n is the dimensionality of the latent space and $\|\cdot\|_2$ is the L2-norm. Putting all three of these terms together, we arrive at the final objective function,

$$\mathbf{z}_1* = \arg\min_{\mathbf{z}_1} L_{reconst} + \alpha L_{percep} + \beta L_{reg}, \tag{5}$$

where α and β are chosen weight parameters. We note that while the original and recovered images are nearly identical, the pairwise Euclidean distance between the original and recovered latent vectors may not be small. For example, with PGAN on CelebA, the average distance is 25.

To demonstrate the difference that such a Euclidean distance can make in the pixel-space, we added noise vectors of an equivalent magnitude to the original latent vectors and generated the images of the resulting vectors. These can be seen in Appendix F. While an equivalent distance away, these vectors provide dramatically different images from the original. Thus, this suggests that our recovered latent vectors are in a particular direction where the image attributes do not change very much. We further explore this hypothesis in the next section.

3.2 Traversing the Latent Space

In this section, we explore the effects of moving along the hypersphere in the direction of the recovered latent vector. As a simple experiment, we take the difference between the recovered, $\mathbf{z}_1$, and original latent vector, $\mathbf{z}_0$ and add it to the recovered latent vector. The resulting vector is projected onto the hypersphere and named $\mathbf{z}_2$. By repeating the sequence, $\mathbf{z}_{k+1} = proj(\mathbf{z}_k + (\mathbf{z}_k - \mathbf{z}_{k-1}))$ (where $proj(\cdot)$ is the projection onto the hypersphere), we are able to traverse the hypersphere in the direction of the recovered latent vector. The results of this visualization can be seen in Fig. 2b.

Note that as the latent vector is moved about the hypersphere, the image quality improves by bringing clarity to the already well-defined faces or by bringing previously unidentifiable faces to images that resemble a face. The key to this process is to note that this walk arrives at a similar set of protoimage around the same iteration of the sequence. These protoimages are in fact sample mean images. For the CelebA dataset, for example, the protoimage resembles a mean face of a subset of the training set.

Up until this point, each iteration improves quality but retains most of the unique attributes of the original image. After the protoimage, the subsequent iterations provide a completely new identity, which worsens in quality as the walk continues. This behavior is similar to the effects seen using the truncation trick [4]. [19] report a similar effect with the truncation trick applied to the intermediate latent space. In their experiment, they also show different identities, often antifaces, once the mean image is crossed. However, separate from those experiments, our latent vectors are not converging to the same latent vector. In fact, the latent vectors of the mean images are separated by a large angle. While the effects of moving towards the mean images appear to be the same as the truncation trick, here the generator has designed the latent space to have mean images about every great circle on the hypersphere.

3.3 Protoimage Recovery

As we observed the improvement in image quality near the protoimages, we would like to be able to find the locations of these protoimages given a random set of latent vectors.

An interesting feature of some GANs is the inclusion of a minibatch standard deviation layer in the discriminator. This layer encourages variation among the images of an input batch, which results in a lower discriminator score for images that are similar. As the mean images all have a similar look, the resulting discriminator score is very low for a batch of protoimages. There is also always a large angle between the original latent vector and the protoimage latent vectors. With this in mind, we can establish a criterion for finding the protoimages about the hypersphere walk. The objective function below seeks to minimize the discriminator score for a batch of images while also encouraging a small

cosine similarity value. Formally,

$$\mathbf{z}_p* = \arg\min_{\mathbf{z}_p} D(G(\mathbf{z}_p)) + \lambda M(\mathbf{z}_p, \mathbf{z}_0), \tag{6}$$

where $D(\cdot)$ is the discriminator score, $\mathbf{z}_p$ is the latent vector of the protoimage, $\mathbf{z}_0$ is the original latent vector, λ is a chosen weight parameter, and $M(\cdot, \cdot)$ is the cosine similarity, which can be expressed as

$$M(\mathbf{z}_p, \mathbf{z}_0) = \frac{\mathbf{z}_p \cdot \mathbf{z}_0}{\|\mathbf{z}_p\|\|\mathbf{z}_0\|}. \tag{7}$$

With this objective function, we would like to restrict the search space to vectors along the walk around the hypersphere in the direction of the difference vector.

First, we initialize a vector $\mathbf{q}_1$ to control the magnitude and forward/backward direction of the next step. We utilize a vector for controlling the hypersphere traversal to provide more robustness in the optimization. For each iteration, we find the difference vector between the current latent vector and the previous. For the first iteration, this is the difference between the recovered $\mathbf{z}_1$ and original latent vector, $\mathbf{z}_0$. The vector, $\mathbf{q}_k$, is projected onto the difference vector and added to the previous vector. The new vector, $\mathbf{z}_{k+1}$, is, then, projected onto the hypersphere. The loss is calculated using (6), and this loss is backpropagated through the network to update $\mathbf{q}_{k+1}$. The entire algorithm is summarized in Algorithm 1.

