Improving Inversion and Generation Diversity in StyleGAN using a Gaussianized Latent Space

Jonas Wulff
MIT CSAIL
jwulff@csail.mit.edu

Antonio Torralba
MIT CSAIL
torralba@mit.edu

Abstract

Modern Generative Adversarial Networks are capable of creating artificial, photorealistic images from latent vectors living in a low-dimensional learned latent space. It has been shown that a wide range of images can be projected into this space, including images outside of the domain that the generator was trained on. However, while in this case the generator reproduces the pixels and textures of the images, the reconstructed latent vectors are unstable and small perturbations result in significant image distortions. In this work, we propose to explicitly model the data distribution in latent space. We show that, under a simple nonlinear operation, the data distribution can be modeled as Gaussian and therefore expressed using sufficient statistics. This yields a simple Gaussian prior, which we use to regularize the projection of images into the latent space. The resulting projections lie in smoother and better behaved regions of the latent space, as shown using interpolation performance for both real and generated images. Furthermore, the Gaussian model of the distribution in latent space allows us to investigate the origins of artifacts in the generator output, and provides a method for reducing these artifacts while maintaining diversity of the generated images.

1 Introduction

Generative Adversarial Networks [8] are among the most impressive and surprising use cases of the Deep Learning revolution in computer vision in recent years. Given only a handful of random numbers (the so-called *latent vector*), these networks generate images that, to the untrained eye, are virtually indistinguishable from real photos. This applies to indoor and outdoor settings (e.g. photos of churches or bedrooms [12]), animals such as dogs and horses [5], and even to content to which human observers are highly attuned and sensitive, such as faces. For faces in particular, StyleGAN [12] and its successor, StyleGANv2 [13] have had great success in the generation of high resolution (*i.e.* 1024×1024 pixel), high quality images, leading to entertaining past times such as games where the goal is to "spot the fake"[1], but also to concerns about the authenticity and trustworthiness of photographic evidence. In short, StyleGAN far surpasses the so-called uncanny valley and creates images that we often unquestioning accept as real.

This in turn raises the question of inversion: Given the high quality output of StyleGAN, is it possible to project an existing, real image into the latent space? That is, given an input image I, can we find a latent vector $\mathbf{z}$ so that $G(\mathbf{z}) \approx I$, where G is the generator mapping the latent vector to the image? Since directions in the latent space often correspond to semantic concepts [12, 16, 18, 17], a successful inversion would allow the user to perform intuitive, semantic image editing. For example, to decrease the apparent age of a person, instead of removing every wrinkle manually, the user could simply decrease the "age" direction in latent space and the image would change accordingly, without the artist having to manually touch the pixels.

[1]For example, www.whichfaceisreal.com or www.thispersondoesnotexist.com

Due to its properties, StyleGAN is particularly well-suited for this type of application, since it first maps a random input vector $\mathbf{z} \in \mathcal{Z}$ to an intermediate latent vector (or "style") $\mathbf{w}$ in an intermediate latent space $\mathcal{W}$ using a learned mapping network M. While the distribution of $\mathbf{z}$ is fixed a priori, the distribution of $\mathbf{w}$ in the intermediate latent space is not manually defined but is learned during training. This allows it to capture semantic directions that are specific to the data (for example, "smile" vs "frown" for human faces, which does not make sense for bedrooms), and makes the inversion process to $\mathcal{W}$ easier than to $\mathcal{Z}$. However, inverting to the latent space $\mathcal{W}$ is usually not sufficient to regenerate the input image [2, 13]. Instead, one can exploit that fact that StyleGAN uses the latent vector $\mathbf{w}$ on different scales in the image generation process, and inject *different* styles at different scales [2, 18]. The space of such "composite styles", each of which consists of a separate latent vector for each scale, is usually denoted as the extended latent space $\mathcal{W}^+$, and has been shown to be remarkably effective as a target space when inverting real images [2]. Interestingly, $\mathcal{W}^+$ is powerful enough to draw virtually *any* image, irrespective whether it is likely under the training distribution or not. However, those images (for example, a car drawn by a GAN trained on faces) fall into poorly behaved and unstable regions of the latent space; for example, when interpolating between such a car and a face in latent space, the generated intermediate images usually amount to nothing but noise. When inverting to the powerful latent space $\mathcal{W}^+$, inversion can therefore be seen as an ill-posed problem, and reliably solving it requires a prior on the data distribution in latent space.

In this work, we propose such a prior. We show that, when removing the effect of the last nonlinearity in the mapping network M, the distribution of the resulting data can be modeled as Gaussian. This allows us to describe the data distribution using sufficient statistics (its covariance matrix $\mathbf{\Sigma}$ and mean $\boldsymbol{\mu}$), which in turn provide an easy-to-use prior to integrate into the inversion procedure.

Furthermore, and going beyond the problem of inversion, such a Gaussian model allows us to probe the behavior of the GAN using techniques that assume an underlying Gaussian distribution. We demonstrate this by performing a Principal Component Analysis (PCA) on the data in the Gaussian latent space, which helps us identify when the Generator generates images that contain artifacts. Isolating the causes of such artifacts allows us to selectively remove them without resorting to the truncation trick [12], which, while effective, considerably decreases diversity in the output images, even for images that do not contain artifacts.

To summarize, in this work we show that (a) after a simple non-linear correction, the data distribution in the intermediate latent space of StyleGAN can be modeled as Gaussian; (b) that imposing this Gaussian as a regularization on the task of inversion leads to embeddings that are located in better parts of the latent space, as measured by interpolation between embeddings; and (c) that the Gaussian model allows us to analyze and remove artifacts of the image generator while maintaining diversity.

1.1 Related work

Embedding natural images into the latent space of GANs has been an active field of research. In iGAN [20], continuous optimization is used to obtain an embedding for an input image, serving as a starting point for interactive editing. BiGAN [7] trains an inverter simultaneously with the Generator, thereby encouraging the latent space to be well suited for inversion. In both works, DCGAN [15] is used as a Generator, and the quality is therefore limited. Variational Autoencoders [14, 9] suffer from a similar problem. Their focus lies on learning a latent representation of real images; while they often learn latent spaces that are semantically well behaved, their output quality is typically lower than contemporary GANs. Furthermore, VAEs commonly impose a distribution on the latent *a priori*, which can prevent them to capture modes of variation inherent in the training data, thereby reducing expressiveness. The key insight of our work is that, under an appropriate transformation, an analytical prior can be fitted to the *learned* latent distribution, which gives all advantages of a prior while at the same time capturing the idiosyncrasies of the data. Furthermore, autoencoders fix the encoder architecture during training; in contrast, our latent prior can be used with any projection method and optimization technique that can accommodate a prior.

Bau *et al.* [3] propose an inversion method for PGAN [11], which has been shown to encode semantic concepts in the unit activations of intermediate layers [4]. They first train a separate encoder network $E(I)$, which maps an image to an approximate location in latent space. From there, they use continuous optimization to find the best latent representation of an image; crucially, during optimization they also adjust the weights of the generator to best replicate the input image. This

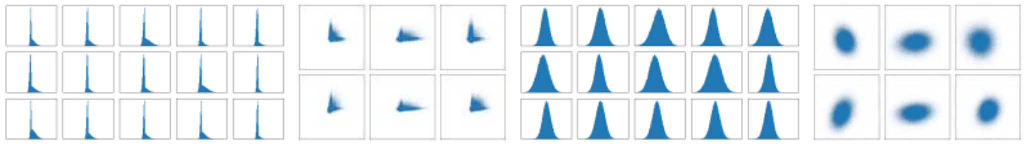

(a) Marginals in $\mathcal{W}$ (b) Pairwise distributions in $\mathcal{W}$ (c) Marginals in $\mathcal{V}$ (d) Pairwise distributions in $\mathcal{V}$

Figure 1: Statistics on $\mathcal{W}$ and $\mathcal{V}$. Marginal (a) and pairwise (b) distributions of w are highly irregular. After mapping into $\mathcal{V}$, the marginal (c) and pairwise (d) distributions show that the data can be well modeled as a high dimensional Gaussian. All plots are centered at 0 and show the same range.

procedure yields an editable latent representation of the input image and a generator that is fine-tuned to this image, which then allows modifications of real images such as the removal of windows.

Moving to higher-quality generators, the original StyleGANv2 [13] proposes an inversion method to the intermediate latent space $\mathcal{W}$. Their method, using a curriculum of noise and learning rates, can invert natural images while retaining semantics (for example, the rotation of a car); however, the exact appearance often differs due to a lack of flexibility in $\mathcal{W}$.

Considering the inversion of an input image with the goal of editability, Image2StyleGAN [2] proposes to project an image into an extended latent space $\mathcal{W}^+$, effectively optimizing a separate style for each scale. This leads to reconstructions with high fidelity and can be used for interesting editing applications, in particular when combined with additional optimization of the noise maps [1]. None of these works impose a data-dependent prior during the inversion process.

2 Gaussianizing the latent space of StyleGAN

To recapitulate, the processing pipeline of StyleGAN [12, 13] can be roughly divided into two stages.[2] First, an input latent vector $\mathbf{z} \in \mathbb{R}^d$ is sampled from a uniform distribution on the d-dimensional hypersphere. This $\mathbf{z}$ is then passed through a trained mapping network M which computes a style $\mathbf{w} \in \mathcal{W}$. Since no explicit constraints are imposed on the structure of $\mathcal{W}$, it learns to capture and disentangle the inherent structure of the training data [12]. Due to this disentanglement, latent vectors $\mathbf{w}$ are semantically more meaningful than $\mathbf{z}$ [18], and $\mathcal{W}$ is commonly used as "the" latent space of StyleGAN [12, 13, 2]. To generate the final output image, $\mathbf{w}$ is then passed to the generation network and used to modulate noise maps with increasing resolution via affine transformations of the features, resulting in the final output image. Here, we keep the generation network fixed, and focus our attention on the structure of the latent space $\mathcal{W}$ and the distribution of $\mathbf{w}$.

When inverting a real image I, it is typically found that inversion to $\mathcal{W}$ is easier than to $\mathcal{Z}$, and common inversion methods compute $\mathbf{w} \in \mathcal{W}$ so that some distance $dist(G(\mathbf{w}), I)$ is minimized. Due to the myriad possible variations of output images, however, not every image can be perfectly (or even well) recreated from a single $\mathbf{w}$. Instead, the resulting regenerated image $G(\mathbf{w})$ commonly matches the semantic content of I, but not the exact appearance [13]. An alternative is to use an extended latent space $\mathcal{W}^+$ [1, 2, 16, 18]. A point in this space is effectively a set of multiple styles, each of which is used as an input to a different scale of G. Styles on different scales are thus decoupled from each other, greatly increasing the flexibility and coverage of the output.

However, as pointed out by Abdal *et al.* [2], this increased flexibility has a curious side-effect – the GAN can now reproduce virtually *any* image, even those far outside of the domain of the training data. For example, a GAN trained on faces has no difficulties generating a car. However, in case of such out-of-domain inversion, the generator merely draws the pixels, but does not "understand" the image content, and the latent space in the immediate surrounding of the projection result is poorly behaved [2]. For example, interpolating between such an out-of-domain image and an in-domain image rarely yields good results, instead, the intermediate images contain mostly noise. In this work,

[2]While StyleGAN [12] and its successor StyleGANv2 [13] are separate pieces of work, their main difference lies in how they generate the output image from a given style $\mathbf{w}$. The style vector $\mathbf{w}$ and how it is generated from $\mathbf{z}$ does not change between both versions of StyleGAN; hence, we denote both as "StyleGAN-based architectures".

we observe that this effect is not just relevant for out-of-domain images; instead, when inverting to $\mathcal{W}^+$, even images that semantically belong to the training domain (*e.g.* faces) are often inverted to poor regions of the latent space, resulting in similar (albeit more subtle) interpolation issues.

To alleviate this issue, we propose to impose a prior on the latent vector $\mathbf{w}$, thereby encouraging an inversion to stay close to the data distribution[3]. Unfortunately, plotting sample marginal distributions of $\mathbf{w}$ (Fig. 1(a) and (b)) shows that the data distribution is highly irregular and hence hard to describe analytically. Upon closer inspection of the mapping network M, however, we note that the very last processing step before computing $\mathbf{w}$ is a Leaky ReLU (LRU) with a fixed negative slope; in the case of StyleGANv2, the slope is $\nu = 0.2$. Undoing this effect is as simple as passing $\mathbf{w}$ through another LRU with negative slope $\nu = 5.0$ (from here on, we will denote Leaky ReLUs as LRU_ν). This yields $\mathbf{v} = \text{LRU}_{5.0}\left(\mathbf{w}\right)$; we denote the mapped space as $\mathcal{V}$. Plotting marginals of the distribution of $\mathbf{v} \in \mathcal{V}$ shows very strong Gaussian characteristics (Fig. 1(c) and (d)). We can therefore model the distribution $p(\mathbf{v})$ on $\mathcal{V}$ as a high-dimensional Gaussian, and describe $p\left(\mathbf{v}\right)$ using the empirical covariance matrix $\mathbf{\Sigma}$ and mean $\boldsymbol{\mu}$ fitted to samples $\mathbf{v} = \text{LRU}_{5.0}\left(M\left(\mathbf{z}\right)\right)$. In the remainder of this paper, we show that this Gaussian model can improve results in two independent applications, inversion of real images (Section 3) and removal of artifacts in generated images (Section 4).

3 Improving image inversion using the Gaussian prior

The goal of inversion is to find a point in latent space from which a given (real or generated) image can be reconstructed by the generator as accurately as possible. If semantically meaningful directions in latent space are known, the latent can then be modified (for example to change the facial expression or camera angle) and the image re-generated with the modifications, without the user having to touch the actual pixels. While approaches exist that directly predict the latent using a trained model [3], inversion is commonly formulated as a continuous optimization problem of the form

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}' \in \mathcal{W}} L(I, G(\mathbf{w}')), \tag{1}$$

where $L(\cdot)$ is some image reconstruction loss, for example the LPIPS perceptual distance [19] between the input image I and the generated image $G(\mathbf{w}')$ [13].

After computing the Gaussian model of $p\left(\mathbf{v}\right)$ as described in the last section, it is now trivial to plug this as a prior into Eq. (1) by converting it to an energy term, yielding an estimate $\hat{\mathbf{w}}_p$,

$$\hat{\mathbf{w}}_p = \arg\min_{\mathbf{w}' \in \mathcal{W}} L(I, G(\mathbf{w}')) + \lambda \left(\mathbf{v}' - \boldsymbol{\mu}\right)^\top \mathbf{\Sigma}^{-1} \left(\mathbf{v}' - \boldsymbol{\mu}\right), \tag{2}$$

with $\mathbf{v}' = \text{LRU}_{5.0}(\mathbf{w}')$ mapping from the original latent space $\mathcal{W}$ to the Gaussianized latent space $\mathcal{V}$, the empirical covariance matrix $\mathbf{\Sigma}$ and mean $\boldsymbol{\mu}$ computed as described above, and the weight λ determined empirically as $\lambda = 10^{-4}$. As in [13], in both cases (with and without the prior) we solve the inversion using ADAM as an optimizer, a learning rate of 0.1, add ramped-down noise to the estimated latent, and run the optimization for 1000 iterations.