Algorithm 1: Finding the protoimages

$\mathbf{z}_0 \in \mathbb{R}^n$ (original latent vectors)
$\mathbf{z}_1 \in \mathbb{R}^n$ (recovered latent vectors)
$\mathbf{q}_k \in \mathbb{R}^n$
initialize $\mathbf{q_1}$;
$k = 1$;
$\bar{\mathbf{z}} = \mathbf{z}_1 - \mathbf{z}_0$;
$\epsilon \in \mathbb{R}, \epsilon > 0$ (stopping threshold)
while $\|\bar{\mathbf{z}}\|_2 > \epsilon$ **do**
  $\mathbf{z}_{k+1} = \mathbf{z}_k + (\mathbf{q}_k^T \bar{\mathbf{z}}) \frac{\bar{\mathbf{z}}}{\|\bar{\mathbf{z}}\|_2}$;
  $\mathbf{z}_{k+1} = \mathbf{z}_{k+1} \frac{\sqrt{n}}{\|\mathbf{z}_{k+1}\|_2}$;
  $L_p = D(G(\mathbf{z}_{k+1})) + \lambda \frac{\mathbf{z}_{k+1}^T \mathbf{z}_0}{\|\mathbf{z}_{k+1}\|_2 \|\mathbf{z}_0\|_2}$;
  Backpropagate to update $\mathbf{q}_{k+1}$;
  $k = k + 1$;
  $\bar{\mathbf{z}} = \mathbf{z}_k - \mathbf{z}_{k-1}$;
end

3.4 Optimizing for Improved Images

With the protoimages found, improving any randomly generated image is simply a matter of pushing the latent vectors towards the latent vectors corresponding to the protoimages.

This can be accomplished with an easy linear interpolation between the original latent vector $\mathbf{z}_0$ and the protoimage latent vector, $\mathbf{z}_p$ and then projecting the point onto the hypersphere.

$$\hat{\mathbf{z}} = proj(\sigma \mathbf{z}_p + (1 - \sigma)\mathbf{z}_0), \tag{8}$$

where the value of σ controls the proximity to the original latent vector and the protoimage latent vector.

While setting a constant σ can yield a decent balance between fidelity improvement and variation, we recognize that certain images need more improvement, and some minor improvements to already good images are not worth the reduction in variation.

To automate the balance between fidelity and variation, we optimize for the linear interpolation coefficient, σ. The discriminator is trained to discriminate between real photos and generated images. With the Wasserstein loss [1], the discriminator acts as a critic, providing higher scores for real

images and lower scores for images that are believed to be fake. As the image quality of generated images improves, the discriminator score improves. However, as the improved images begin to lose variation, the discriminator score drops again due to the minibatch standard deviation layer. While one option would be to find the σ's which provided the highest discriminator score for a batch of images, the optimization tends to favor the extremes, bringing some values of σ up very high towards 1 but dropping σ for other images to counteract for the loss of variation.

To derive a criterion that is beneficial for most images, we instead bring the discriminator score to the middle ground between scores given for real images and scores given for poorly generated images. We find bringing the discriminator score towards this center value allows the optimization to find values of σ that balance realism and variation. Thus, we set the objective to minimize the mean-squared L2 norm of the discriminator score vector. For a set of k original latent vectors $\mathbf{Z}_0 \in \mathbb{R}^{k \times n}$ and the protoimage vectors $\mathbf{Z}_p \in \mathbb{R}^{k \times n}$ where each row is a vector, we can define the optimization problem as

$$\sigma^* = \arg \min_{\sigma \in [0,1]^n} \frac{1}{k} \sum_{i=0}^{k} D(G(\sigma_i \mathbf{Z}_{p,i} + (1 - \sigma_i)\mathbf{Z}_{0,i}))^2 \tag{9}$$

where $\sigma \in [0, 1]^k$ is a vector of linear interpolation coefficients with elements σ_i, $\mathbf{Z}_{p,i}$ is the i^{th} row of $\mathbf{Z}_p$, and $\mathbf{Z}_{0,i}$ is the i^{th} row of $\mathbf{Z}_0$. Results of the optimization can be seen in Appendix G along with the resulting values of σ. It can be seen that images that are already high-quality have smaller values of σ while poor images receive larger values.

4 Experimental Results

We apply our framework to a multitude of experiments to evaluate the effectiveness of the image quality improvements and the limitations of our method.

For the all optimizations, we use an Adam optimizer [21] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For the latent vector recovery, we use a learning rate of 0.01. The α and β coefficients from equation 5 are set to 2 and 1, respectively. For the initialization, we randomly generate $1,000$ latent vectors and chose the latent vector with the smallest Euclidean distance to the original image in the feature space of the discriminator. In searching for the protoimages, we use a learning rate of 0.1 and a λ of 3. This algorithm must be applied to a batch of latent vectors in order for the minibatch standard deviation layer to give the protoimages small discriminator scores. Finally, for finding the linear interpolation coefficient, we use a learning rate of 0.01 and set the initial values of σ to 0.7.

4.1 Quantitative Evaluation

To evaluate the improvement of image quality and balance of image variation, we compare the Frechet Inception Distance (FID) score [12] of several variations using 10000 images. In Table 1, we compare two popular GANs with and without our proposed optimization with three datasets. For the truncation trick, we take the best score given for a sequence of threshold values from 0.1 to 1.0. Similarly, for our method, we take the best score given for a series of σ values, which gives the top score along the path towards the protoimages. We also compare our method to the top score along the direction found using the supervised approach of [30] on the CelebAHQ1024x1024 [18] dataset. Our algorithm successfully improves the FID score (by making it smaller) of the original GAN without making any changes to the network itself, and we outperform the truncation trick for all by one dataset and network. This result shows that the proposed algorithm consistently improves FID across different GANs and datasets. It should be noted that the truncation trick resamples latent vectors beyond the threshold, resulting in completely different images from the baseline; whereas our method retains many defining attributes of the original images. Surprisingly, we are able to improve upon the performance of [30] even though our method is completely unsupervised. Qualitative comparisons can be seen in Fig. 4.

4.2 Other Datasets and Architectures

To evaluate the scope of our framework, we apply our method to additional datasets and architectures. Further implementation details can be found in Appendix C. First, we investigate the structure of

Table 1: FID scores of GANs with and without our proposed algorithm. Numbers in bold show the best scores for a given pair of GAN and dataset. The proposed algorithm consistently improves FID metric across most GANs and datasets.

		FID↓			
Network	Dataset	Baseline	Truncation	[30]	Proposed
PGAN	CelebA	12.09	11.84	-	**11.69**
PGAN	Church	8.02	8.00	-	**7.81**
WGAN-GP	CelebA	58.53	58.36	-	**57.38**
PGAN	CelebAHQ	8.30	**8.11**	8.21	8.20

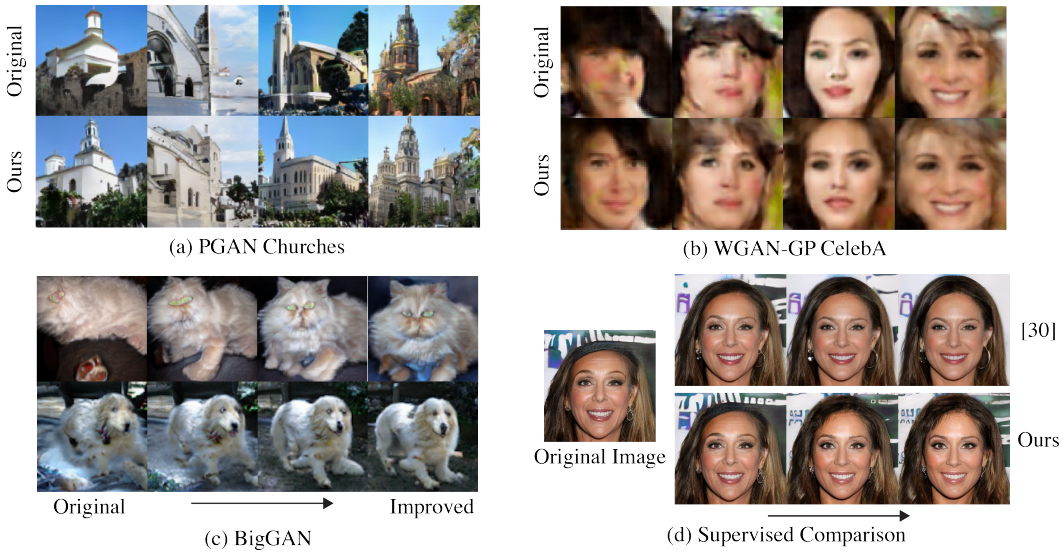


Figure 4: Results from multiple datasets and network architectures. (a), (b), and (c) show the original output and our improved results for different architectures and datasets. (d) shows a qualitative comparison with [30] on CelebAHQ as we move in our detailed directions.

PGAN trained on another dataset, the LSUN Church Dataset [36]. As the structure of the church set is much more complex than the CelebA set, we found the latent vector recovery to be much more sensitive to the initialization of the latent vector than in the CelebA set. To provide more consistent performance, we trained and utilized an encoder using the procedure of [2]. The encoder is able to provide a very good initial vector, and our optimization method brings the recovered image to nearly the same as the original.

As with the CelebA set, we see a similar structure in the latent space with the presence of protoimages. Following the same framework, we are able to find images of improved realism, which maintain most of the features of the original images. The outcome of this is shown in Fig. 4.

We further explore the scope of our proposed algorithm as it applies to other GAN architectures. We start with WGAN-GP [11]. We trained a WGAN-GP on the CelebA dataset for 10 epochs with a 512-dimensional generator latent space to match the dimensions of the PGAN latent space. Applying our latent vector recovery algorithm and traversing the latent space as described in Section 3.2, we identify the presence of protoimages. As seen with the PGAN, the images improve as they approach the protoimages, resembling more of a face when the original image is distorted. Results of our framework with the WGAN can be found in the Fig. 4.

While we have focused on enhancing the applicability of low-complexity networks, another branch of GAN accessibility is making the most of existing networks. Thus, we investigate the generalizability and utility of our method by applying it to BigGAN [4] pretrained on ImageNet [6]. For the latent vector recovery procedure in Sec. 3.1, we leave the class-conditional code fixed when searching over the latent space. Performing the walk as in Sec. 3.2 reveals a similar behavior as before with image realism improving until the walk reaches a protoimage. Examples can be seen in Fig. 4.

This illustrates the generality of the latent space direction we define. Without a minibatch standard deviation layer in the discriminator, our method can be simplified to a user-controlled manipulation as in [30, 31, 35, 9, 33] allowing users to specify the amount to travel in our discovered direction.