As a second baseline, and to be able to better invert natural images, we modify this to project into the extended latent space $\mathcal{W}^+$, yielding the prior-free optimization

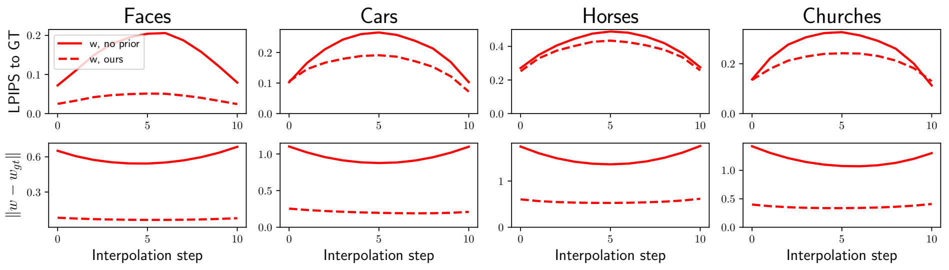
$$\hat{\mathbf{w}}_{(+)} = \arg\min_{\mathbf{w}'_{(+)} \in \mathcal{W}^+} L(I, G(\mathbf{w}'_{(+)})), \tag{3}$$

and, adding the prior,

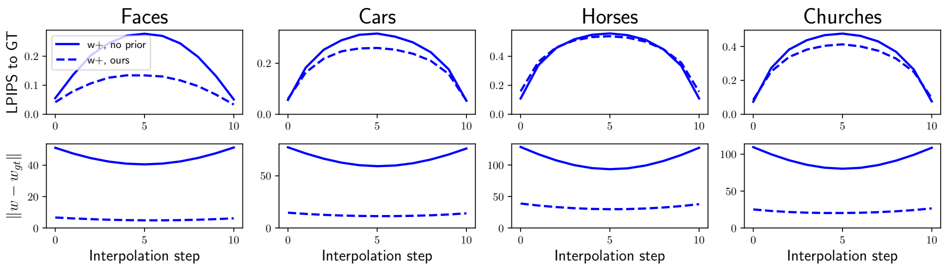
$$\hat{\mathbf{w}}_{(+),p} = \arg\min_{\mathbf{w}'_{(+)} \in \mathcal{W}^+} L(I, G(\mathbf{w}'_{(+)})) + \lambda \left(\mathbf{v}'_{(+)} - \boldsymbol{\mu}_{(+)}\right)^\top \mathbf{\Sigma}^{-1}_{(+)} \left(\mathbf{v}'_{(+)} - \boldsymbol{\mu}_{(+)}\right). \tag{4}$$

Here, as above, $\mathbf{v}'_{(+)} = \text{LRU}_{5.0}(\mathbf{w}'_{(+)})$, $\boldsymbol{\mu}_{(+)}$ is $\boldsymbol{\mu}$ stacked s times (where s is the number of style scales for a given generator), $\mathbf{\Sigma}_{(+)} = \mathbf{I}_s \otimes \mathbf{\Sigma}$, and, in a slight abuse of notation, we allow the generator G to map from both $\mathcal{W}$ and $\mathcal{W}^+$ to images. The optimization parameters are the same as above, with the difference of a lower learning rate of 0.05 and a longer optimization of 10K iterations. Note that, since we investigate the properties of the latent space, we do not optimize the noise maps, but keep them fixed to the noise learned during training.

[3]In case of $\mathcal{W}^+$, we impose our prior on each of the different styles, see the following section.



(a) Reconstruction errors for images (top) and latents (bottom) when optimizing to $\mathcal{W}$.



(b) Reconstruction errors for images (top) and latents (bottom) when optimizing to $\mathcal{W}^+$.

Figure 2: Average image and latent reconstruction errors. Adding the prior (dashed lines) helps both when reconstructing to $\mathcal{W}$ (top, red) and $\mathcal{W}^+$ (bottom, blue). Note that we match the model to the data, *i.e*. the images in (a) were generated using a single style, and the images in (b) were generated using different styles on different scales.

Table 1: Reconstruction errors on real images (R), images generated from $\mathcal{W}$ (G), and from $\mathcal{W}^+$ (G+)

	Faces			Cars			Horses			Churches		
Method	R	G	G+	R	G	G+	R	G	G+	R	G	G+
$\mathcal{W}$	0.29	0.07		0.22	0.12		0.34	0.27		0.32	**0.11**	
$\mathcal{W}$ + prior	0.35	**0.05**		0.27	**0.04**		0.40	**0.22**		0.39	0.14	
$\mathcal{W}^+$	**0.15**		0.06	**0.11**		0.06	**0.15**		**0.11**	**0.13**		**0.07**
$\mathcal{W}^+$ + prior	0.20		**0.05**	0.15		**0.05**	0.22		0.17	0.21		0.09

3.1 Experimental setup

We test our inversion method on four classes of images (faces, cars, horses, and churches), and use the pre-trained generator weights provided by [13]. For each class, we use three conditions with 20 images each: real images, images generated using a single style, and images generated using different styles on different scales. We then invert all images using the approaches described above.

Table 2: Latent reconstruction errors on images generated from $\mathcal{W}$ (G) and $\mathcal{W}^+$ (G+)

	Faces		Cars		Horses		Churches	
Method	G	G+	G	G+	G	G+	G	G+
$\mathcal{W}$	18.00		32.53		50.14		37.69	
$\mathcal{W}$ + prior	**3.10**		**5.58**		**20.40**		**15.66**	
$\mathcal{W}^+$		227.65		316.28		493.26		414.96
$\mathcal{W}^+$ + prior		**31.02**		**59.64**		**151.23**		**101.55**

Table 3: User study of interpolation quality (given in percent preference)

Dataset	Inversion without prior	Inversion with prior (ours)
Faces	10.1%	**89.9%**
Cars	34.2%	**65.8%**
Horses	32.6%	**67.4%**
Churches	42.9%	**57.1%**
All	29.8%	**70.2%**

3.2 Results

Quantitative results. Table 1 shows the image reconstruction errors [19] for real and generated images; for generated images, Table 2 shows the L_2 distances in latent space between the estimated latent and the true latent. Note that, when using generated images, we match the model to the data, *i.e.* when inverting images that were generated from a single style, we invert to $\mathcal{W}$; otherwise, we invert to $\mathcal{W}^+$. As can be seen in Table 1, our method slightly decreases raw reconstruction performance for real images; this is to be expected, since without a prior, the model overfits to the data and hence achieves better reconstruction. However, when comparing the error of the latent in Table 2, our method finds latents that are significantly closer to the true latents.

Figure 2 shows the implications of this. Here, we sample 80 pairs per condition and class, interpolate between the generated images using the ground truth latents, and compare these ground truth interpolations to the interpolations between the estimated latents. As can be seen, imposing the prior significantly improves performance in the middle points of the interpolations, both in case of optimizing to $\mathcal{W}$ (red) and optimizing to $\mathcal{W}^+$ (blue). That this is the case even for the classes "Horse" and "Church" in Fig. 2(b), where the prior causes worse inversion performance for the start and end images, indicates that the prior causes the inversion to find points in the latent space that are better behaved and overfit less to the input data.

Qualitative results. Figure 3 shows this effect visually on interpolations between real images. As can be seen, inverting without our prior often leads to interpolations that contain unrealistic appearances, shapes, and color artifacts (top rows); when inverting with the prior (bottom rows), the interpolations stay realistic.

User study. To quantify the impact of the prior on interpolation quality, we perform a user study. We first invert all real images (20 each for faces, cars, horses, and churches) to $\mathcal{W}^+$, both with and without prior. For all pairs within the same condition (with or without prior), we then compute the interpolated center image by generating the image from the midpoint of their $\mathbf{w}_{(+)}$. The center images for inversions with and without prior are then shown side-by-side in shuffled order, and users are asked which of the two images they judge to be more realistic. In total, $n = 22$ subjects took part in the study, resulting in 1388 judgements. Table 3 shows the results. In all cases, interpolations from inversions with prior (*i.e.* our method) are preferred over interpolations from inversions without prior, and overall, our prior is preferred in 70% of all cases. For faces, around 90% of users prefer our prior. For the other datasets, preference is somewhat lower. We believe this is because the quality of generated images is lower for cars, horses, and churches, occasionally resulting in unrealistic interpolations for both methods. Furthermore, in some instances it is hard to judge the realism, for example, whether a church has zero, one, or two spires does not influence how realistic the image is perceived to be.