Limitations Our method is based on pushing samples towards the photo-realistic protoimages implicitly defined by the learned structure of the GAN latent space. Thus, our method can exhibit limited performance with extremely unstructured data, such as the LSUN Bedroom, where the sample average itself is not apart of the natural image manifold. Additionally, the protoimage tends to take the form of the majority class of a dataset. This raises ethical concerns over fair representation as our method may exacerbate dataset bias. This can be mitigated by applying our method on GANs trained on well-balanced datasets, where generated image quality is equal among the classes. A pairing with other GAN control methods can also allow manual mitigation of bias as detailed in Appendix E.

5 Conclusion

We presented a novel method for navigating the GAN latent space to increase the photo-realism of images. While previous work has achieved improvements through increasing the model complexity, our framework improves photo-realism on even shallow networks. Paired with the ability to control the attributes of generated images, we provide new insights into the structure of GAN's latent space by demonstrating the existence of protoimages. By understanding the structure of the latent space, we open opportunities to manipulate the structure during training in order to provide higher-realism and improve class variability. We showed the effectiveness of our method with different architectures and different datasets. The results presented above show great improvements on realism, especially in datasets with controlled pose structure such as faces as protoimages resemble the object itself.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of Machine Learning Research*, volume 70, pages 214–223, 2017.

[2] D. Bau, JY. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba. Seeing what a GAN cannot generate. In *International Conference Computer Vision (ICCV)*, 2019.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*, pages 36–38. Springer, 2006.

[4] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

[5] A. Creswell and A. A. Bharath. Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, 30(7):1967–1974, 2019.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] J. Donahue and K. Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[8] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[9] Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. Ganalyze: Toward visual definitions of cognitive image properties. *arXiv preprint arXiv:1906.10112*, 2019.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial Nets. In *International Conference on Neural Information Processing Systems (NIPS)*, 2014.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and AC. Courville. Improved training of wasserstein GANs. In *Advances in neural information processing systems (NIPS)*, 2017.

[12] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *International Conference on Neural Information Processing Systems (NIPS)*, 2017.

[13] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *Proc. NeurIPS*, 2020.

[14] Ali Jahanian, Lucy Chai, and Phillip Isola. On the "steerability" of generative adversarial networks. In *International Conference on Learning Representations*, 2020.

[15] Yuming Jiang, Ziqi Huang, Xingang Pan, Chen Change Loy, and Ziwei Liu. Talk-to-edit: Fine-grained facial editing via dialog. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[16] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, 2016.
[17] A. Karnewar, O. Wang, and R. Iyengar. MSG-GAN: Multi-scale gradient gan for stable image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
[18] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.
[19] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[20] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[21] DP. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.
[22] Z. C. Lipton and S. Tripathi. Precise recovery of latent vectors from generative adversarial networks. In *International Conference on Learning Representations Workshops (ICLRW)*, 2017.
[23] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, 2015.
[24] F. Ma, U. Ayaz, and S. Karaman. Invertibility of convolutional generative networks from partial measurements. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.
[25] S. Menon, A. Damian, S. Hu, N. Ravi, and C. Rudin. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[26] Antoine Plumerault, Hervé Le Borgne, and Céline Hudelot. Controlling generative models with continuous factors of variations. In *International Conference on Machine Learning (ICLR)*, 2020.
[27] G. Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *International Journal of Computer Vision*, 128:1118–1140, 2019.
[28] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.
[29] A. Razavi, A. van den Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
[30] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, 2020.
[31] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *CVPR*, 2021.
[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.
[33] A. Voynov and A. Babenko. Unsupervised discovery of interpretable directions in the gan latent space. *ArXiv*, abs/2002.03754, 2020.
[34] Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, and Qifeng Chen. High-fidelity gan inversion for image attribute editing. *arxiv:2109.06590*, 2021.
[35] Ceyuan Yang, Yujun Shen, and Bolei Zhou. Semantic hierarchy emerges in deep generative representations for scene synthesis. *International Journal of Computer Vision*, 2020.
[36] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.
[37] JY Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision (ECCV)*, 2016.
[38] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.
[39] P. Zhuang, O. Koyejo, and A. G. Schwing. Enjoy Your Editing: Controllable GANs for Image Editing via Latent Space Navigation. In *Proc. ICLR*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the end of Section 4.2
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See limitations in Section 4.2
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix A. We assume that latent vectors are drawn from a normal distribution and each random variable is identically and independently distributed.

(b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Included in the supplemental material

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Details of the hyperparameters are described in Section 4 and Appendix C

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We follow the general procedure of the GAN community using 10000 images to calculate the FID based on a wide spread of images to limit noise in the metrics.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] The authors are cited throughout the paper and in the References. The specific repositories and their links are included in the Appendix C

(b) Did you mention the license of the assets? [Yes] The licenses are mentioned in Appendix C

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] New assets are included in the supplemental material

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] This is discussed in Appendix C

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] This is addressed in Appendix C

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

In Section A, we prove that a high-dimensional Gaussian distribution is a soap-bubble. Section B discusses the traits of the protoimages and show examples of the protoimages for different datasets. C provides implementation details of our method with different architectures and datasets. Section D describes variations of our method that can improve speed and user control. Section E describes how our method can be paired with other GAN manipulation methods to get more photo-realistic images with particular attributes. Section F illustrates the uniqueness of our recovered latent vectors, and Section G shows the results of our method along with the optimized values of σ. Finally, Section H includes additional images of our results with more side-by-side comparisons of randomly generated images and our optimized output.

A Proof for High-dimensional Gaussians

As previously mentioned, the majority of the probability mass of high-dimensional Gaussians lie within a thin shell of a hypersphere. Due to this phenomena, our method of traversing the latent space becomes a simple walk along the great circles of a hypersphere. Here, we provide a formal proof of this mathematical fact. Let $\mathbf{X} = [X_1, ..., X_n]$ where $X_i \sim N(0, 1)$ are i.i.d. Denote the norm of this vector as

$$Q = \|\mathbf{X}\|_2 = \sqrt{X_1^2 + ... + X_n^2} \tag{1}$$

Q follows a chi distribution with variance

$$\text{var}(Q) = n - E[Q]^2 \tag{2}$$

where $E[\cdot]$ denotes the expected value. The variance for a chi distribution tends toward $\frac{1}{2}$ for large n, which gives

$$E(Q) \approx \sqrt{n - \frac{1}{2}} \approx \sqrt{n} \tag{3}$$

Thus, the norm of $\mathbf{X}$ follows a distribution with a mean of approximatedly $\sqrt{n}$ and a very small variance, proving that $\mathbf{X}$ must have the majority of its probability mass in a thin shell around a hypersphere of radius $\sqrt{n}$. Additional and alternate proofs can be found in [3, 30, 26].

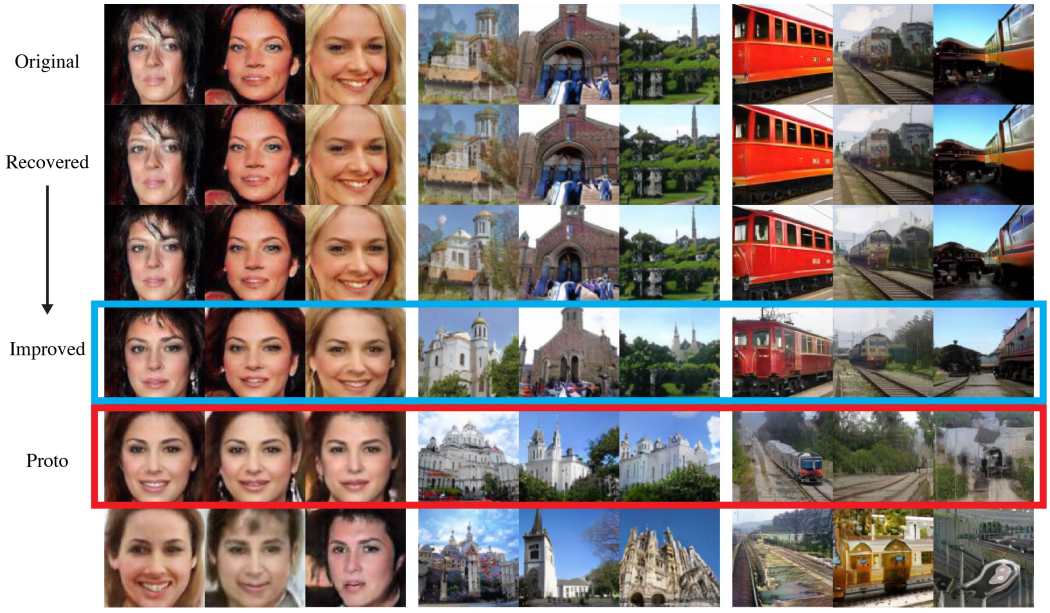
B The Protoimage

Here we present a more detailed discussion and analysis of the protoimages. These images serve as a step in our method of improving image realism, but their existence and qualities are interesting in themselves, warranting further study.

By following our procedure for traversing the latent space in Section 3.2 of the main paper, we are able to observe the protoimage along the path around the hypersphere. This effect can be seen in Fig. 1. The protoimages are very similar in appearance and seem to represent a sample mean. Despite the similarity between these images, the corresponding latent vectors are distinct and a large distance from one another. As we approach the protoimages, we can observe an improvement in image fidelity. The protoimages are observed in the same iteration of the traversal, and the proceeding iteration yields a new identity or scene, as shown in the last row of Fig. 1. Additionally, we observe that the protoimages take the attributes of the majority class within the training set. For the CelebA dataset, the protoimages are all female as the dataset contains more images of female celebrities than male. Likewise, the protoimages of the LSUN Church dataset are images of white churches as the majority of churches in the dataset are white. Interestingly, we observe changes in the protoimages for PGAN trained on different resolutions of CelebAHQ. For CelebAHQ 512x512, the protoimages exhibit the same behavior as previously described. For CelebAHQ 1024x1024, the protoimages start to include a unique artifact, Fig. 2. Despite the presence of the artifact, we do find that moving towards the protoimage still improves the realism of the face but warrants earlier stopping along the path.