4 Removing image artifacts using the Gaussian prior

Leaving the subject of inversion behind, we ask whether our Gaussian model can help us better understand the behavior of the GAN. Modeling the data distribution in the latent space as Gaussian allows us to probe this behavior using methods that assume that the underlying data is Gaussian distributed, such as Principal Component Analysis (PCA). Here, we show how this helps us to identify and remove artifacts from generated images while maintaining diversity, which is crucial in the context of face image generation.

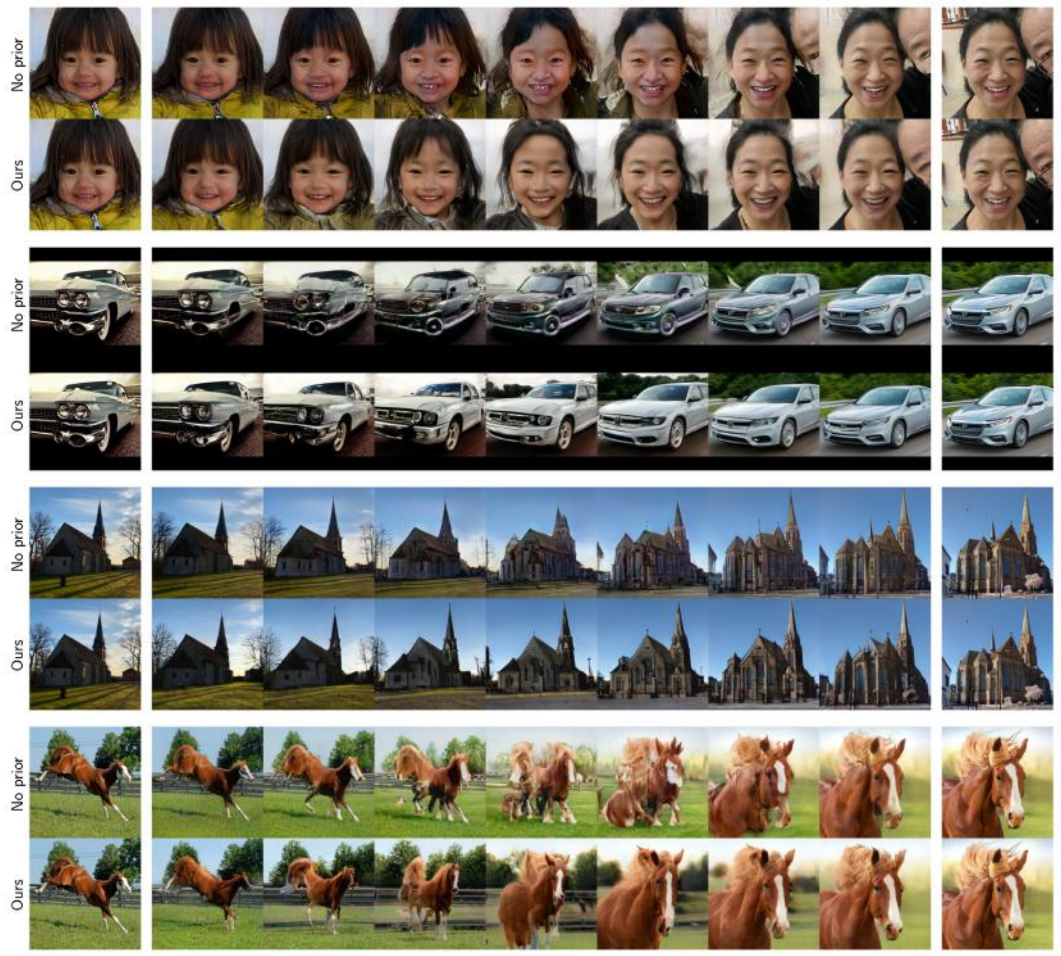


Figure 3: Example interpolations between inversions of real images to $\mathcal{W}^+$. Without a prior (top row in each example), the latents often fall into poorer areas of the latent space, causing distorted appearances in the intermediate images. Using a prior (bottom rows) encourages the latent to lie in good regions, causing the intermediate images to be more realistic. The left and right columns show the start and end frames, respectively. Please see the appendix for additional examples.

4.1 Analyzing artifacts

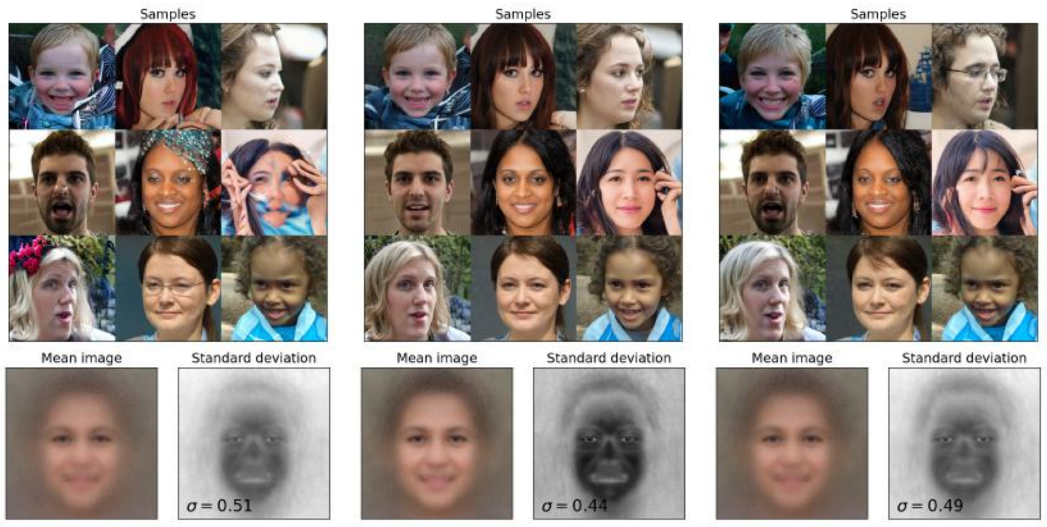
While StyleGAN-based architectures generally produce outputs of a high visual quality, they often still contain a number of artifacts. In Fig. 4(a), examples are the distorted occluder of the face (middle row, right) or artificial flower-like patches on the head (bottom row, left).

A common trick to improve visual appearance and remove such artifacts is *truncation*[4]. For a given input $\mathbf{z}$ and its corresponding $\mathbf{w} = M(\mathbf{z})$, a truncated latent $\tilde{\mathbf{w}}$ is computed by moving $\mathbf{w}$ towards the empirical mean $\bar{\mathbf{w}}$ by a factor of ψ,

$$\tilde{\mathbf{w}} = \bar{\mathbf{w}} + \psi(\mathbf{w} - \bar{\mathbf{w}}) = \psi\mathbf{w} + (1-\psi)\bar{\mathbf{w}}. \tag{5}$$

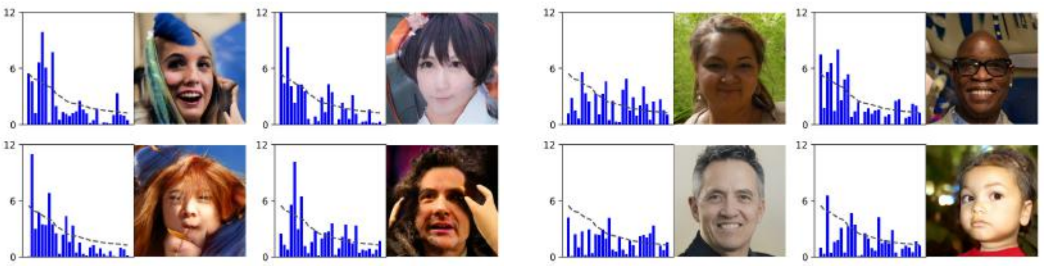
The resulting $\tilde{\mathbf{w}}$ is then used as the input to the generator network. As shown in Fig. 4(b), this removes virtually all of the artifacts. At the same time, visual diversity is reduced. The generated faces are closer to a canonical, frontal pose, the lighting is very flat, there is much less facial hair, and the skin is lighter. This can also be seen when comparing the mean images and per-pixel standard deviations (Fig. 4, bottom); when using truncation, the mean image is sharper and the per-pixel

[4]Note that, mathematically, this process does not truncate individual values, but compresses them towards the mean. However, for consistency with previous work, we keep the terminology introduced in [12]



(a) Uncorrected output (b) Truncation, $\psi = 0.7$ (c) Ours, $\tau = 0.5$

Figure 4: Comparison of different correction methods. Raw samples (a) are commonly corrected using truncation (b), which removes artifacts, but also reduces diversity, as can be seen from the sharper average image and lower per-pixel standard deviations in (b). Our method (c) reduces artifacts, while maintaining a high degree of diversity.