C Implementation Details

We outline the different hyperparameters used for different networks and datasets. For the PGAN with CelebA, we used $\alpha = 2$, $\beta = 1$, $\lambda = 3$. For the PGAN with LSUN Church and Train, we used the



(a) CelebA (b) Church (c) Trains

Figure 1: Here we show the progression as we traverse the latent space. The image quality improves as we approach the protoimages marked in red. Note the similarity of the protoimages. We see the step after the protoimages shows a completely different identity or scene.

same values for α and β but set $\lambda = 5$. For the PGAN with CelebAHQ, we used $\alpha = 10$, $\beta = 1$, and $\lambda = 100$. While using BigGAN pretrained on ImageNet, the hyperparameters were $\alpha = 15$, $\beta = 1$, and $\lambda = 100$. When searching for the protoimage with WGAN-GP trained on CelebA, we found we needed to add two addition terms to the loss function to prevent overshooting the protoimage. For this case, we add a term for the standard deviation of the discriminator scores and standard deviation of the cosine similarity between the original latent vectors and the current latent vectors in the batch. This allows the latent vectors to move along the hypersphere at a similar pace, which makes the detection of the protoimages more apparent. For a set of k original latent vectors $\mathbf{Z}_0 \in \mathbb{R}^{k \times n}$ and the protoimage vectors $\mathbf{Z}_p^* \in \mathbb{R}^{k \times n}$ where each row is a vector, the optimization can be refined as

$$\begin{aligned} \mathbf{Z}_p^* = \arg\min_{\mathbf{Z}_p} \frac{1}{k} \sum_{i=0}^{k} D(G(\mathbf{Z}_{p,i})) + \lambda \frac{1}{k} \sum_{i=0}^{k} M(\mathbf{Z}_{p,i}, \mathbf{Z}_{0,i}) \\ + \gamma \operatorname{std}(D(G(\mathbf{Z}_p)) + \delta \operatorname{std}(M(\mathbf{Z}_p, \mathbf{Z}_0)) \end{aligned} \tag{4}$$

where γ and δ are hyperparameters, $M(\cdot, \cdot)$ is the pairwise cosine similarity as before, and $\operatorname{std}(\cdot)$ is the standard deviation. We set $\lambda = 0.2$, $\gamma = 1$, and $\delta = 3$.

We used the PGAN and WGAN-GP implementations from `https://github.com/facebookresearch/pytorch_GAN_zoo.git` with a BSD-3-Clause license. The model weights for CelebA, CelebAHQ256, and CelebAHQ512 were taken from the same repository. The CelebAHQ1024 dataset and the weights for CelebAHQ1024 and all LSUN datasets were ported from `https://github.com/tkarras/progressive_growing_of_gans.git` with a Creative Commons Public License. The weights were converted from Tensorflow to PyTorch. For BigGAN, we used the author's PyTorch implementation at `https://github.com/ajbrock/BigGAN-PyTorch.git` with an MIT License. We use the encoder architecture for inverting the generator from `https://github.com/davidbau/ganseeing`. The repositories for the encoder, CelebA, and LSUN datasets did not include a license. All datasets used are publicly available and consent is not required; however, you must agree to not redistribute the data or use it for commercial purposes. The data does not contain personally identifiable information or offensive content. Identity labels are given as numbers with no relation to personal names. All networks and our method were capable of running on a single NVIDIA RTX 2070 graphics card with 8GB of RAM.

(a) CelebAHQ-512 (b) CelebAHQ-1024

Figure 2: We observe a similar traversal progression with larger resolution datasets such as CelebAHQ512 and CelebAHQ1024. The 1024x1024 dataset presents unique artifacts to the protoimages; however, the preceeding images still improve in realism.

D Method Variations

In Section 3.4 of the main paper, our method finds the latent vector of the protoimage for each starting sample. While this is effective, we alternatively find that the procedure can be simplified for applications to a large number of samples. Instead, we can find the protoimage latent vectors for a subset of 1000 images and calculate the mean latent vector. Then, we can apply the procedures in Section 3.2 or 3.4 to move the samples towards the mean latent vector. This results in a similar improvement in photo-realism while reducing the required computational load.

The optimization in Section 3.4 can be further controlled by the user by defining bounds on the value of σ. If the user wants to guarantee some level of change in the image without allowing the image to get too close to the protoimage, they can place an lower and upper bound on the value of σ. During the optimization, σ would simply be projected back into the set if it were to exceed the bounds. We find this variation to be a balance between full user control as in [30, 31, 35, 9, 33] and our automated method.

E Attribute Control

We explore if we can apply our framework to retrieve higher-quality images with a particular attribute using one of the methods described in Sec. 2.2. Following the same procedure as [28], we find a set of generated images with a particular attribute and a set without the attribute. By taking the difference vector between the mean vectors of both sets, we can obtain a vector which adds or subtracts the attribute from a given image. However, the method in [28] does not necessarily result in realistic images. Thankfully, by applying our method on the resulting images, we can drastically increase the image realism while preserving the target attribute.

To demonstrate this, we found a set of 100 generated images of males and 100 generated images of females. By adding the difference vector between the set averages, we can make any randomly

(a) Original Images

(b) After Adding/Subtracting "Male" Vector

(c) Optimized Male Images

Figure 3: Controlling image attributes. We show the compatibility of our proposed method with an attribute control method [28]. By first adding or subtracting the "male" vector to a randomly generated image and then applying our method, we are able to obtain high-fidelity images of a particular gender.

generated image into a male. By subtracting the vector, the resulting image is of a female. Then, we apply our method to improve the photo-realism. Figs. 3 shows qualitative results of our experiment, compared to Figs. 3(b) and 3(c). We can see a significantly improvement in image realism with target female/male attribute preserved when using our proposed algorithm.