(a) Images with artifacts and magnitudes of their first 30 principal components (b) Images without artifacts and magnitudes of their first 30 principal components

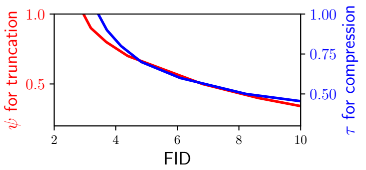
Figure 5: Principal components of images with and without artifacts. Images with artifacts exhibit significantly larger magnitudes in the low principal components than good images; the dashed lines indicate one standard deviation.

standard deviation is lower, especially in regions where skin is visible. This is obviously a problem when trying to capture the diversity of humans and their appearances.

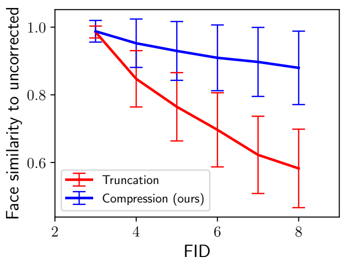
To break this tradeoff between artifacts and diversity, we investigate if it is possible to remove artifacts in a targeted manner, leaving images without artifacts untouched. Since we model $p(\mathbf{v})$ as Gaussian, we can compute the main axes of variation in $\mathcal{V}$ using Principal Component Analysis (PCA), which assumes the underlying data to be Gaussian distributed. Projecting latents into the PCA space (we denote the projection of $\mathbf{v}$ as $\mathbf{v}^p$) reveals that latents that generate images with artifacts exhibit significantly larger values, especially in the lowest dimensions (*i.e*. those with the largest variation). Figure 5 shows a comparison of the values in the first 30 dimensions of $\mathbf{v}^p$ for images with artifacts (Fig. 5a) and without artifacts (Fig. 5b). Note that the latents generating images with artifacts are still valid samples from a Gaussian; the artifacts are merely hiding in the tails of the distribution.

4.2 Reducing artifacts by logarithmic compression

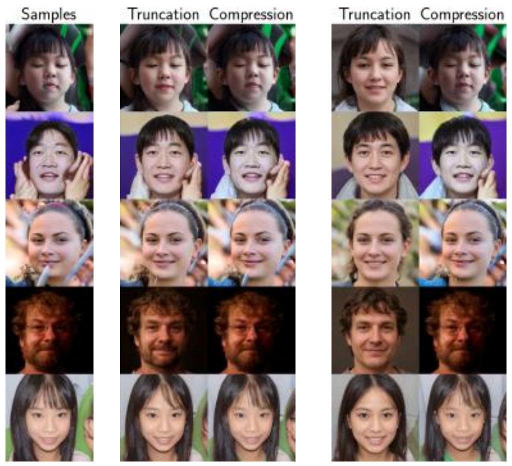
To reduce the impact of these large components, we propose to logarithmically compress values with a magnitude larger than a threshold $\tau\sigma$, where τ is a scaling factor and $\sigma = \max_i \sigma_i = \max_i \sqrt{\lambda_i}$ is



(a) Correction parameters vs. FID



(b) Face distance vs. correction strength



(c) Samples (d) FID = 4 (e) FID = 8

Figure 6: Comparison of correction methods. (a) The impact of the truncation parameter ψ and the compression parameter τ on the FID is very similar. (b) At the same FID, our method better preserves facial identity. (c) shows a random selection of samples, (d-e) compare visual quality at the same FID values for truncation and compression (left and right columns, respectively). Both methods are effective at removing artifacts, but our method better preserves diversity.

the maximum standard deviation along the principal components, corresponding to the square root of the maximum eigenvalue of $\mathbf{\Sigma}$,

$$\tilde{v}_i^p = \begin{cases} \operatorname{sign}\left(v_i^p\right) \tau\sigma \left[\log\left(\frac{|v_i^p|}{\tau\sigma}\right) + 1\right] & \text{if } \ |v_i^p| > \tau\sigma \\ v_i^p & \text{otherwise ,} \end{cases} \tag{6}$$

where v_i^p, $\tilde{v}_i^p$ are the i-th component of $\mathbf{v}^p$ and $\tilde{\mathbf{v}}^p$, respectively. After thus computing $\tilde{\mathbf{v}}^p$, we reproject $\tilde{\mathbf{v}}^p$ into $\mathcal{W}$ as $\tilde{\mathbf{w}} = \text{LRU}_{0.2}\left(\mathbf{E}\tilde{\mathbf{v}}^p + \boldsymbol{\mu}\right)$, where the columns of $\mathbf{E}$ contain the eigenvectors of $\mathbf{\Sigma}$, and generate the image from $\tilde{\mathbf{w}}$ as before.

4.3 Results

To be able to compare our proposed compression with the commonly used truncation method, we first need to align their parameters (ψ for truncation and τ for compression). Here, we use the Fréchet Inception Distance [10] (FID, computed using 50K samples) to the training set as an alignment target, that is, we align ψ and τ so that the corresponding corrected outputs have comparable FIDs to the training set. Figure 6(a) illustrates the concept, and shows that the overall shape of the curve is similar. Figure 6(c-e) show visual example for truncation and our compression method at matched FIDs. As we move away from the training distribution, truncation causes diversity to decrease. In the case of compression, on the other hand, the distance to the training set is caused by a lack of artifacts. Interestingly, we note that for the same FID, our images are subjectively closer to the uncorrected image. This indicates that FID in itself is not a particularly good metric to evaluate the presence or absence of artifacts in the image generation process and is unable to distinguish for example real makeup from failed attempts to draw makeup.

Measuring face distortion. To provide a different perspective on the effects of correction, we investigate how either truncation or compression changes the identity of generated faces. We sample 1024 images of faces without correction and correct them using both methods, with parameters matched to the same FIDs. We then project all images into a face embedding space using the `SE-ResNet-50-128D` model from VGGFace2 [6], and compute the cosine similarity between the uncorrected and corrected images in the embedding space. Figure 6(b) shows the results. In case of truncation (red), image correction significantly changes the identity of the samples, and the similarity decreases. In contrast, our method (blue) preserves identity better, even with strong corrections.

5 Conclusion

In this work, we propose to analytically describe the distribution of data in a learned latent space of a Generative Adversarial Network. We show how, under a simple nonlinear transformation (an element-wise leaky ReLU), the distribution in the latent space can be modeled as a Gaussian in closed form. We have demonstrated the benefits in two scenarios. First, we have shown that the Gaussian distribution can serve as an effective prior for the task of image inversion, which significantly increases the accuracy of the recovered latent vector. Second, we have shown that our Gaussian model allows us to analyze the cause of artifacts in the image generation pipeline and improve upon the common "truncation" trick by compressing principal components with large magnitudes. This effectively removes artifacts, while keeping a higher degree of visual diversity compared to truncation.

6 Acknowledgements

This work was supported by a grant from Intel Corp. and a GPU gift from NVIDIA.