F Uniqueness of Recovered Latent Vectors

As discussed in Section 3.1, the recovered latent vector can be a large distance away from the original latent vector despite the resulting images being nearly identical. We note that latent vectors of an equivalent distance but different direction from our recovered vector yield very different images. This can be visualized in Fig. 4. Our recovered latent vectors lie in a direction where image attributes do not change very much, and the uniqueness of this direction forms the basis of our proposed method.

G Optimized Linear Interpolation Results

Following the method described in Section 3.4, we show the resulting images after moving the latent vector by the optimized value of σ in Fig. 5. We demonstrate that some latent vectors may need to move further towards the protoimage to achieve improved photo-realism while others may need to move very little. Well-generated images receive a small value of σ, which allows the method to preserve high variation among samples.

H Additional Figures

In Fig. 6, 7, 8 we show a set of results on the CelebA, LSUN Church, and LSUN Train dataset. We randomly generate 30 images and show the resulting output from our framework. The results show a balance of improving the realism while not allowing all the images to converge to the protoimages. As a result, not all of the images experience a large transformation. The framework yields better results

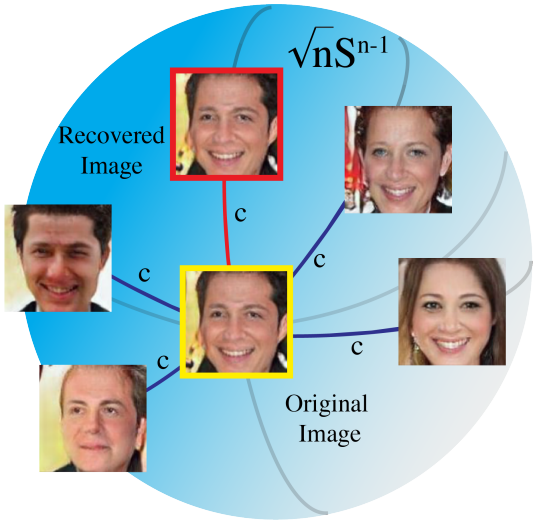
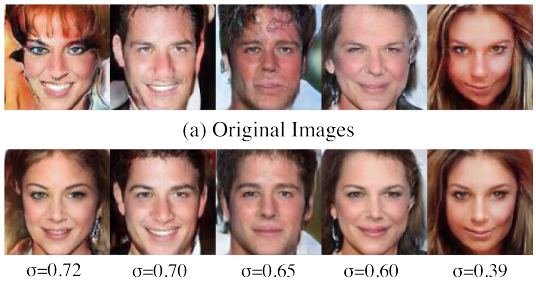


Figure 4: We demonstrate the uniqueness of the latent vectors recovered by our optimization. The recovered image in red is nearly identical to the original image in yellow. Images of an equivalent distance, c, in other directions yield very different images.



(a) Original Images

(b) Optimized Linear Interpolation

Figure 5: Results of applying our optimization framework to randomly generated images.

for datasets with objects of consistent structure such as faces; however, it is still able to improve images of datasets with more variety as seen with the churches and trains.

Figure 6: A set of 30 randomly generated images from a PGAN pretrained on CelebA with the corresponding output of our method.

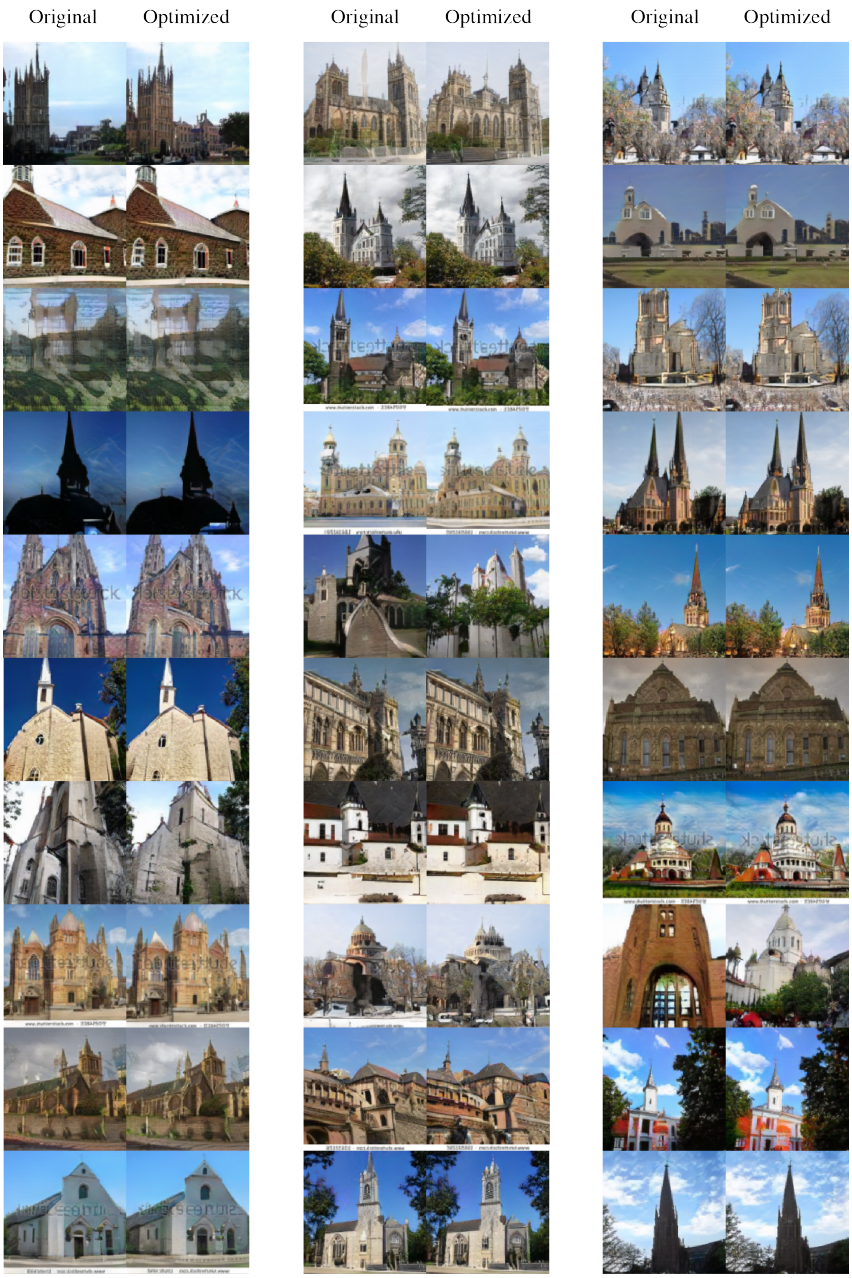


Figure 7: A set of 30 randomly generated images from a PGAN pretrained on LSUN Church with the corresponding output of our method.

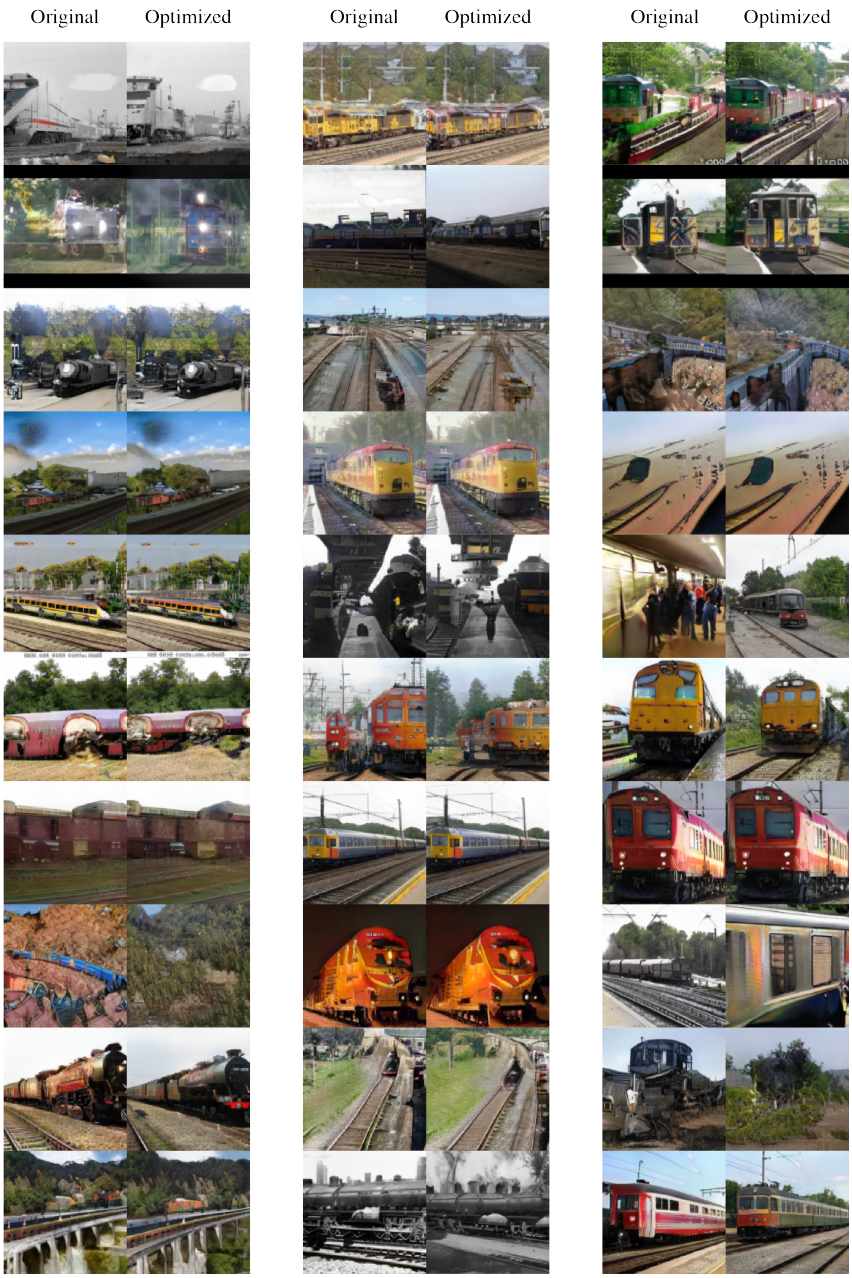


Figure 8: A set of 30 randomly generated images from a PGAN pretrained on LSUN Train with the corresponding output of our method.