References

[1] Abdal, R., Qin, Y., Wonka, P.: Image2stylegan++: How to edit the embedded images? arXiv preprint arXiv:1911.11544 (2019)

[2] Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the stylegan latent space? In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)

[3] Bau, D., Strobelt, H., Peebles, W., Wulff, J., Zhou, B., Zhu, J.Y., Torralba, A.: Semantic photo manipulation with a generative image prior. ACM Trans. Graph. **38**(4) (Jul 2019). https://doi.org/10.1145/3306346.3323023, `https://doi.org/10.1145/3306346.3323023`

[4] Bau, D., Zhu, J.Y., Strobelt, H., Zhou, B., Tenenbaum, J.B., Freeman, W.T., Torralba, A.: Visualizing and understanding generative adversarial networks. In: International Conference on Learning Representations (2019)

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

[6] Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: Vggface2: A dataset for recognising faces across pose and age. In: International Conference on Automatic Face and Gesture Recognition (2018)

[7] Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. arXiv preprint arXiv:1605.09782 (2016)

[8] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)

[9] Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A.A., Visin, F., Vazquez, D., Courville, A.: Pixelvae: A latent variable model for natural images. arXiv preprint arXiv:1611.05013 (2016)

[10] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 6626–6637. Curran Associates, Inc. (2017)

[11] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations (2018)

[12] Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

[13] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of StyleGAN. arXiv preprint arXiv:1912.04958 (2019)

[14] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)

[15] Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)

[16] Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of gans for semantic face editing. arXiv preprint arXiv:1907.10786 (2019)

[17] Voynov, A., Babenko, A.: Unsupervised discovery of interpretable directions in the gan latent space. arXiv preprint arXiv:2002.03754 (2020)

[18] Yang, C., Shen, Y., Zhou, B.: Semantic hierarchy emerges in deep generative representations for scene synthesis. arXiv preprint arXiv:1911.09267 (2019)

[19] Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)

[20] Zhu, J.Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: Proceedings of European Conference on Computer Vision (ECCV) (2016)

7 Appendix

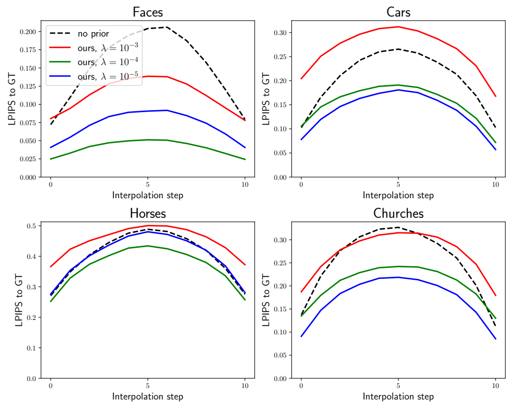
7.1 Broader impact

In this work, we investigate the computational properties of Generative Adversarial Networks, and their impact on inversion and output quality. While our primary goal is to understand GANs from a computational standpoint, both applications carry undeniable risks. First, improving the quality of inversion could be desirable from an artistic standpoint, but can also be problematic if the subject whose image is inverted does not give consent. Since our method improves inversion quality, modified images will also be more realistic, and therefore potentially harder to distinguish from real images. Further work on detecting synthetic images is therefore paramount. Similarly, artifacts are often indicative of synthetic images, and removing them might make such images harder to identify.

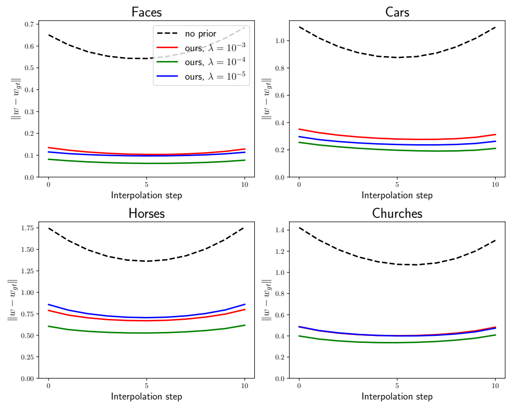
However, a number of positive impacts of the the methods presented here are also possible. For example, explicitly characterizing the distribution on the latent space could make it possible to develop better methods to detect synthetic images, by evaluating whether an inversion is likely to come from the latent distribution. Second, increasing inversion quality could lead to a shift in focus in research on GANs, concentrating more on aspects of representation learning and less on generating better and more accurate fake images. Lastly, we present a method to reduce artifacts on synthetic images without sacrificing diversity. If a GAN is used to generated synthetic training data for a downstream application, a lack in diversity is problematic, and can lead for example to systems that achieve much better classification performance for subjects with lighter skin color than for subjects with darker skin color. Using our method as an alternative to the truncation trick could be a way to circumvent this issue.

7.2 Hyperparameter tests

Here, we give results for interpolation performance (similar to Figure 2 in the primary paper) for different prior weights λ. While the best parameter differs between datasets, for consistency we choose $\lambda = 10^{-4}$. As can be seen from the graphs, this value provides a good compromise between interpolation performance and reconstruction performance at the endpoints (*i.e.* raw inversion performance). In Fig. 8(a), for example, it can be seen that $\lambda = 10^{-3}$ (red curve) often provides better performance at the interpolation midpoint; however, in particular for Cars and Horses, this choice would provide a significantly worse inversion quality.

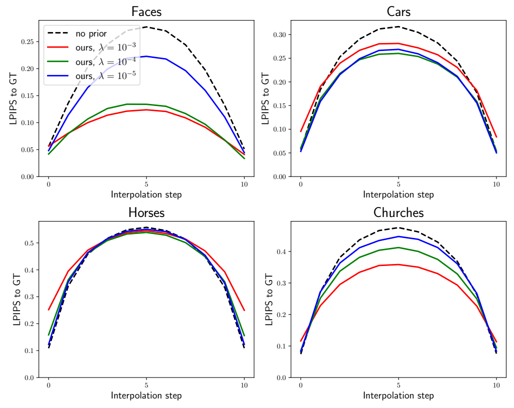


(a) Reconstruction errors for images when optimizing to $\mathcal{W}$.

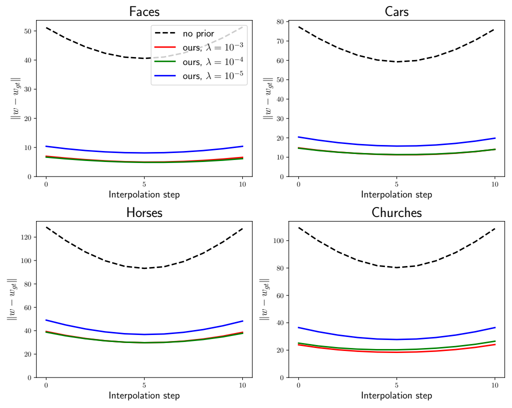


(b) Reconstruction errors for latents when optimizing to $\mathcal{W}$.

Figure 7: Average image and latent reconstruction errors when optimizing in $\mathcal{W}$ for different values for prior weight λ. The dashed line indicates the results when using an unconstrained optimization as described in [13].



(a) Reconstruction errors for images when optimizing to $\mathcal{W}^+$.



(b) Reconstruction errors for latents when optimizing to $\mathcal{W}^+$.

Figure 8: Average image and latent reconstruction errors when optimizing in $\mathcal{W}^+$ for different values for prior weight λ. The dashed line indicates the results when using an unconstrained optimization as described in [13].

7.3 Additional interpolation results

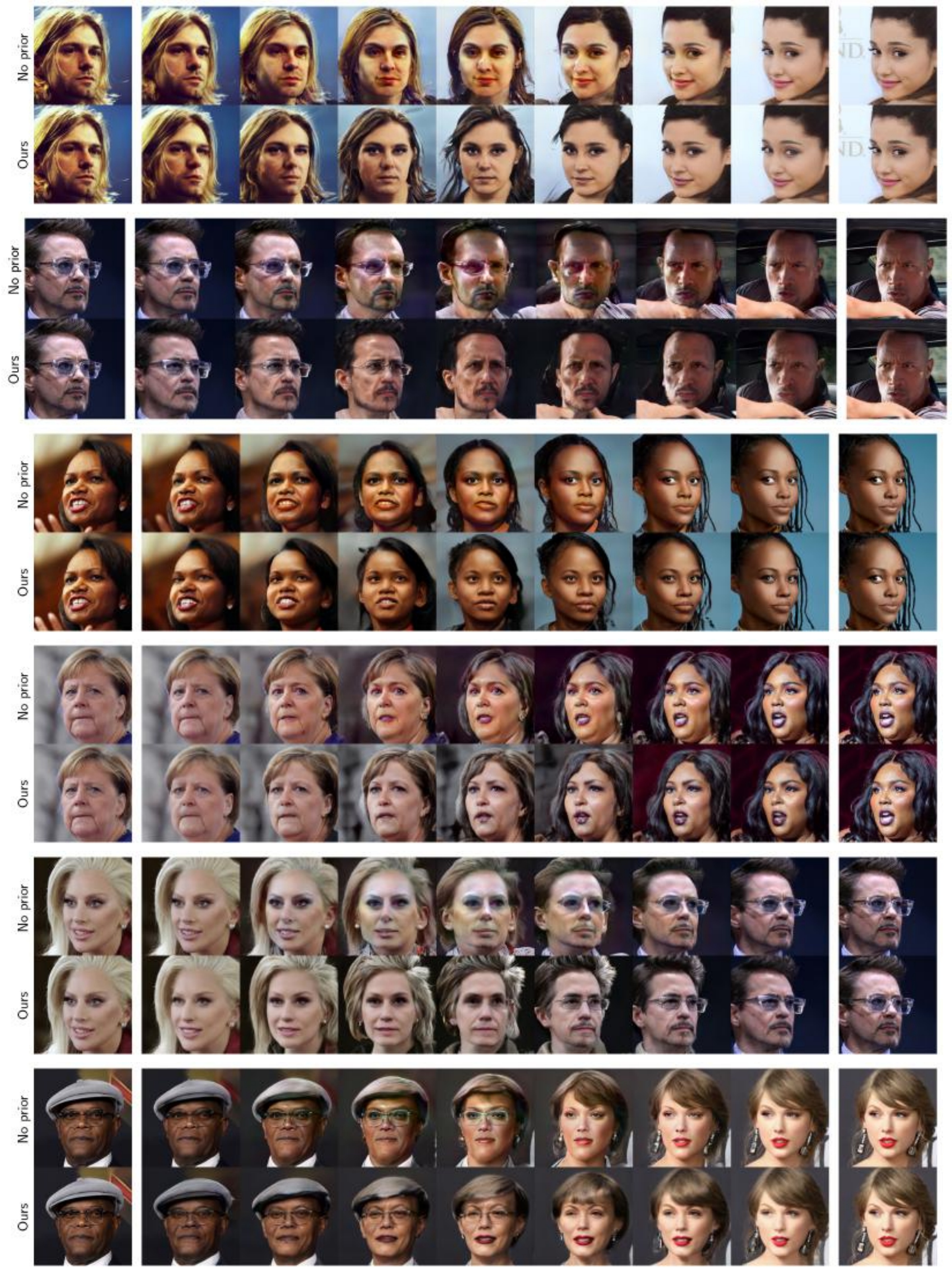


Figure 9: More examples for face interpolations. In each example, the top row shows an unconstrained optimization as described in [13], but to $\mathcal{W}^+$ [2]. The bottom row shows the results when using our proposed prior on the Gaussianized latent space $\mathcal{V}^+$.

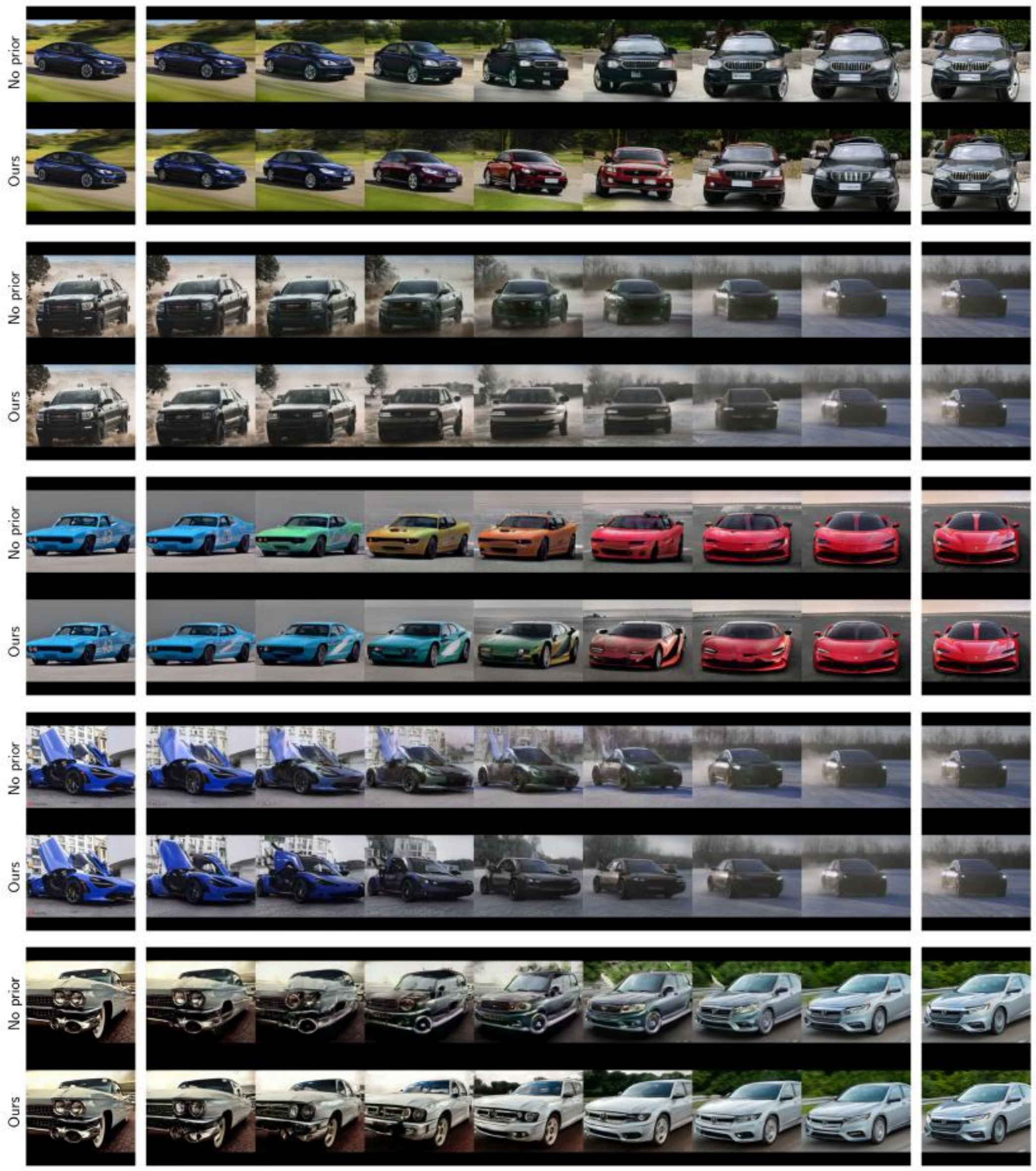


Figure 10: Examples for car interpolations. In each example, the top row shows an unconstrained optimization as described in [13], but to $\mathcal{W}^+$ [2]. The bottom row shows the results when using our the proposed prior on the Gaussianized latent space $\mathcal{V}^+$.

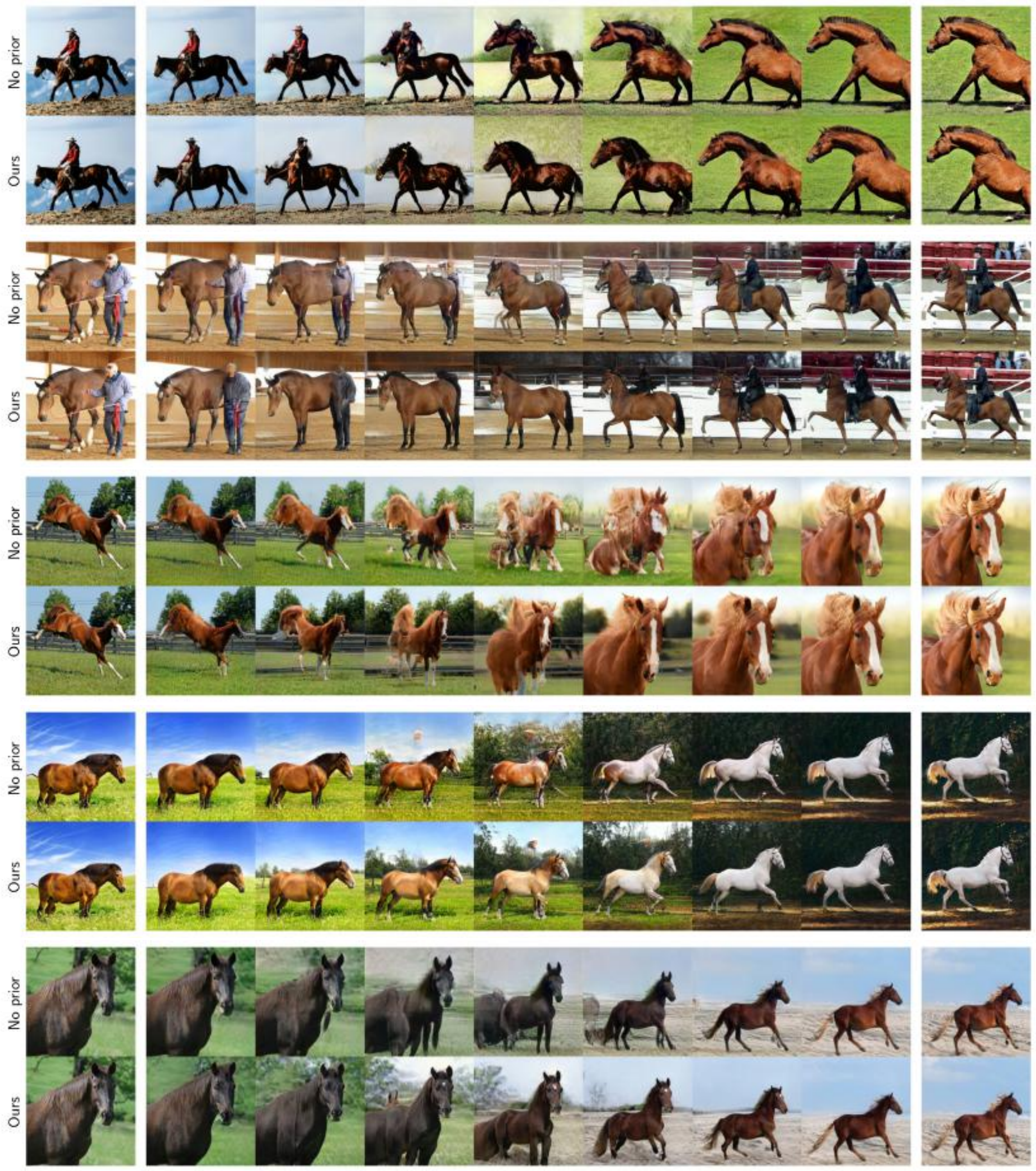


Figure 11: Examples for horse interpolations. In each example, the top row shows an unconstrained optimization as described in [13], but to $\mathcal{W}^+$ [2]. The bottom row shows the results when using our the proposed prior on the Gaussianized latent space $\mathcal{V}^+$.

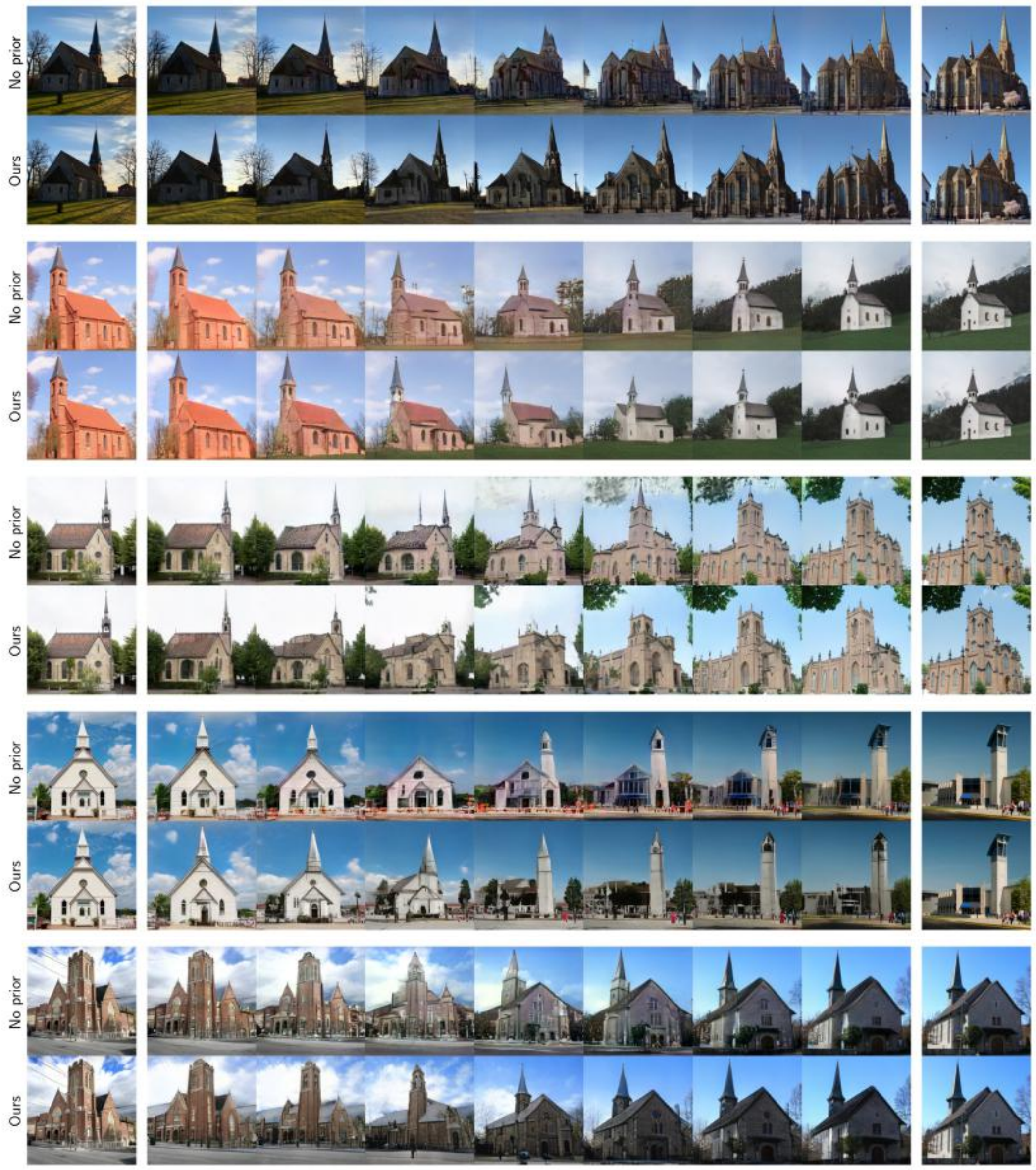


Figure 12: Examples for church interpolations. In each example, the top row shows an unconstrained optimization as described in [13], but to $\mathcal{W}^+$ [2]. The bottom row shows the results when using our the proposed prior on the Gaussianized latent space $\mathcal{V}^+$.